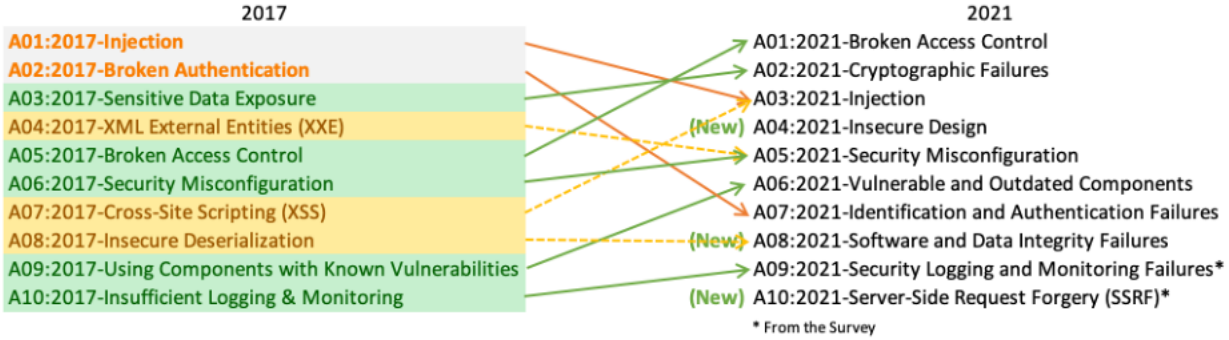
Task 1: Introduction

- The TryHackMe OWASP Top 10 module provides a practical tour through the most common web vulnerabilities. It's a hands-on learning experience, covering everything from basic setup to advanced security threats. This course is essential for anyone keen on mastering web application security.
- Overview of the OWASP Top 10, a standard awareness document for developers and web application security.
- Importance of understanding common web vulnerabilities and threats. https://tryhackme.com/p/Damiano254



Task 2: Accessing Machines Through OpenVPN

- Setting up and using OpenVPN to access virtual machines for practical exercises.
- Significance of secure connections while accessing remote systems.

Task 3: [Severity 1] Injection

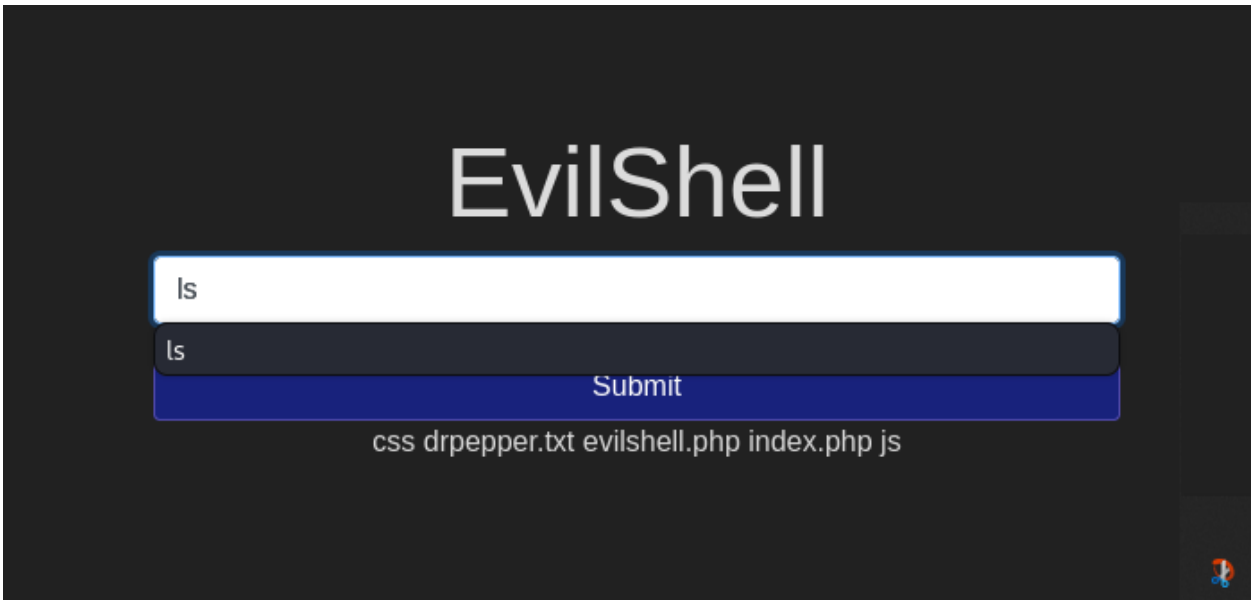
- Understanding injection flaws, especially SQL, NoSQL, OS, and LDAP injection.
- Attackers can use these flaws to execute unauthorized commands or access data.
- Example: SQL injection where malicious SQL statements are inserted into an entry field.

Task 4: [Severity 1] OS Command Injection

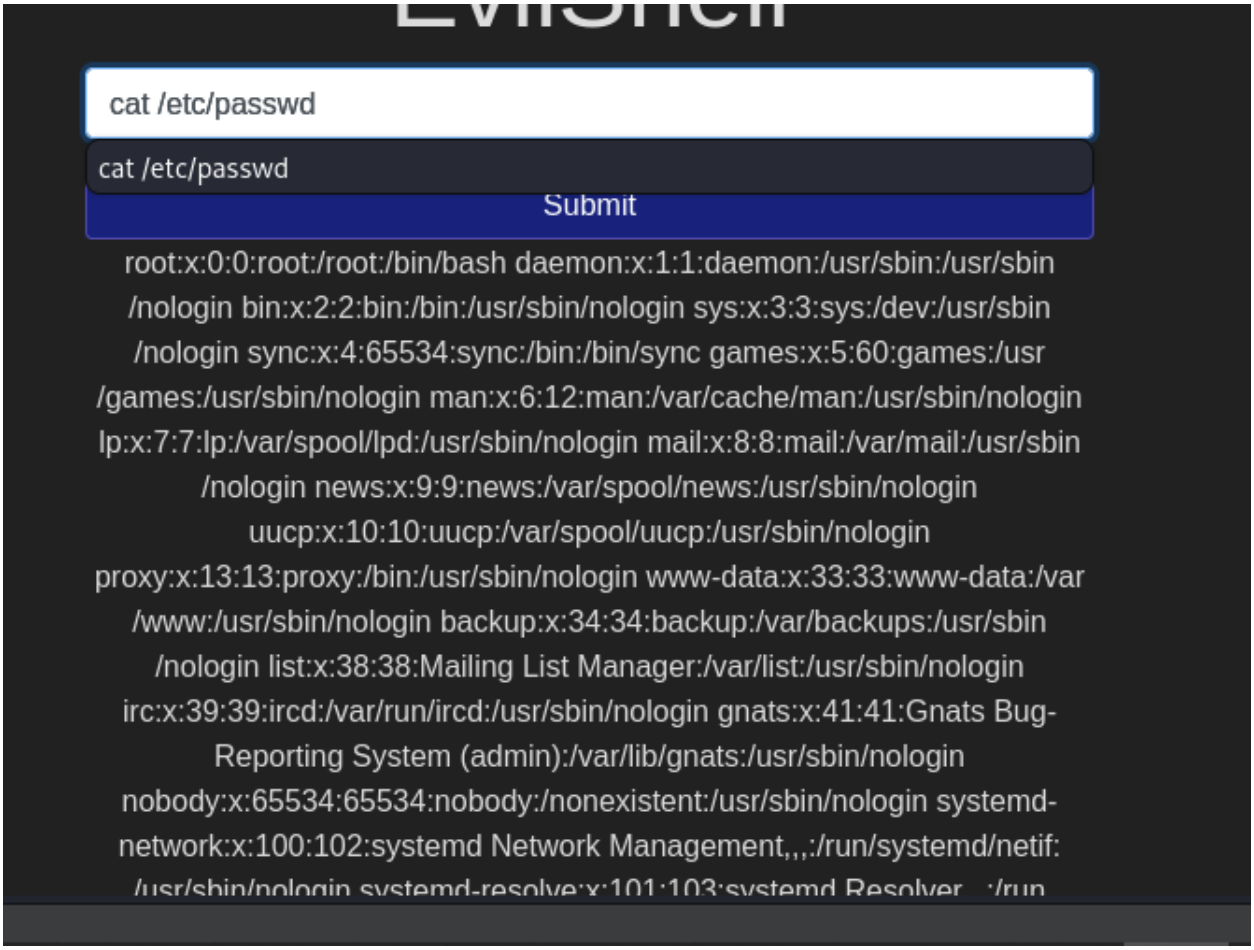
- Understanding how attackers execute arbitrary commands on the host operating system.
- This vulnerability arises due to insufficient input validation.
- Example: Web application calls a system command with user input, which is exploitable.

