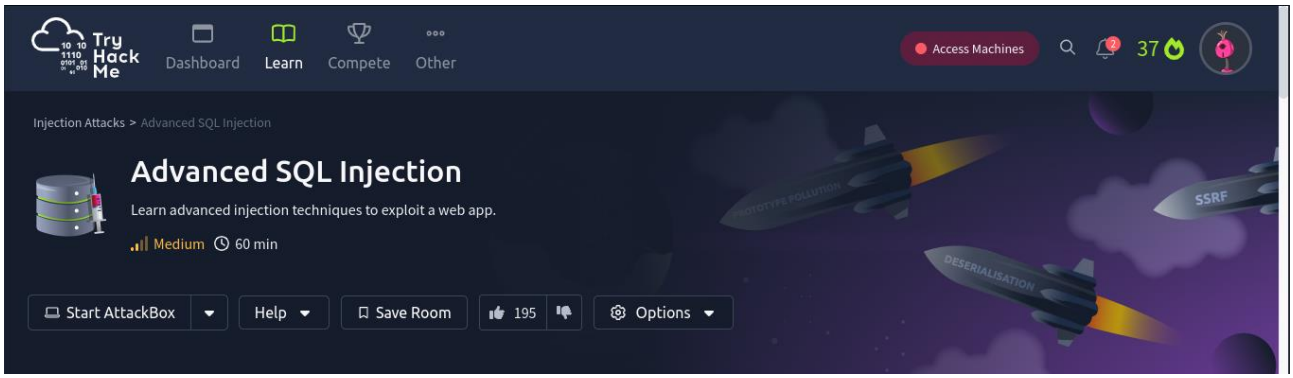
CORVIT SYSTEM MULTAN

Muhammad Fatiq
Submitted To: Muhammad Bilal

Tryhackme Room

Walkthrough

Advance SQL Injection

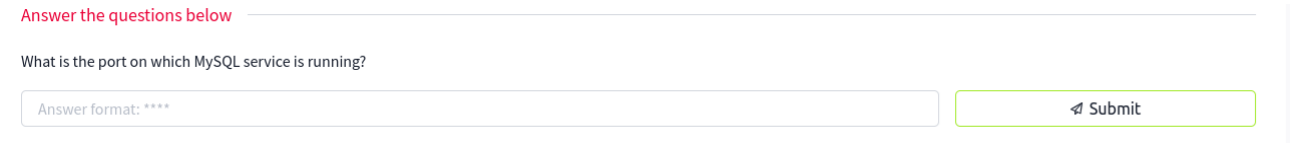


Task 1:Introduction

SQL injection remains one of web applications' most severe and widespread security vulnerabilities. This threat arises when an attacker exploits a web application's ability to execute

arbitrary SQL queries, leading to unauthorised access to the database, data exfiltration, data manipulation, or even complete control over the application. In this room, we will understand

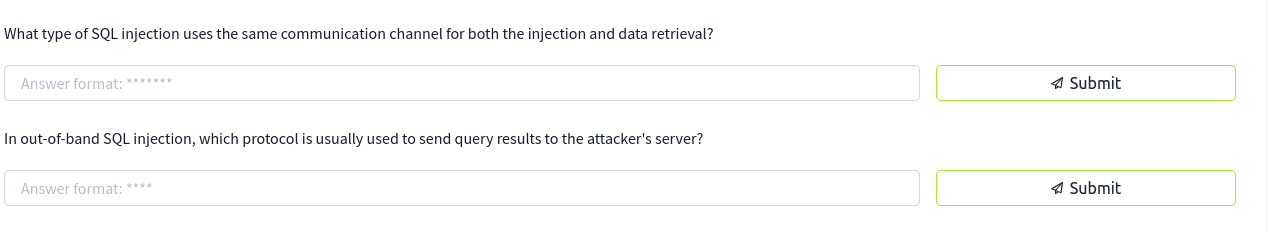
advanced SQL injection techniques, providing a comprehensive understanding of sophisticated attack vectors and mitigation strategies.



ANSWER: 3306

TASK 2: Quick Recap

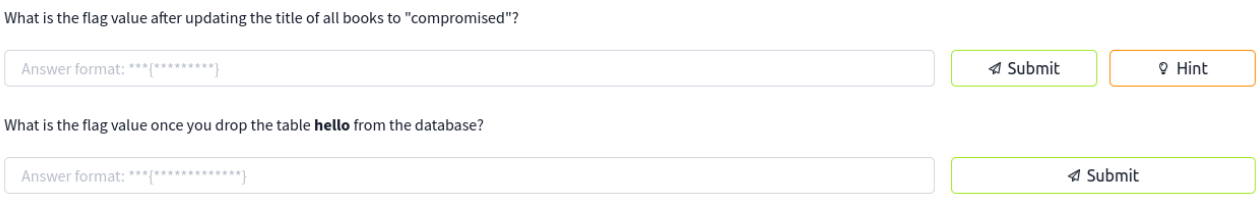
In the last SQL injection room, we explored the basics of SQL injection, understanding how attackers exploit vulnerabilities in web applications to manipulate SQL queries and access unauthorised data. We covered essential techniques, such as error-based and union-based SQL injection, and blind SQL injection methods, such as boolean-based and time-based attacks. Here is a quick recap of the room covering the core essential types of SQL injection.



