EncryptEdge Labs

Cybersecurity Analyst Internship

Task Report

atalmamun@gmail.com

Task No: 19

Credit: Offensive Security

EncryptEdge Labs

Table of Contents

1.0 EncryptEdge Labs Internship Task Report

1.1 Introduction

Web application security is a crucial component of cybersecurity that focuses on protecting web applications from various threats and vulnerabilities. As web applications continue to play a vital role in personal, corporate, and governmental operations, ensuring their security is essential to maintain confidentiality, integrity, and availability of data. This task aims to introduce the foundational concepts of web application security, highlighting the importance of identifying and mitigating common vulnerabilities. By exploring key threats like Cross-Site Scripting (XSS), SQL Injection (SQLi), and Cross-Site Request Forgery (CSRF), this task provides insight into how attackers exploit weaknesses in web systems and the potential consequences of such breaches.

1.2 Objective

The primary objectives of this task are as follows:

- To understand the fundamentals of web application security and the dynamic challenges involved in securing web environments.
- To explore and gain knowledge about common web vulnerabilities such as XSS, SQL Injection, and CSRF.
- To analyze real-world case studies involving web security breaches and evaluate their impact on organizations and users.
- To reinforce theoretical understanding through hands-on experience via TryHackMe labs focused on web application vulnerabilities and OWASP Top 10.
- To reflect on personal learning experiences and challenges encountered during the task.

1.3 Requirements

To complete this task effectively, the following tools and resources are required:

- **Internet Access:** For conducting research on web application security concepts, vulnerabilities, and real-world breach case studies.
- **Kali Linux Virtual Machine (Optional):** Useful for running web security tools or performing practical labs.
- **TryHackMe Account:** To access and complete the relevant hands-on labs on web application security and OWASP Top 10 vulnerabilities.
- **Documentation Tools:** For writing and organizing the final report, such as a word processor or markdown editor.

2.0 Fundamentals of Web Application Security

Web application security involves the practices and technologies used to protect web-based applications from threats and vulnerabilities that could compromise the confidentiality, integrity, or availability of data. As web technologies evolve, so do the tactics of cybercriminals, making web application security a constantly shifting challenge. This section explores the foundational concepts of web application security, the common threats faced by web applications, and the importance of community-led initiatives like OWASP in defending the web ecosystem.

2.1 Understanding Web Application Security

Web applications are inherently complex, often involving multiple components such as front-end interfaces, back-end servers, databases, APIs, and third-party integrations. Each of these layers can introduce security risks if not properly configured or maintained.

Web application security focuses on identifying and mitigating these risks to prevent unauthorized access, data breaches, and service disruptions. Key aspects include secure coding practices, input validation, session management, authentication and authorization controls, and regular security testing.

2.2 The Dynamic Threat Landscape

The web threat landscape is continuously evolving due to:

- The widespread use of open-source software and third-party libraries, which may contain known vulnerabilities.
- Rapid deployment cycles in modern development (DevOps/Agile), sometimes leading to insufficient security testing.
- The increasing complexity of user interactions and personalization, which opens more attack vectors.
- Sophisticated attackers using automated tools and AI to exploit vulnerabilities at scale.

Because of these factors, web applications must be designed with security in mind from the ground up and continuously monitored throughout their lifecycle.

2.3 Role of OWASP in Web Security

The **Open Worldwide Application Security Project (OWASP)** is a non-profit organization dedicated to improving the security of software. OWASP plays a critical role in the web security community by:

- Publishing the **OWASP Top 10**, a regularly updated list of the most critical web application security risks.
- Providing open-source tools, documentation, and frameworks to help developers and security professionals.
- Raising awareness and promoting best practices for secure development.

The OWASP Top 10 has become an industry standard and is widely used by organizations to assess and improve their web security posture.

2.4 Key Challenges in Web Security

Securing web applications is challenging due to several persistent and emerging issues:

- **Input validation flaws** that lead to injection attacks like SQLi.
- **Cross-site scripting vulnerabilities** from improper handling of user-supplied content.
- **Authentication and session management issues**, allowing attackers to hijack user accounts.
- **Insecure APIs and third-party integrations** that may be exploited as backdoors.
- **Lack of developer awareness or security training**, resulting in insecure coding practices.