Task 5: [Severity 1] Command Injection Practical

- Practical exploration of command injection.
- Identifying and exploiting command injection vulnerabilities.
- Example Answers:
    - **Strange file:** Adrpepper.txt



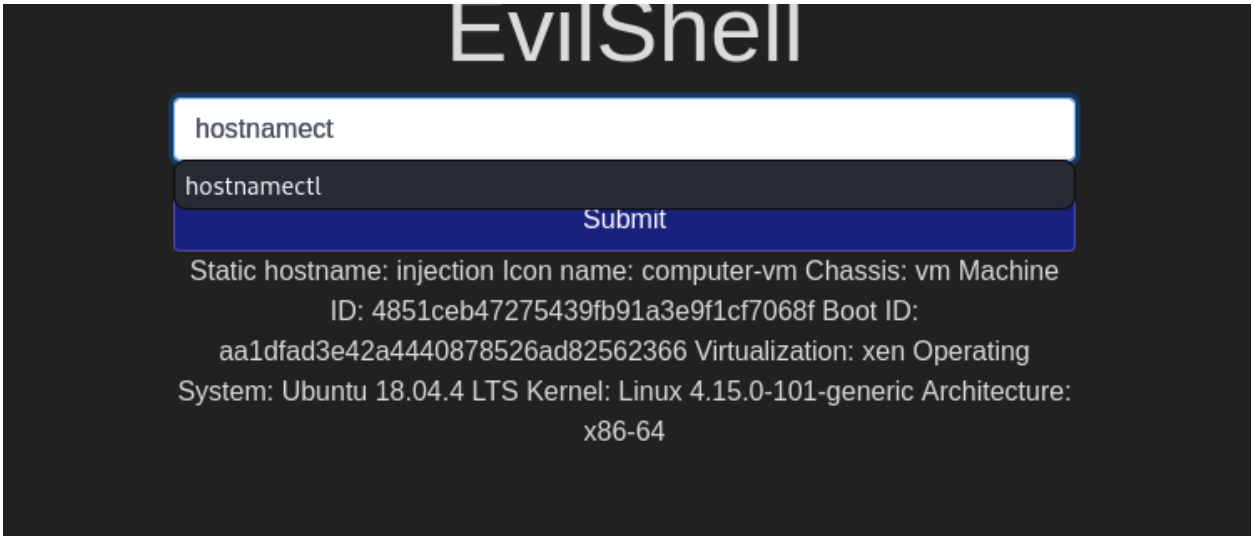
- **Non-root users:** 0



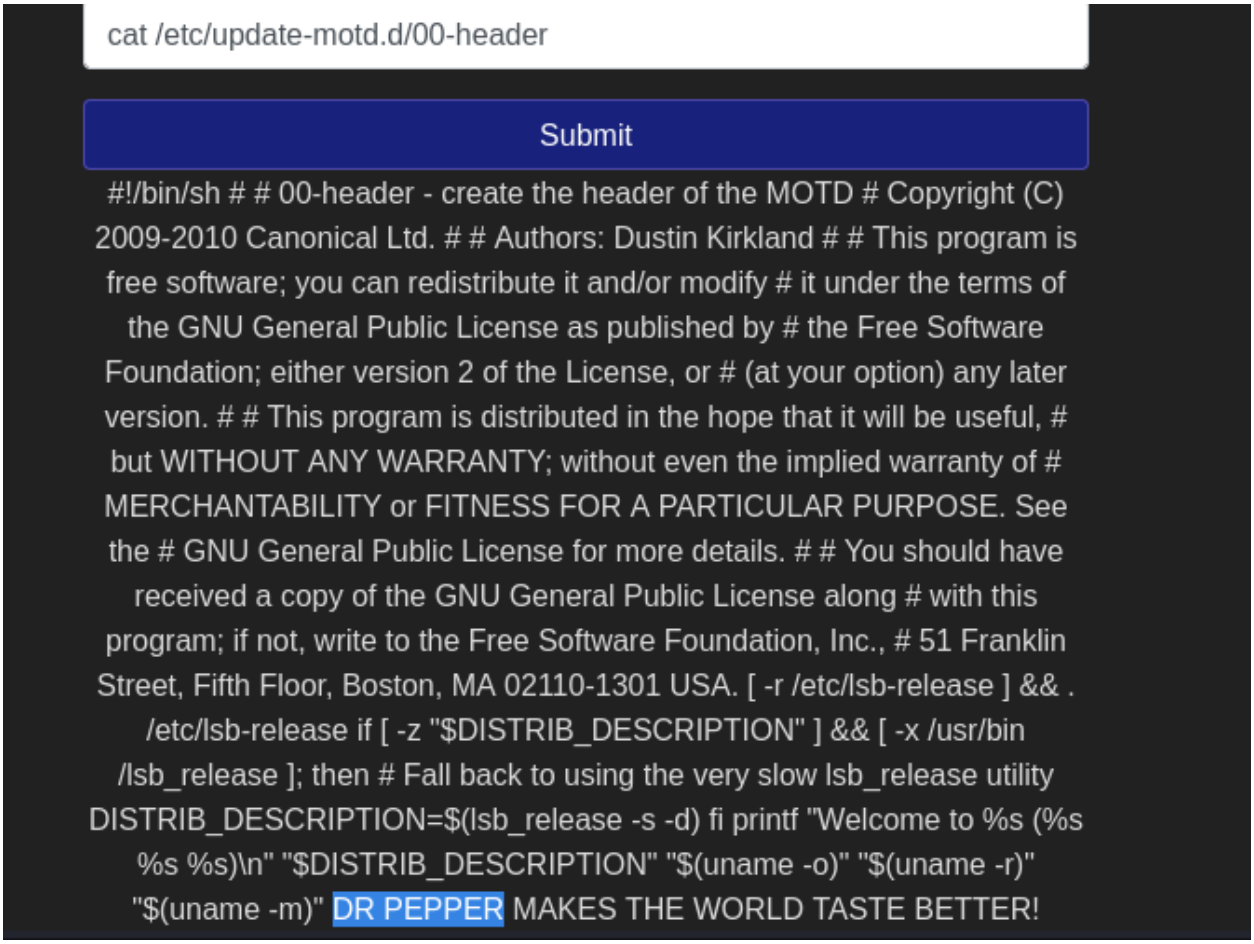
- **App running user:** www-data

- **User's shell:** /usr/sbin/nologin

- **Ubuntu version:** 18.04.4



- **MOTD beverage:** Dr Pepper



Task 6: [Severity 2] Broken Authentication

- Exploring issues related to authentication mechanisms.