ANSWER : In-band
ANSWER : HTTP

TASK 3: Second order SQL Injection

Second-order SQL injection, also known as stored SQL injection, exploits vulnerabilities where user-supplied input is saved and subsequently used in a different part of the application, possibly after some initial processing. This type of attack is more insidious because the malicious SQL code does not need to immediately result in a SQL syntax error or other obvious issues, making it harder to detect with standard input validation techniques. The injection occurs upon the second use of the data when it is retrieved and used in a SQL command, hence the name "Second Order".



ANSWER: THM{SO_HACKED}
ANSWER: THM{Table_Dropped}

TASK 4: Filter Evasion Techniques

In advanced SQL injection attacks, evading filters is crucial for successfully exploiting vulnerabilities. Modern web applications often implement defensive measures to sanitise or block common attack patterns, making simple SQL injection attempts ineffective. As pentesters, we must adapt using more sophisticated techniques to bypass these filters. This section will cover such methods, including **character encoding**, **no-quote** SQL injection, and handling scenarios where **spaces** cannot be used. We can effectively penetrate web applications with stringent input validation and security controls by understanding and applying these techniques.

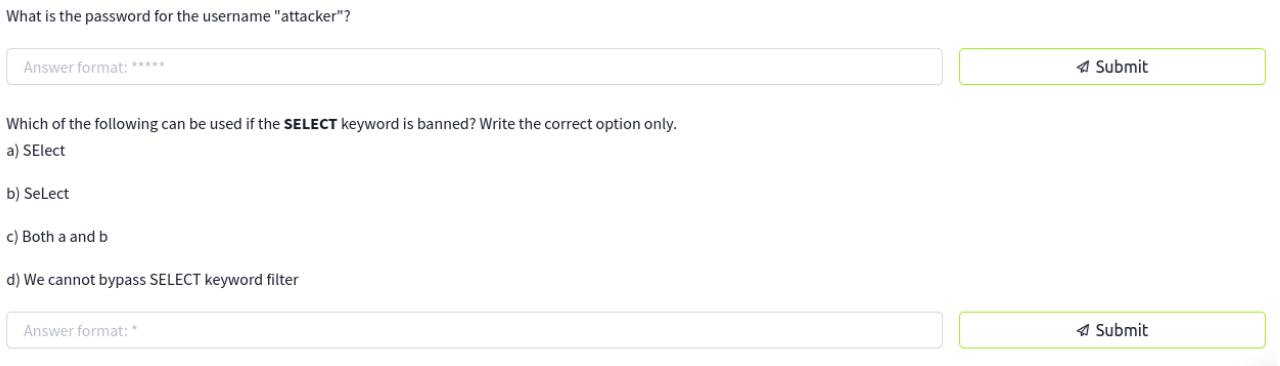
ANSWER: 1064
ANSWER: Animal Series

TASK 5: Filter Evasions Techniques

No-Quote SQL Injection

No-Quote SQL injection techniques are used when the application filters single or double quotes or escapes.

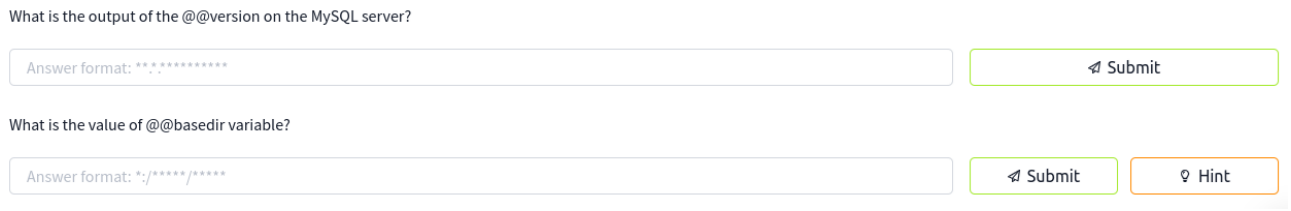
- **Using Numerical Values:** One approach is to use numerical values or other data types that do not require quotes. For example, instead of injecting **' OR '1'='1**, an attacker can use **OR 1=1** in a context where quotes are not necessary. This technique can bypass filters that specifically look for an escape or strip out quotes, allowing the injection to proceed.
- **Using SQL Comments**: Another method involves using SQL comments to terminate the rest of the query. For instance, the input **admin'--** can be transformed into **admin--**, where the **--** signifies the start of a comment in SQL, effectively ignoring the remainder of the SQL statement. This can help bypass filters and prevent syntax errors.
- **Using CONCAT() Function**: Attackers can use SQL functions like **CONCAT()** to construct strings without quotes. For example, **CONCAT(0x61, 0x64, 0x6d, 0x69, 0x6e)** constructs the string admin. The **CONCAT()** function and similar methods allow attackers to build strings without directly using quotes, making it harder for filters to detect and block the payload.