Overcoming these challenges requires a combination of secure coding, proactive threat modeling, regular testing, and education.

3.0 Common Web Vulnerabilities

Web applications are frequent targets for cyberattacks due to the rich data they handle and their exposure to the internet. Among the various threats, some vulnerabilities consistently rank among the most exploited by attackers. This section outlines three of the most common web vulnerabilities—**Cross-Site Scripting (XSS)**, **SQL Injection (SQLi)**, and **Cross-Site Request Forgery (CSRF)**—explaining how they work, their impact, and notable real-world cases.

3.1 Cross-Site Scripting (XSS)

Description:
Cross-Site Scripting (XSS) is a client-side vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts typically execute in the victim's browser and can steal session cookies, redirect users, or perform actions on behalf of the user without their knowledge.

How It Works:
An attacker identifies a vulnerable input field on a web page that fails to properly sanitize or encode user input. By injecting malicious JavaScript code into the field (e.g., a comment box), the script is stored and executed when another user loads the page.

Impact:

- Theft of sensitive data such as session tokens or login credentials.
- Defacement of websites or redirection to malicious domains.
- Full control over a user's interaction with a vulnerable site.

Real-World Example:
In 2014, eBay suffered from an XSS vulnerability that allowed attackers to inject malicious JavaScript into product listings. Users who clicked on the affected listings were redirected to phishing pages, risking credential theft.

3.2 SQL Injection (SQLi)

Description:
SQL Injection is a code injection attack that allows attackers to interfere with the queries an application makes to its database. This occurs when input fields are not properly sanitized, allowing attackers to manipulate SQL queries and gain unauthorized access to data.

How It Works:
An attacker submits specially crafted SQL statements into input fields (e.g., login forms or search bars). If the input is not properly validated or sanitized, the SQL query may be altered to execute unintended commands—such as retrieving all records from a database or deleting data.

Impact:

- Unauthorized access to sensitive data (e.g., usernames, passwords, credit card info).
- Modification or deletion of data.
- Potential full database compromise or administrative access.

Real-World Example:
The 2008 Heartland Payment Systems breach was partially attributed to SQL injection. Over 130 million credit and debit card numbers were stolen, making it one of the largest data breaches in history.

3.3 Cross-Site Request Forgery (CSRF)

Description:
Cross-Site Request Forgery (CSRF) is an attack that tricks a user into performing actions they did not intend to, typically on a web application where they are authenticated. It leverages the trust that a web application has in the user's browser.

How It Works:
While the victim is logged into a web application, an attacker crafts a malicious request

(e.g., to change account settings) and tricks the user into unknowingly submitting it—often via a hidden form on a malicious website. Since the user is authenticated, the request is accepted by the server.

Impact:

- Unauthorized transactions or settings changes.
- Data leaks or account takeovers.
- Exploitation of trusted user privileges.

Real-World Example:
In 2010, Twitter was vulnerable to a CSRF attack that allowed users to force others to follow them without consent. Attackers embedded malicious links in tweets that, when clicked, executed unauthorized follow actions.

4.0 Real-World Case Studies

Real-world security breaches serve as powerful lessons for understanding the consequences of web vulnerabilities and highlight the importance of proactive security measures. This section examines two high-profile web application security incidents, detailing the vulnerabilities that were exploited, the scale of the impact, and how these attacks could have been mitigated with better practices.

4.1 Case Study 1: The 2017 Equifax Data Breach

Overview:
In 2017, Equifax, one of the largest credit reporting agencies in the United States, suffered a massive data breach that exposed the personal information of approximately 147 million individuals. The attack was the result of a failure to patch a known vulnerability in a web application.

Exploited Vulnerability:
The attackers exploited a known vulnerability in **Apache Struts (CVE-2017-5638)**, a widely used open-source framework for building Java web applications. The

vulnerability allowed for **Remote Code Execution (RCE)** via improper handling of user-supplied input in HTTP headers.