- Vulnerabilities can allow attackers to compromise passwords, keys, or session tokens.

Task 7: [Severity 2] Broken Authentication Practical

- Practical example: exploiting broken authentication.
- Finding flags in user accounts by bypassing authentication.
- **Example Answers:**
    - **Darren's flag:** fe86079416a21a3c99937fea8874b667

Username:

darren

Email:

try@fakeemail.com

Password:

●●●●

Register

Error: This user is already registered

Username:

darren

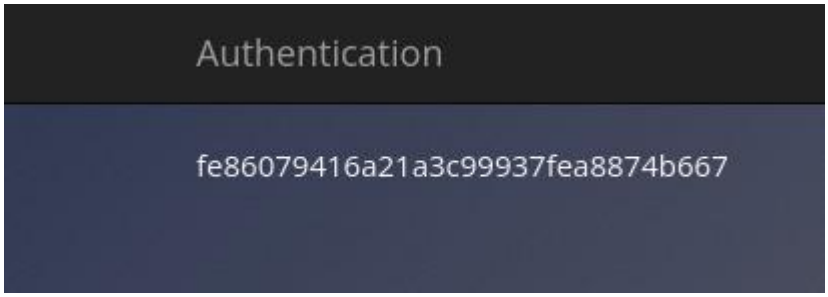
Email:

email@fake.com

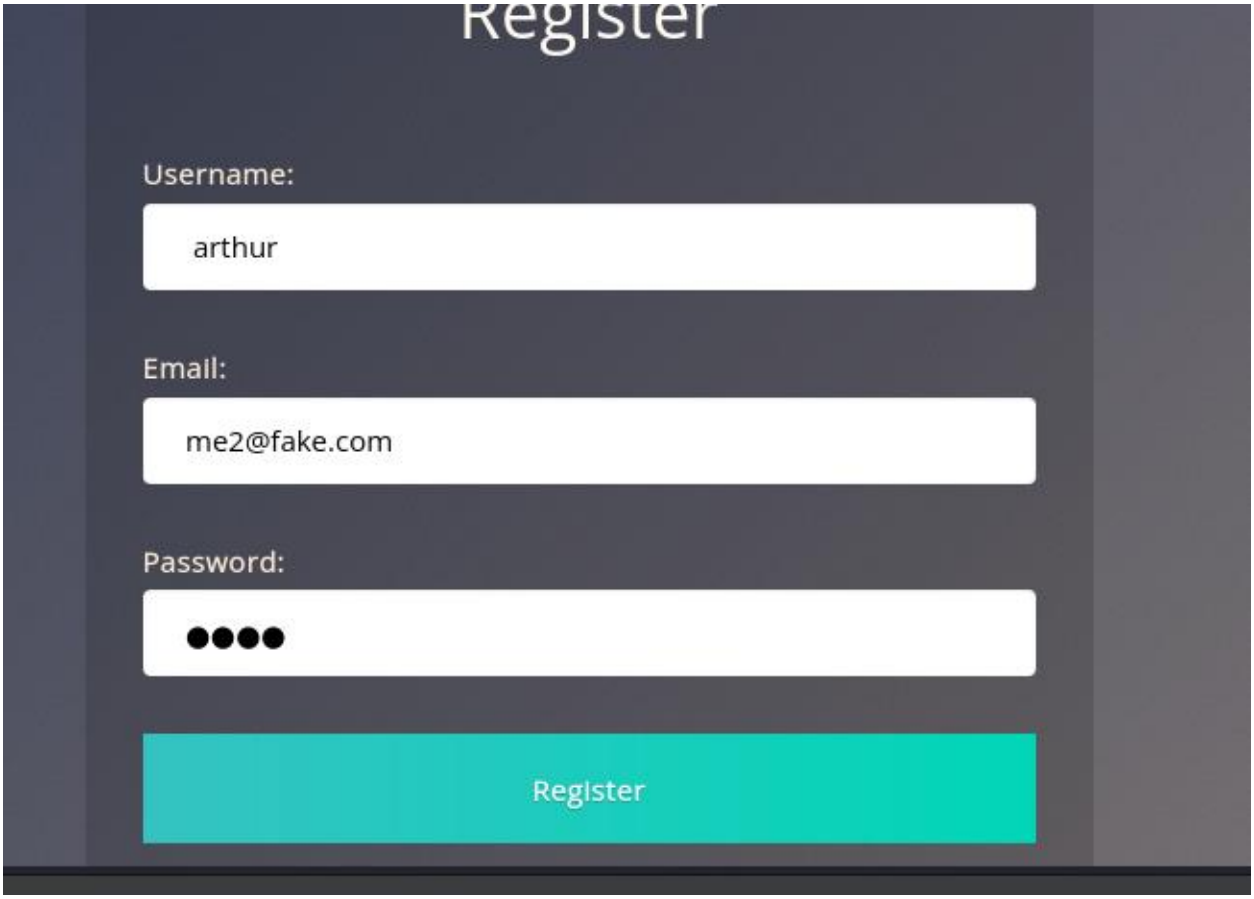
Password:

●●●●

Register

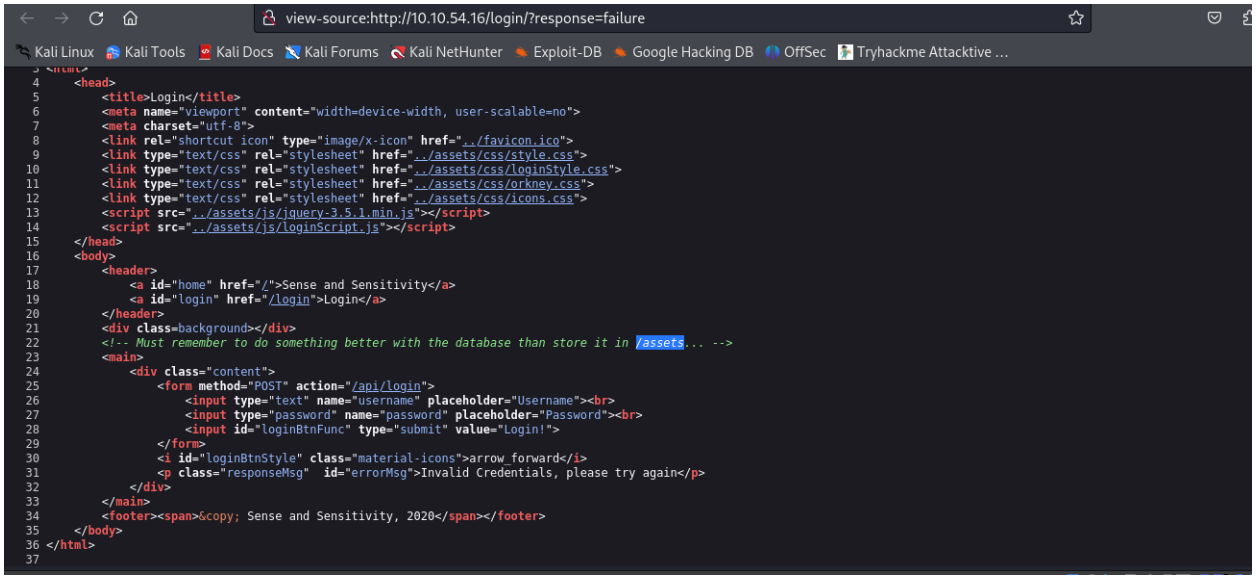


- **Arthur's flag:** d9ac0f7db4fda460ac3edeb75d75e16e



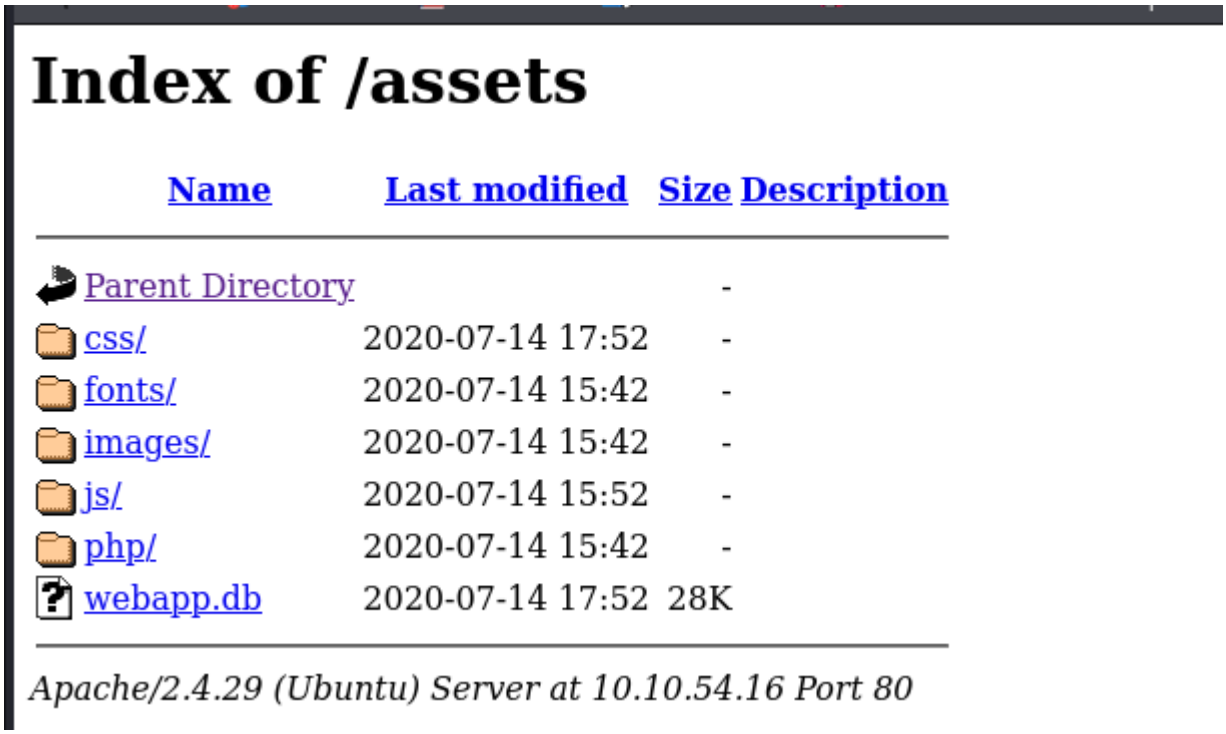
Task 8-11: [Severity 3] Sensitive Data Exposure

- Understanding the risks of exposing sensitive data.
- Inadequate protection of sensitive data like financial, healthcare, or PII.
- Practical challenge: identifying and accessing sensitive data.
- **Example Answers:**
  - **Sensitive directory:** /assets




```
<html>
    <head>
        <title>Login</title>
        <meta name="viewport" content="width=device-width, user-scalable=no">
        <meta charset="utf-8">
        <link rel="shortcut icon" type="image/x-icon" href="../favicon.ico">
        <link type="text/css" rel="stylesheet" href="../assets/css/style.css">
        <link type="text/css" rel="stylesheet" href="../assets/css/loginStyle.css">
        <link type="text/css" rel="stylesheet" href="../assets/css/orkney.css">
        <link type="text/css" rel="stylesheet" href="../assets/css/icons.css">
        <script src="../assets/js/jquery-3.5.1.min.js"></script>
        <script src="../assets/js/loginScript.js"></script>
    </head>
    <body>
        <header>
            <a id="home" href="/">Sense and Sensitivity</a>
            <a id="login" href="/login">Login</a>
        </header>
        <div class=background></div>
        <!-- Must remember to do something better with the database than store it in /assets... -->
        <main>
            <div class="content">
                <form method="POST" action="/api/login">
                    <input type="text" name="username" placeholder="Username"><br>
                    <input type="password" name="password" placeholder="Password"><br>
                    <input id="loginBtnFunc" type="submit" value="Login!">
                </form>
                <i id="loginBtnStyle" class="material-icons">arrow_forward</i>
                <p class="responseMsg"  id="errorMsg">Invalid Credentials, please try again</p>
            </div>
        </main>
        <footer><span>&copy; Sense and Sensitivity, 2020</span></footer>
    </body>
</html>
```

