WEB APPLICATION SECURITY ASSESSMENT REPORT

Internship Project – Future Interns

Test Application: Altoro Mutual (https://demo.testfire.net)

Tool Used: UpGuard Web Scanner

Prepared By: Akansha Bhagat

Date: 14th June 2025

Table of Contents

Introduction

Objective of the Assessment

Tools & Technologies Used

Methodology

Identified Vulnerabilities

OWASP Top 10 Mapping

Risk Mitigation Strategies

Conclusion

1. Introduction

Web applications are a common target for attackers due to their exposure on the internet and often weak configurations. In this internship project, a security assessment was conducted on the demo web application Altoro Mutual, an intentionally vulnerable banking website hosted at https://demo.testfire.net .

The primary aim was to analyze this application's security posture using a free online vulnerability scanning tool, identify potential flaws, and present mitigation recommendations. This report outlines the steps taken, the vulnerabilities discovered, and how they align with OWASP's Top 10 web application security risks.

The results of this project reflect areal-world scenario where ethical hackers and cybersecurity professionals evaluate applications to ensure they are secure and resilient against modern threats.

2. Objective

The objective of this project was to:

- Perform a security scan on a sample web application using beginner-friendly tools.
- Identify misconfigurations or security flaws in the web application's architecture.
- Simulate how attackers might exploit these weaknesses.
- Create a formal Security Assessment Report for the internship portfolio.
- Align findings with OWASP Top 10 to understand standard attack surfaces.
- Recommend relevant mitigation steps to improve the security posture.

This simulation mimicked how cybersecurity consults or analysts would evaluate a client's website and prepare a vulnerability report.

3. Tools & Technologies Used

The following tools and technologies were used during the course of this assessment:

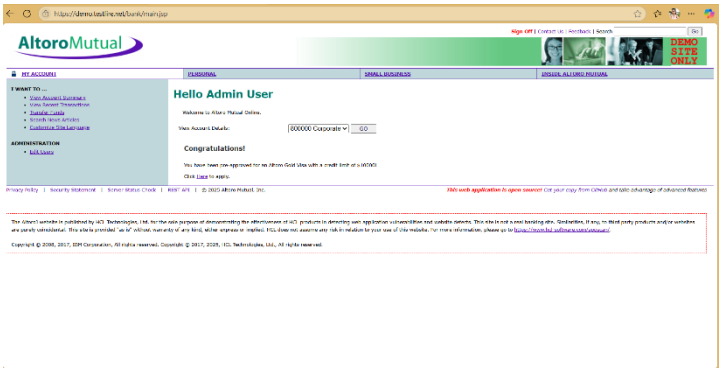
- **Altoro Mutual:** A purposely vulnerable online banking site used for educational and security training.
- **UpGuard Online Web Scanner:** A browser-based vulnerability scanning tool used to assess HTTP headers, cookies, and security policies.
- **Microsoft Edge Browser:** Used to manually navigate and interact with the application.
- **Snipping Tool:** Used to capture visual evidence for documentation.
- **Microsoft Word:** Used for compiling this formal assessment report.

4. Methodology

The steps taken to complete the vulnerability assessment are as follows:

1. Access to Web Application:

Visited https://demo.testfire.net in Microsoft Edge and explored the login and dashboard pages using demo credentials.



2. Security Scanner Launch:

Used the UpGuard Online Security Scanner and entered the Altoro Mutual URL to initiate a vulnerability scan.

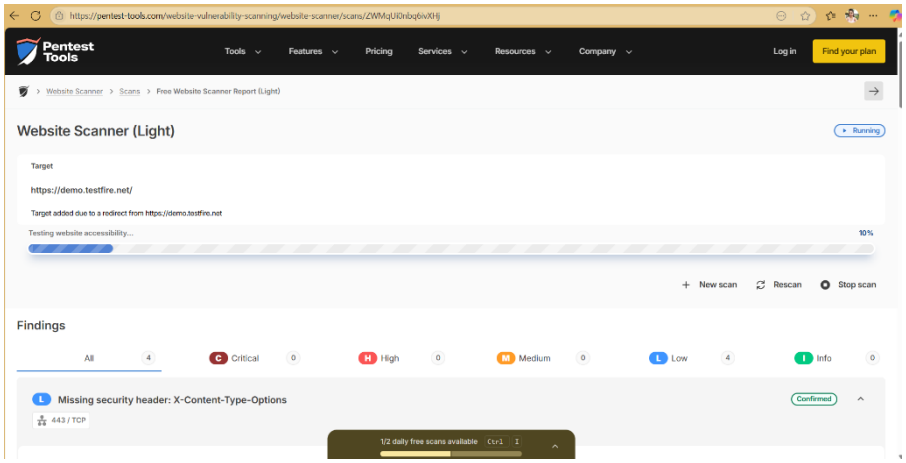


Figure 2: Live scanning in progress on UpGuard while testing Altoro Mutual

3. Automated Scan Analysis:

The scanner completed a full check of public headers, cookies, and page responses in under 30 seconds.