Impact:

- Exposure of sensitive data, including Social Security numbers, birth dates, addresses, and in some cases, driver's license numbers.
- Loss of public trust and severe reputational damage.
- Estimated financial cost: over $700 million in settlements and fines.

Prevention:

- Timely application of critical security patches.
- Regular vulnerability scanning and threat intelligence monitoring.
- Use of a Web Application Firewall (WAF) to filter malicious requests.

4.2 Case Study 2: The 2012 LinkedIn SQL Injection Breach

Overview:
In 2012, LinkedIn experienced a data breach in which approximately 6.5 million user passwords were stolen and leaked on a Russian hacker forum. The breach stemmed from a SQL injection vulnerability in the website's login form.

Exploited Vulnerability:
SQL Injection (SQLi) was used to extract data from LinkedIn's database. The vulnerability allowed attackers to manipulate SQL queries and gain access to hashed user passwords.

Impact:

- Massive leakage of user credentials.
- Reputational damage and a loss of user trust.
- Triggered a shift in industry-wide practices related to password hashing.

Prevention:

- Proper input validation and use of prepared statements or parameterized queries.
- Stronger password hashing mechanisms (LinkedIn was using SHA-1, which is now considered insecure).
- Regular security code audits and penetration testing.

4.3 Lessons Learned

These case studies emphasize several key lessons:

- **Patching promptly** is crucial: Delays in applying security updates can be catastrophic.
- **Input validation and sanitization** are non-negotiable: SQLi and similar attacks thrive on poor input handling.
- **Security is a continuous process**: Organizations must invest in regular security assessments, employee training, and automated tools to maintain a secure environment.

5.0 Hands-on Labs

Hands-on practice is essential for solidifying theoretical knowledge in cybersecurity. As part of this task, two TryHackMe labs were completed to explore common web vulnerabilities and understand their real-world impact. These labs provided a practical environment to identify, exploit, and mitigate web application security risks aligned with the OWASP Top 10 vulnerabilities.

5.1 TryHackMe Lab 1: Web Application Security

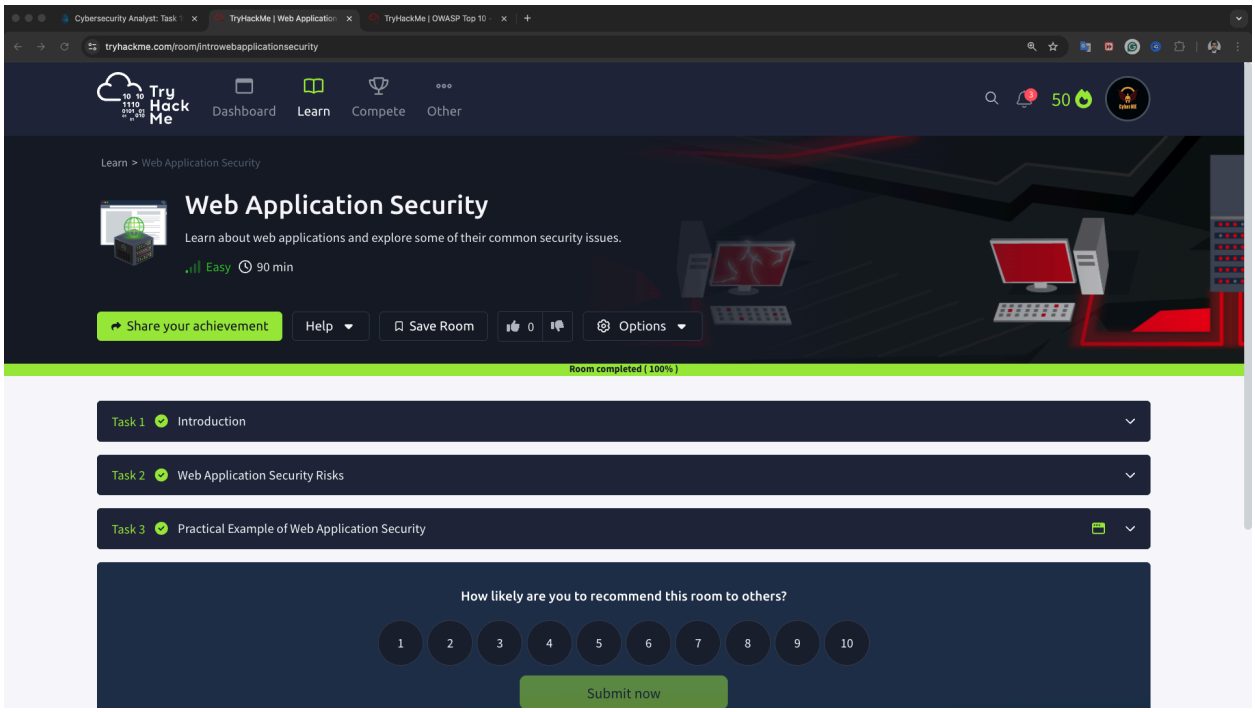
Overview:
This introductory lab focused on teaching the foundational concepts of web application vulnerabilities. It covered a wide range of topics, including client-server architecture,