- **Notable file:** webapp.db

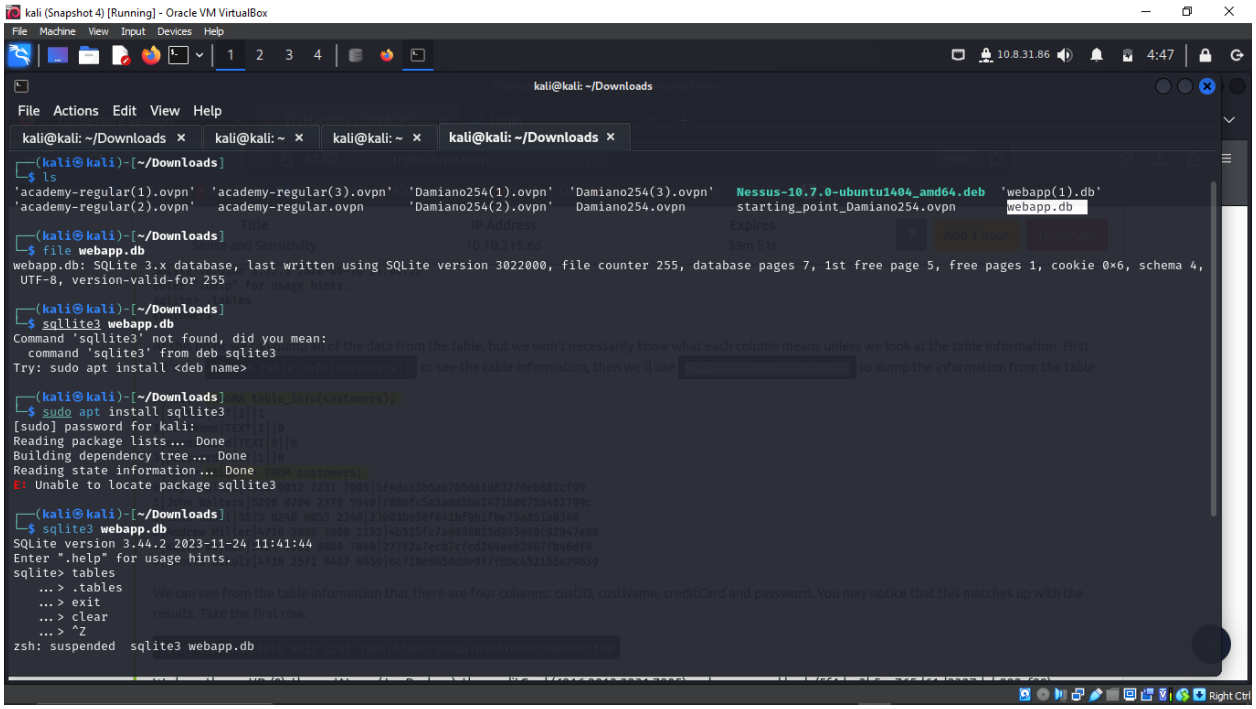


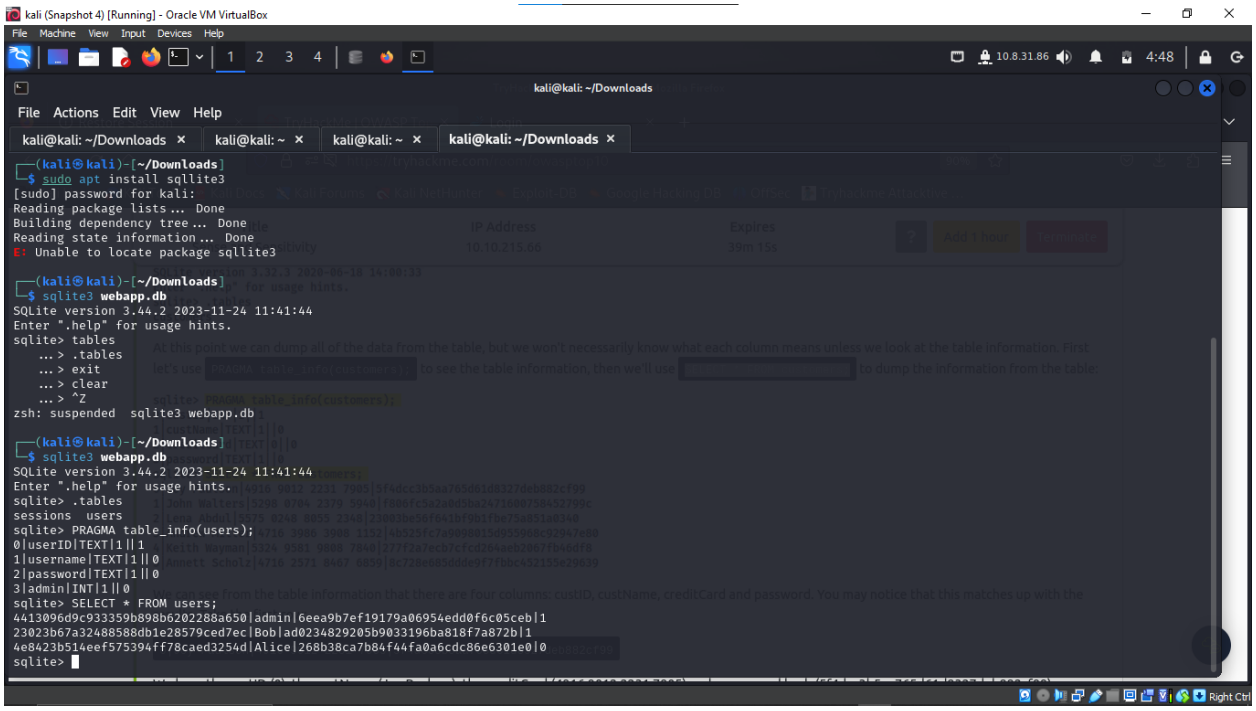
Index of /assets

Name	Last modified	Size	Description
Parent Directory		-	
css/	2020-07-14 17:52	-	
fonts/	2020-07-14 15:42	-	
images/	2020-07-14 15:42	-	
js/	2020-07-14 15:52	-	
php/	2020-07-14 15:42	-	
webapp.db	2020-07-14 17:52	28K	

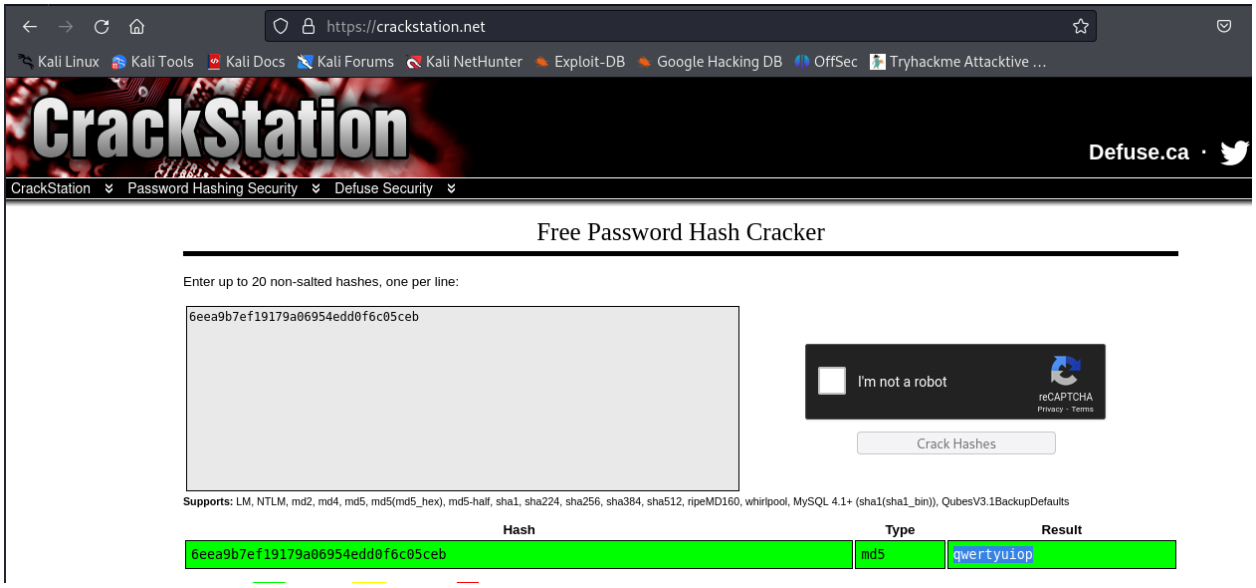
Apache/2.4.29 (Ubuntu) Server at 10.10.54.16 Port 80