4. Findings Compilation:

The report provided by UpGuard was downloaded, and screenshots were taken of both the scan and the application.

5. Report Writing:

The vulnerabilities were interpreted and matched to OWASP Top 10 categories. Mitigation recommendations were then written to complete the assessment.

Note: While the majority of findings were obtained through automated scanning using the UpGuard Security Scanner, manual verification was also performed to identify missing files like robots.txt and security.txt.

These configurations, although not flagged in summary report, are important for responsible disclosure and crawler restrictions.

They are recognized under OWASP A05:2021 – Security Misconfiguration and were confirmed through direct browser-based testing.

5. Identified Vulnerabilities

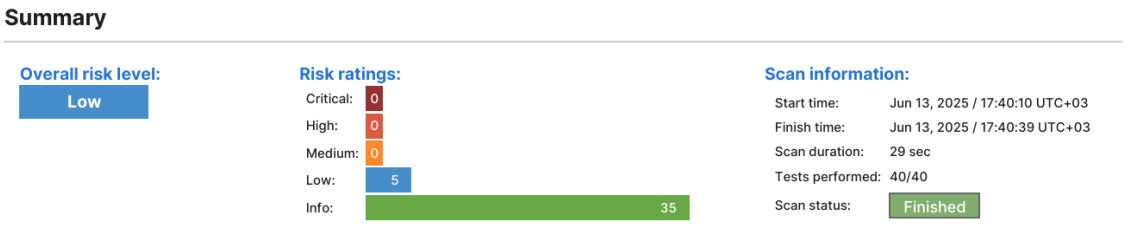


Figure 3: UpGuard scanner summary highlighting five low-risk and thirty-five informational issues.

The scan results showed no high or critical vulnerabilities. However, several important low-risk and informational issues were found. The following are the vulnerabilities:

🔴 Missing HTTP Security Headers

Description:

Altoro Mutual's server does not implement several important HTTP headers like:

- Strict-Transport-Security
- Content-Security-Policy
- Referrer-Policy
- X-Content-Type-Options

These headers are necessary to prevent attacks such as clickjacking, XSS, and leakage of sensitive information via referrer headers.

Impact:

Without these headers, users are exposed to potential threats like content injection or insecure redirects. Modern browsers cannot enforce policies that limit dangerous behaviors.

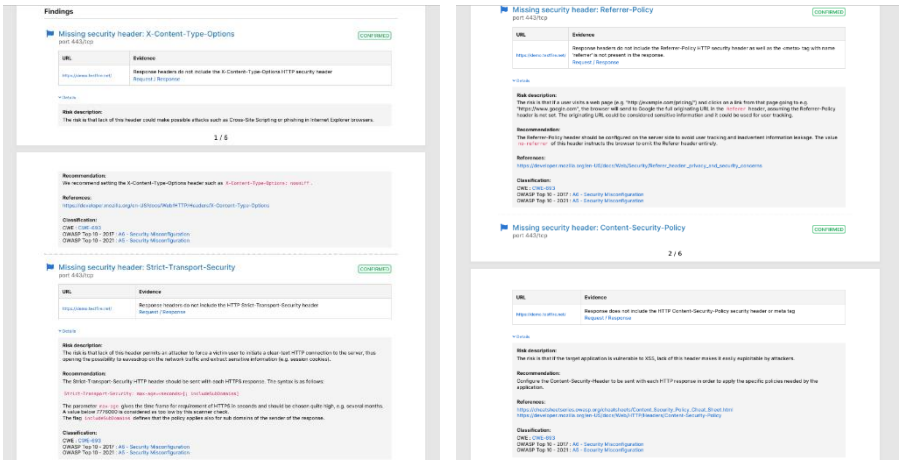


Figure 4: Scanner report showing absence of critical headers such as CSP and HSTS

OWASP Mapping: A05:2021 – Security Misconfiguration

Mitigation:

Set these headers using your web server configuration or framework.

For instance, on Apache:

Header always set X-Content-Type-Options "nosniff"

Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains"

Header always set Content-Security-Policy "default-src 'self'"

📝Absence of security.txt File for Responsible Disclosure

Description:

The application does not host a security.txt file in its. Well-known directory. This file is used to inform security researchers how to responsibly disclose vulnerabilities, including contact information and policies.

Impact:

Without this file, security researchers or ethical hackers may be unaware of how to report issues responsibly. This could result in delayed reporting or even public disclosure of security flaws.