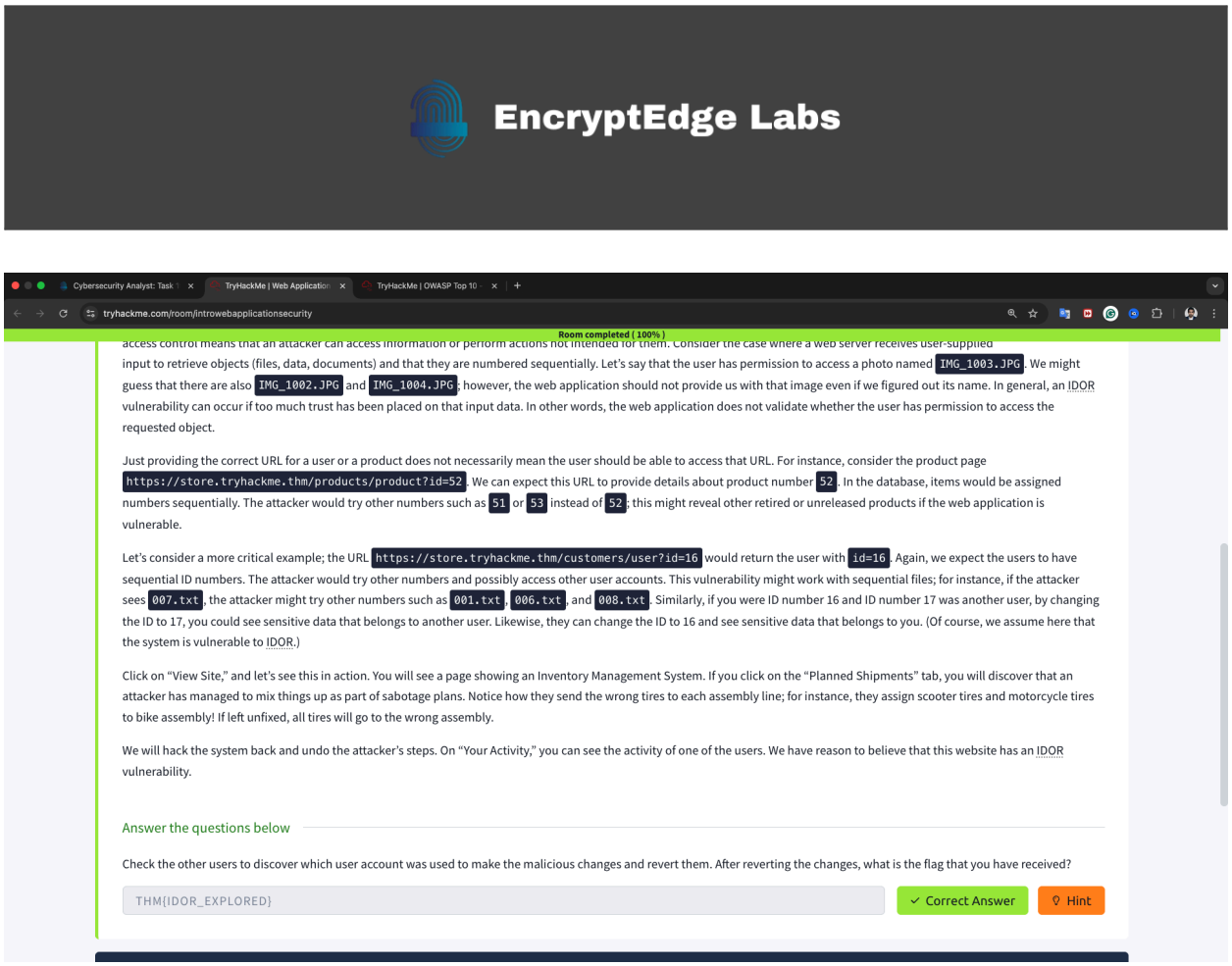
HTTP requests and responses, and basic web security issues like insecure forms and hidden fields.