- **Admin password hash:** 6eea9b7ef19179a06954edd0f6c05ceb

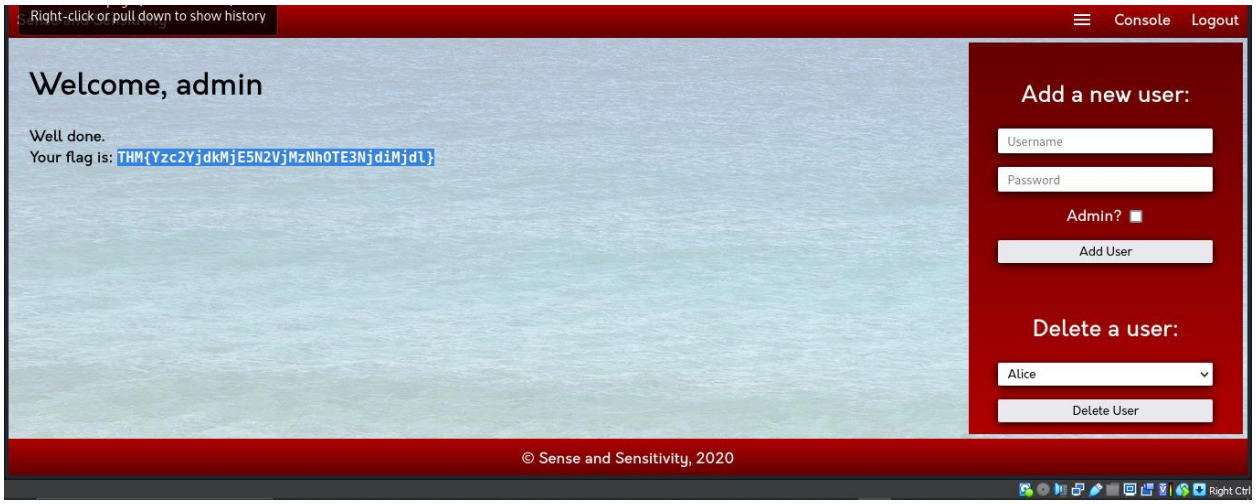




- **Admin's plaintext password:** qwertyuiop



- **Admin flag:** THM{Yzc2YjdkMjE5N2VjMzNhOTE3NjdiMjdl}



Task 12-16: [Severity 4] XML External Entity (XXE)

- Understanding XXE attacks where applications process XML input maliciously.
- Full form of XML: Extensible Markup Language.

What is XML?

XML (eXtensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is a markup language used for storing and transporting data.

- XML prolog is not mandatory in documents.

Above the line is called XML prolog and it specifies the XML version and the encoding used in the XML document. This line is not compulsory to use but it is considered a `good practice` to put that line in all your XML documents.

- Validation against a schema and specifying XML version and encoding in the prolog.

Every XML document mostly starts with what is known as XML Prolog.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Above the line is called XML prolog and it specifies the XML version and the encoding used in the XML document. This line is not compulsory to use but it is considered a `good practice` to put that line in all your XML documents.

- **Understanding DTD in XML**
- DTD stands for Document Type Definition and is crucial for defining how an XML document should be structured.
- It specifies the legal elements and attributes in an XML document
- **Validating XML Document with DTD**
- The XML document provided uses **note.dtd** to ensure it adheres to the defined structure.

- The DTD checks if the XML document contains the specified elements in the correct format and order.
- **Answers to Your Questions**
- **How do you define a new ELEMENT?**
  !ELEMENT

In a DTD, elements are declared with an ELEMENT declaration.

Declaring Elements

In a DTD, XML elements are declared with the following syntax:

```
<!ELEMENT element-name category>
```

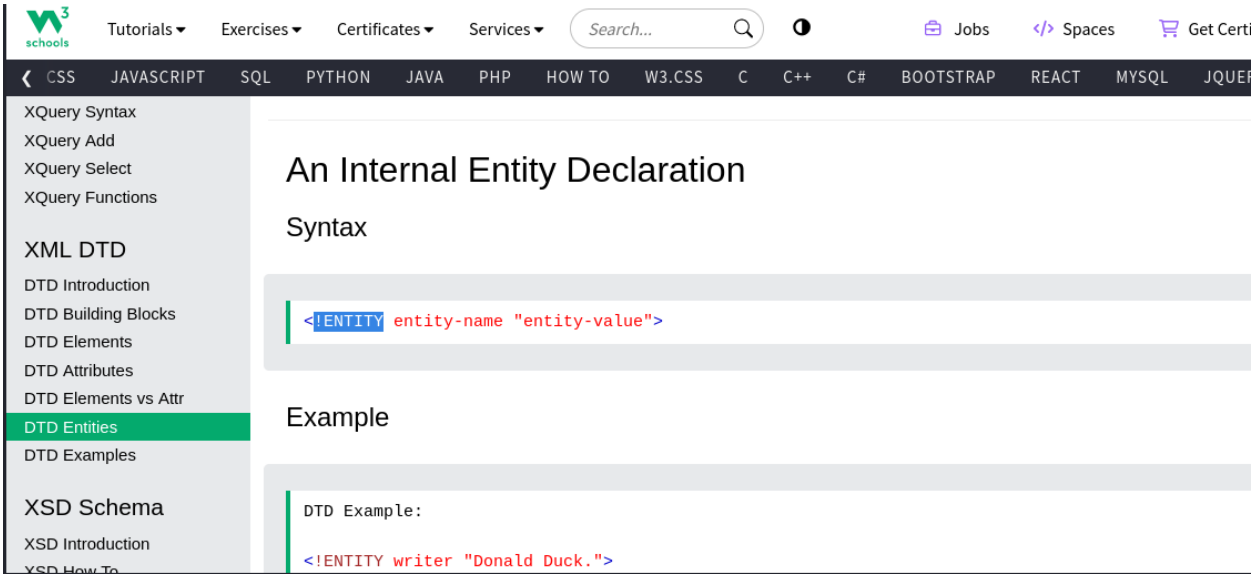
- **How do you define a ROOT element?**
  !DOCTYPE

So now let's understand how that DTD validates the XML. Here's what all those terms used in `note.dtd` mean