OWSP Mapping:

A05:2021 – Security Misconfiguration

Mitigation:

Create a plain text file at:

https://demo.testfire.net/.well-known/security.txt

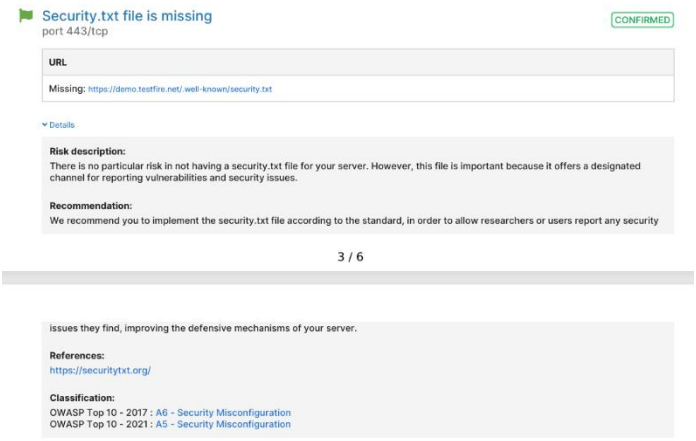


Figure 5: Scanner reporting the absence of a responsible disclosure security.txt file in the expected location

Include your responsible disclosure email, policy, and preferred format.

Example:

Contact: mailto:security@example.com

Preferred-Languages: en

Policy: https://example.com/disclosure-policy

🤖 Missing robots.txt File (Search Engine Control)

Description:

The application does not provide a robots.txt file at its root directory (https://demo.testfire.net/robots.txt). This file is used to control how web crawlers and

search engines index a website. It is important for protecting admin panels, backup directories, or sensitive endpoints from being listed on search engines.

Impact:

Without robots.txt file, internal paths or sensitive areas may be exposed to search engine indexing. This could lead attackers to discover entry points, admin areas, or hidden files simply by searching online.

OWASP Mapping: A05:2021 – Security Misconfiguration

Mitigation:

Create a plain text robots.txt file and host it at:

https://demo.testfire.net/robots.txt

Example content:

User-agent: *

Disallow: /admin/

Disallow: /test/

Disallow: /backup/

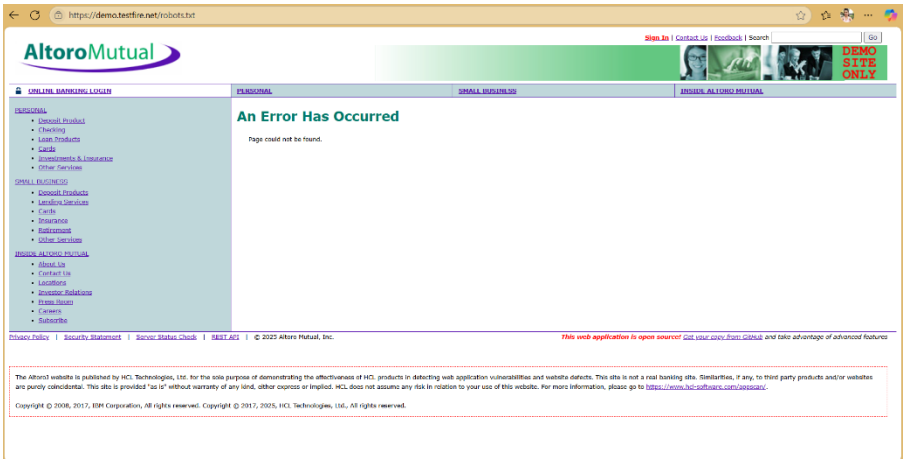


Figure 6: Attempting to access robots.txt file returns error, confirming absence of crawler control configuration

6. OWASP Top 10 Mapping

OWASP Risk ID	Title	Mapped Finding
A05:2021	Security Misconfiguration	Missing headers, insecure cookies
A06:2021	Vulnerable and Outdated	Server info exposed in response

	Components	headers
A07-A10	-	Not applicable in this scan

Note: No injection, auth, or logic flaws were detected in this scan.

7. Risk Mitigation Strategies

To reduce the application's exposure to threats:

- Implement all missing HTTP security headers (CSP, HSTS, etc.)
- Use HTTPS exclusively and enforce it via Strict-Transport-Security
- Set Secure and HttpOnly attributes for all cookies
- Hide technology banners and headers that reveal backend systems
- Regularly perform automated and manual testing on staging environments

8. Conclusion

The web application was found to have no critical or high vulnerabilities, but several low-risk security misconfigurations were detected. These issues, although not immediately exploitable, indicate a lack of secure configuration practices.

The assessment successfully fulfilled the internship objective by:

- Using a real-world vulnerable app
- Running a vulnerability scan
- Interpreting technical issues
- Compiling a structured report with mitigation advice

This report demonstrates a practical understanding of ethical web application testing aligned with OWASP Top 10 principles.