Key Topics Covered:

- Structure of web applications and how they interact with users.
- Identifying insecure input fields.
- Understanding how attackers use simple inspection tools (like browser developer tools) to uncover flaws.
- Hands-on demonstrations of weak login systems and basic vulnerability exploitation.

Screenshots:





5.2 TryHackMe Lab 2: OWASP Top 10

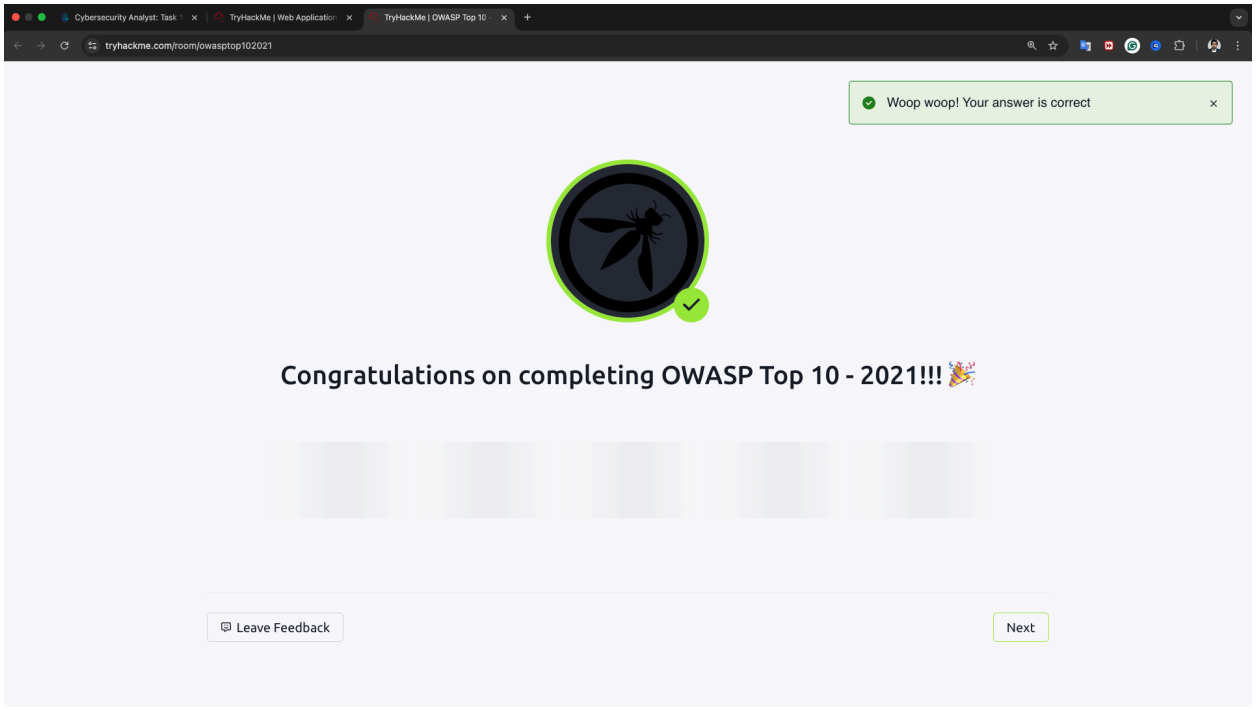
Overview:

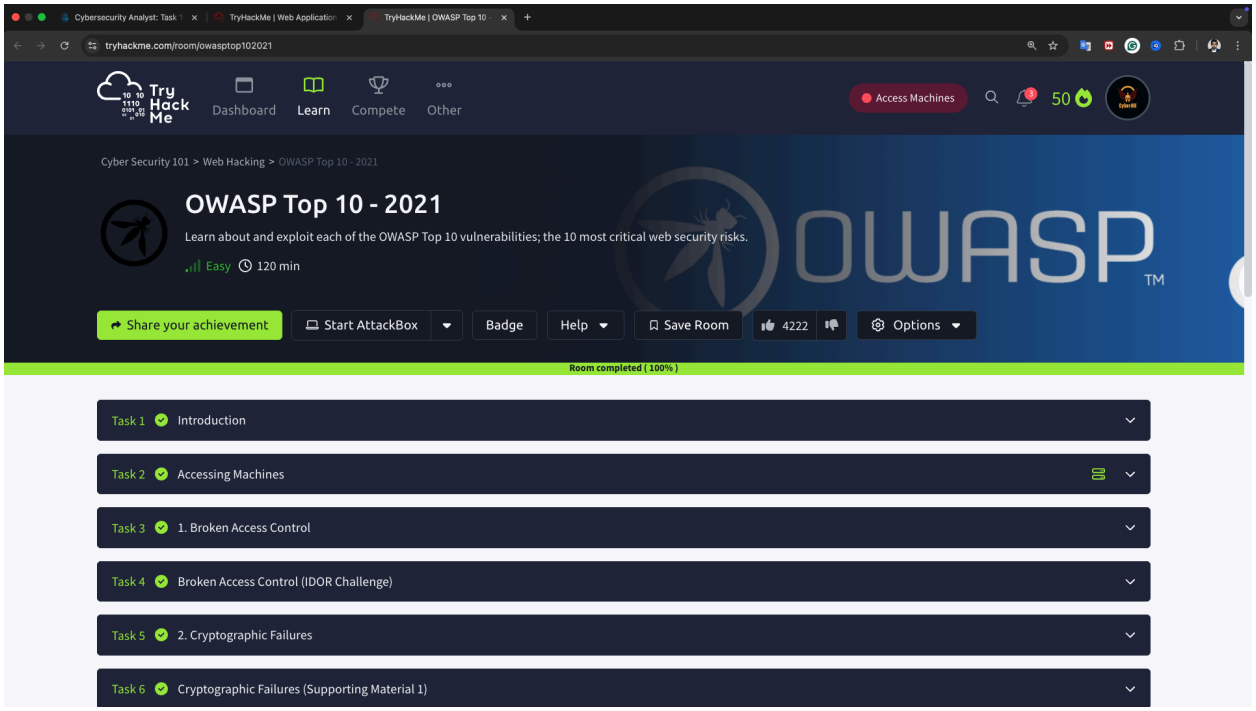
This lab provided in-depth coverage of the most critical web application security risks, as listed by OWASP. The interactive environment allowed for hands-on exploitation of vulnerabilities such as XSS, SQL Injection, and CSRF, and also included secure coding practices and defensive strategies.