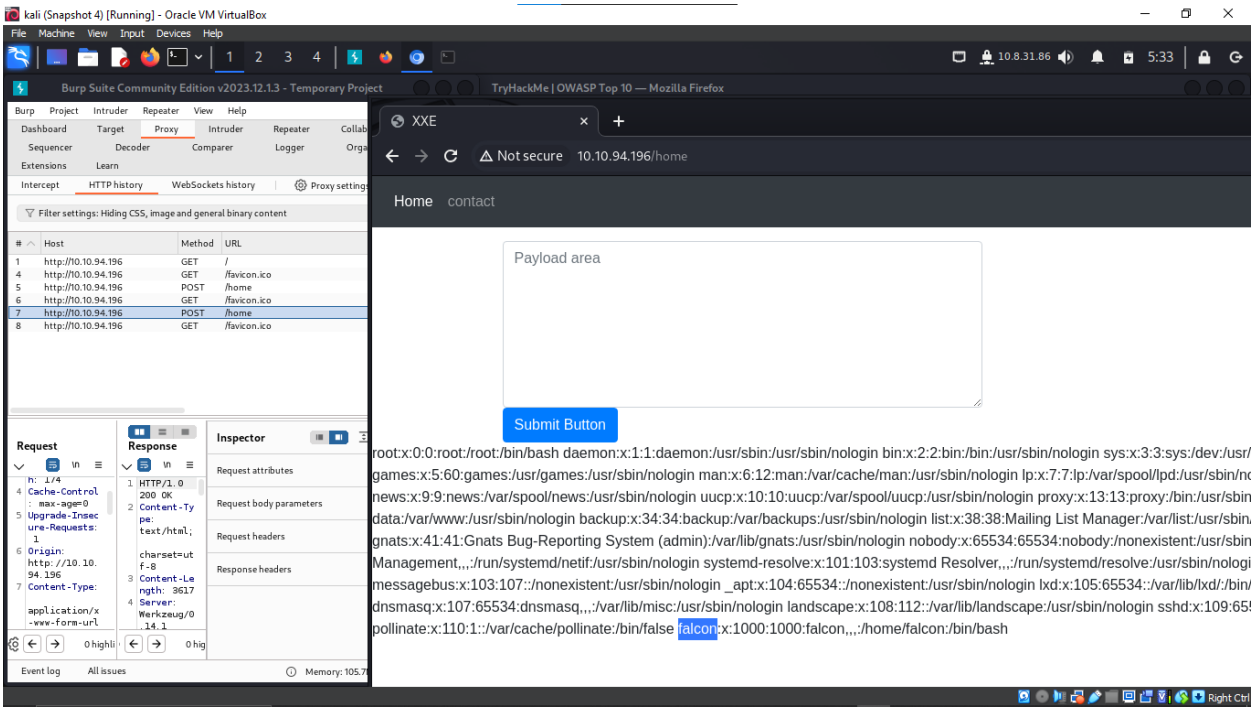
- !DOCTYPE note - Defines a root element of the document named note
- !ELEMENT note - Defines that the note element must contain the elements: "to, from, heading, body"
- !ELEMENT to - Defines the `to` element to be of type "#PCDATA"
- !ELEMENT from - Defines the `from` element to be of type "#PCDATA"
- !ELEMENT heading - Defines the `heading` element to be of type "#PCDATA"
- !ELEMENT body - Defines the `body` element to be of type "#PCDATA"

NOTE: #PCDATA means parseable character data.

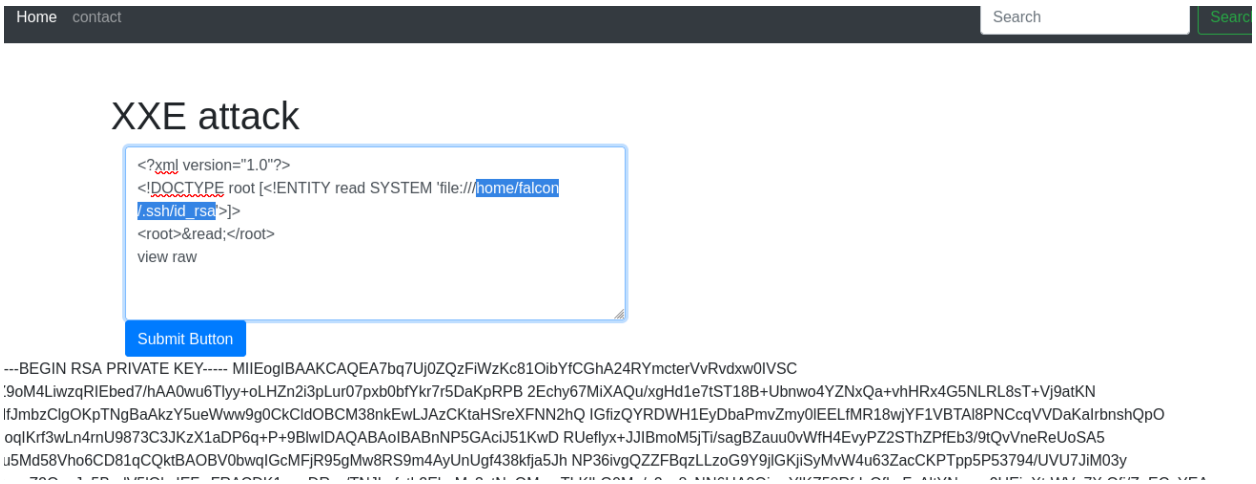
- **How do you define a new ENTITY?**
  !ENTITY



-
- Practical XXE Payload and Exploiting tasks.
- **Example Answers:**
  - **User in /etc/passwd:** falcon



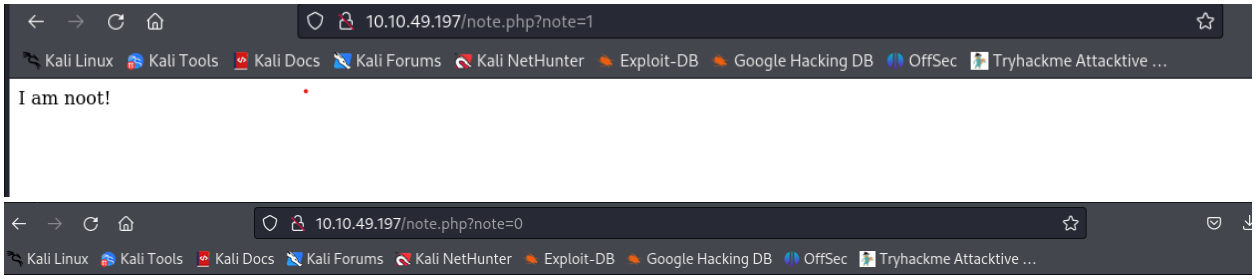
- **Falcon's SSH key location:** /home/falcon/.ssh/id_rsa