ANSWER: Tesla
ANSWER: C

TASK 6: Out of band SQL Injection

Out-of-band (OOB) SQL injection is an attack technique that pentester/red teamers use to exfiltrate data or execute malicious actions when direct or traditional methods are ineffective. Unlike In-band SQL injection, where the attacker relies on the same channel for attack and data retrieval, Out-of-band SQL injection utilises separate channels for sending the payload and receiving the response. Out-of-band techniques leverage features like HTTP requests, DNS queries, SMB protocol, or other network protocols that the database server might have access to, enabling attackers to circumvent firewalls, intrusion detection systems, and other security measures.

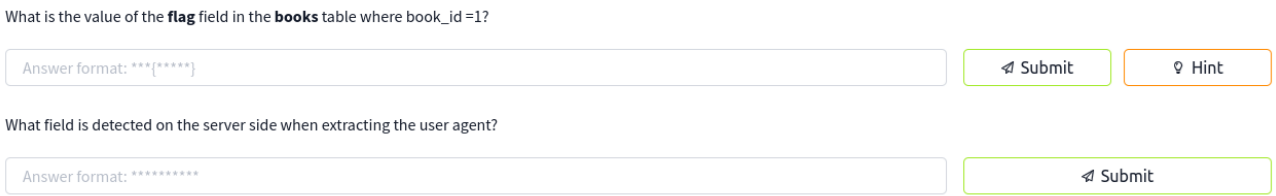


ANSWER: 10.4.24-MariaDB

ANSWER: C:/xampp/mysql

TASK 7: Other Techniques

HTTP Header Injection

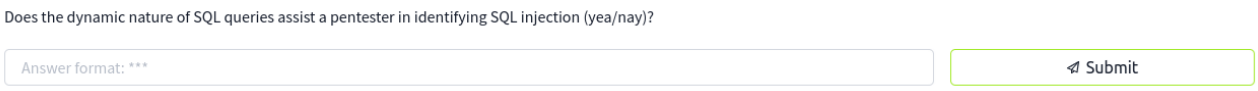


ANSWER: THM{HELLO}

ANSWER: User-Agent

TASK 8: Automation

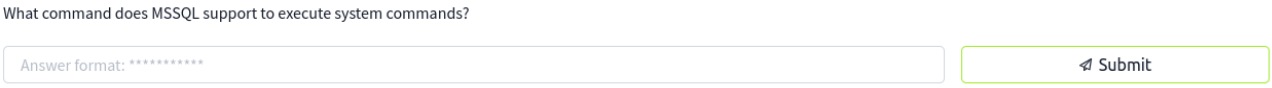
SQL Injection remains a common threat due to improper implementation of security measures and the complexity of different web frameworks. Automating identification and exploiting these vulnerabilities can be challenging, but several tools and techniques have been developed to help streamline this process.



ANSWER: nay

TASK 9: Best Practices

SQL injection is a renowned and pervasive vulnerability that has been a major concern in web application security for years. Pentesters must pay special attention to this vulnerability during their assessments, as it requires a thorough understanding of various techniques to identify and exploit SQL injection points. Similarly, secure coders must prioritise safeguarding their applications by implementing robust input validation and adhering to secure coding practices to prevent such attacks. A few of the best practices are mentioned below:



Answer: xp_cmdshell

TASk 10: Conclution

In this room, we have explored several advanced SQL injection techniques, including Second-Order SQL Injection, Out-of-Band SQLi, and filter evasion. We also covered techniques like cookie injection, illustrating the diverse methods attackers use to exploit web applications. Our journey didn't stop at exploitation; we discussed the importance of automation in identifying and exploiting SQL injection vulnerabilities and leveraging tools to streamline and enhance our testing processes.

Understanding these advanced techniques is crucial for any penetration tester aiming to uncover and address complex security flaws. Additionally, we discussed various mitigation measures to safeguard applications against these sophisticated attacks, emphasising the need for a robust security posture.

As a penetration tester, your role is not only to find vulnerabilities but also to understand the best practices for remediation and prevention. Armed with the knowledge from this room, you are better

equipped to protect web applications, ensuring they are resilient against the evolving landscape of SQL injection threats.