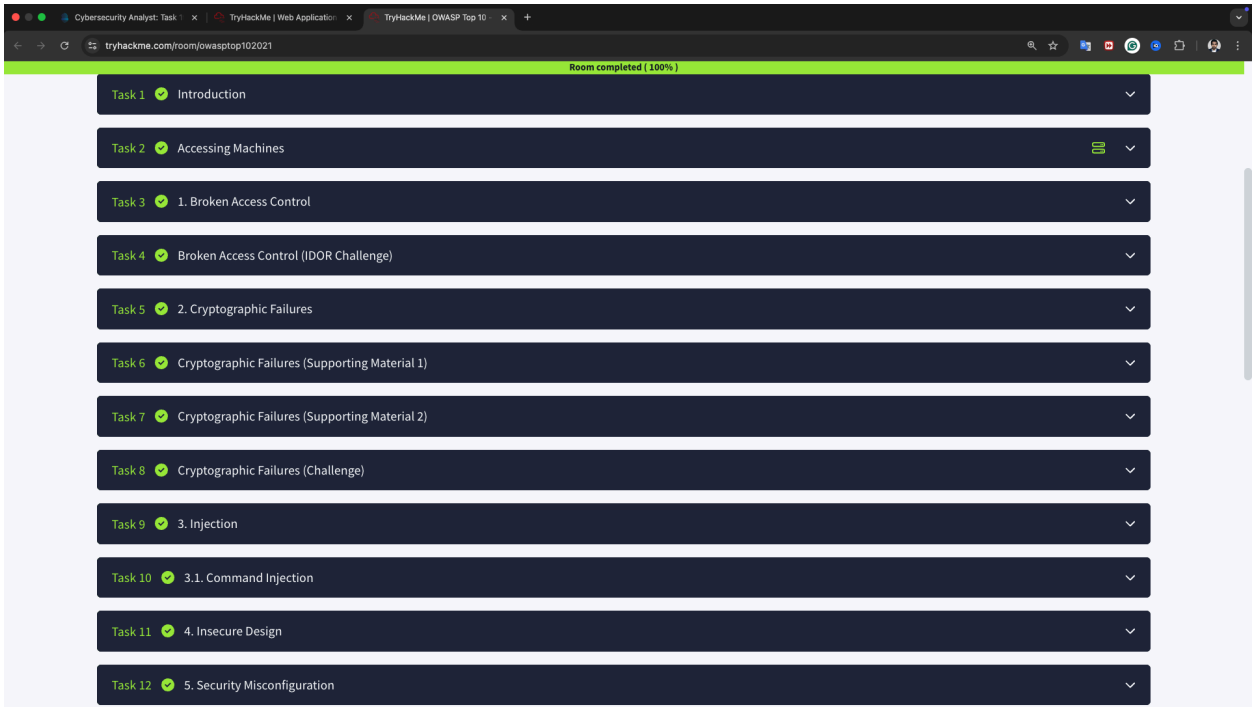
Key Topics Covered:

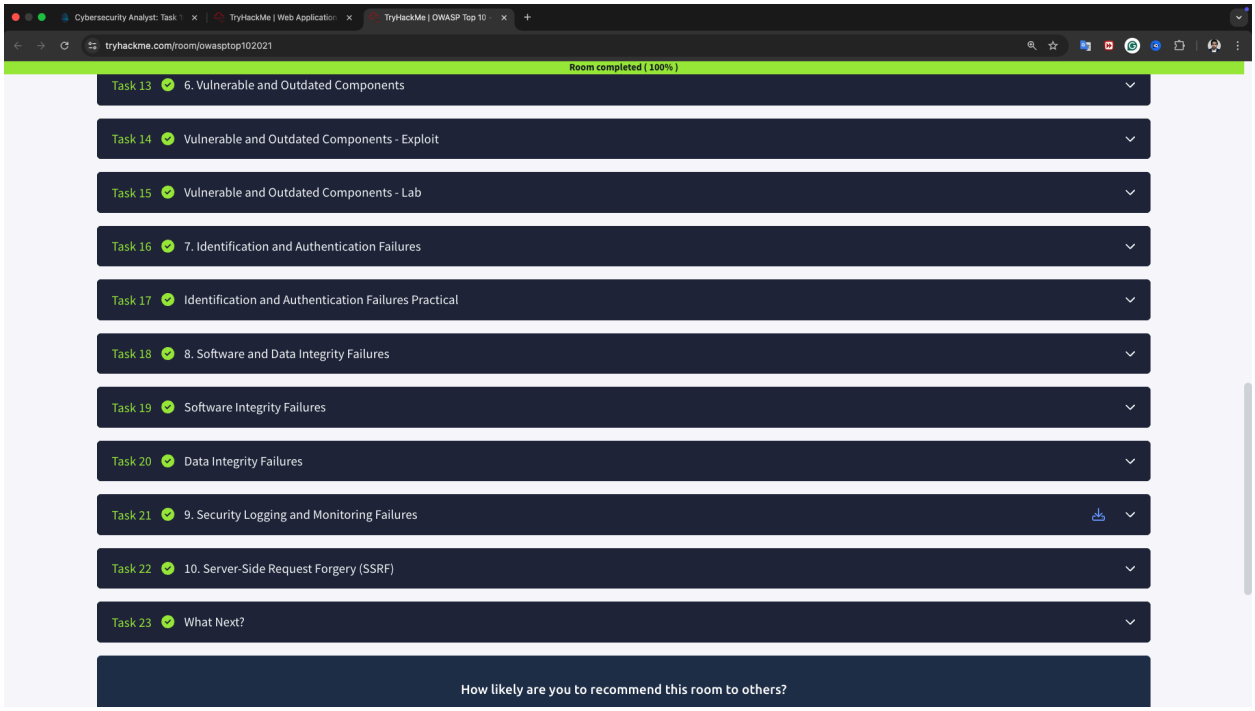
- Cross-Site Scripting (XSS) and how to prevent it using output encoding.
- SQL Injection and use of parameterized queries to prevent database manipulation.
- CSRF and implementing CSRF tokens for secure requests.
- Insecure Deserialization, Security Misconfigurations, and Broken Authentication.

Screenshots:






tryhackme.com/room/owasptop102021
Room completed (100%)
Task 1 Introduction
Task 2 Accessing Machines
Task 3 1. Broken Access Control
Task 4 Broken Access Control (IDOR Challenge)
Task 5 2. Cryptographic Failures
Task 6 Cryptographic Failures (Supporting Material 1)
Task 7 Cryptographic Failures (Supporting Material 2)
Task 8 Cryptographic Failures (Challenge)
Task 9 3. Injection
Task 10 3.1. Command Injection
Task 11 4. Insecure Design
Task 12 5. Security Misconfiguration


tryhackme.com/room/owasptop102021
Room completed (100%)
Task 13 6. Vulnerable and Outdated Components
Task 14 Vulnerable and Outdated Components - Exploit
Task 15 Vulnerable and Outdated Components - Lab
Task 16 7. Identification and Authentication Failures
Task 17 Identification and Authentication Failures Practical
Task 18 8. Software and Data Integrity Failures
Task 19 Software Integrity Failures
Task 20 Data Integrity Failures
Task 21 9. Security Logging and Monitoring Failures
Task 22 10. Server-Side Request Forgery (SSRF)
Task 23 What Next?
How likely are you to recommend this room to others?

6.0 Reflection

Completing this task on the **Basics of Web Application Security** has been an insightful and rewarding experience. Through a combination of research, case studies, and hands-on labs, I have gained a deeper understanding of how web vulnerabilities are exploited and the serious consequences they can have on organizations and users.

One of the main challenges I encountered was fully grasping how some attacks, such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF), exploit the trust between users and web applications. Initially, the technical workings behind these attacks were complex, but the TryHackMe labs provided an interactive and practical approach that helped demystify the exploitation process and clarify the importance of proper input handling and session security.

The case studies, especially the Equifax and LinkedIn breaches, emphasized how even well-established organizations can fall victim to preventable security flaws. These incidents highlighted the critical need for regular patch management, secure coding practices, and a proactive approach to vulnerability management.

Overall, this task has strengthened my foundational knowledge of web security and emphasized the importance of building secure web applications from the ground up. It has also reinforced the value of continuous learning and staying updated with evolving threats, particularly through resources like OWASP and hands-on platforms such as TryHackMe.

This Internship Task report was developed on [April, 21, 2025]

By:

atalmamun@gmail.com