- **First 18 characters of Falcon's private key:** MIIEogIBAAKCAQEA7

Task 17-18: [Severity 5] Broken Access Control

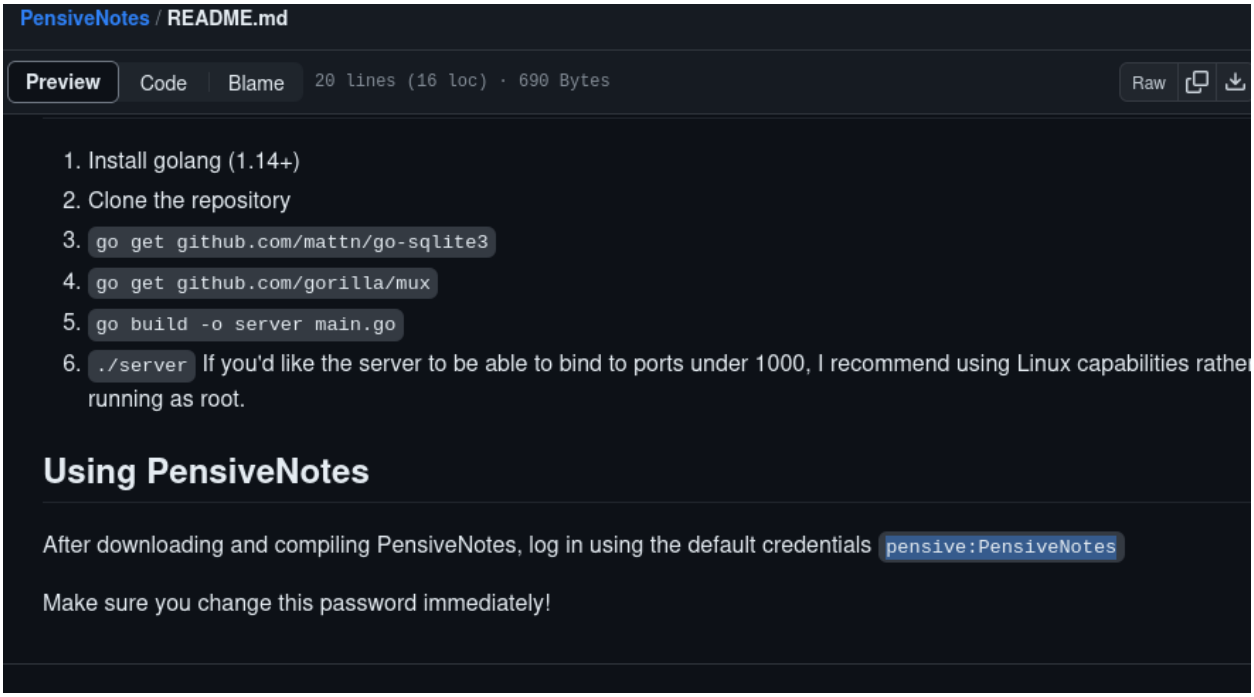
- Understanding how insufficient access control can lead to unauthorized data or functionality access.
- IDOR (Insecure Direct Object References) challenge.
- **Example Answer:**
  - **Other users' notes flag:** flag{fivefourthree}

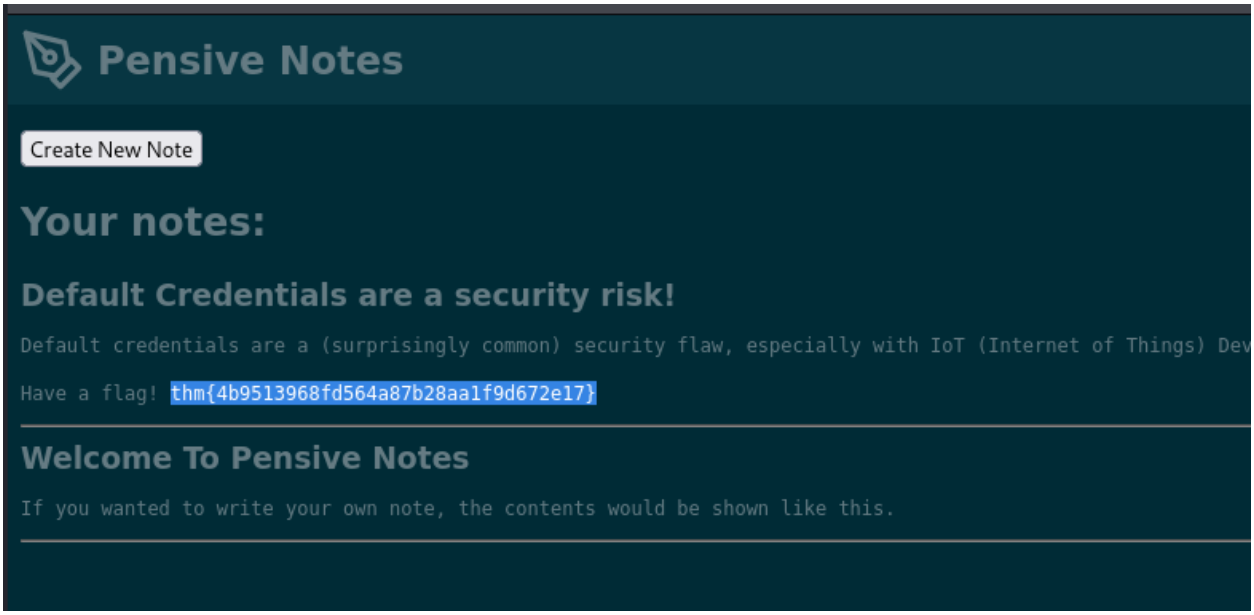


flag{fivefourthree}

Task 19: [Severity 6] Security Misconfiguration

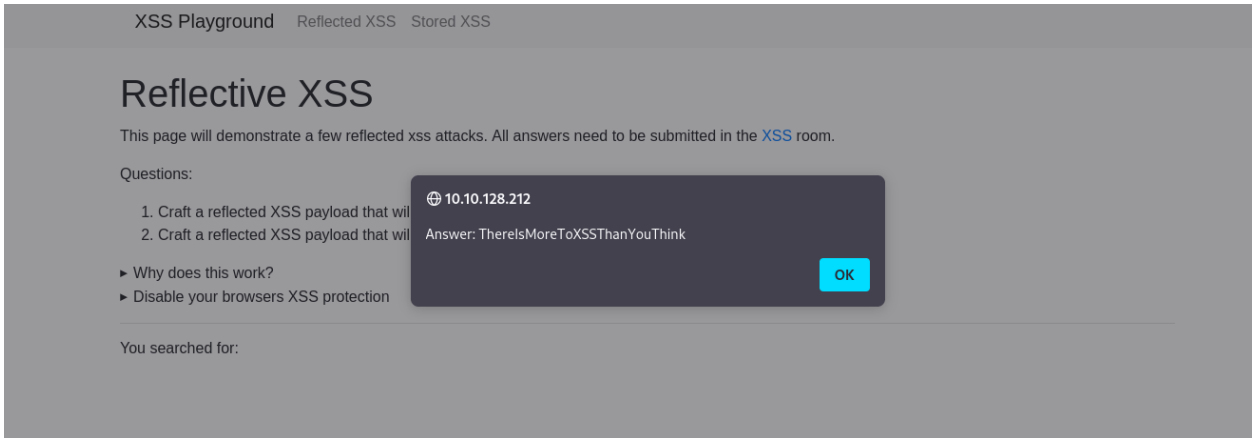
- Discussing risks from default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages.
- Practical task: Finding a flag through security misconfiguration.
- **Example Answer:**
  - **Flag:** thm{4b9513968fd564a87b28aa1f9d672e17}



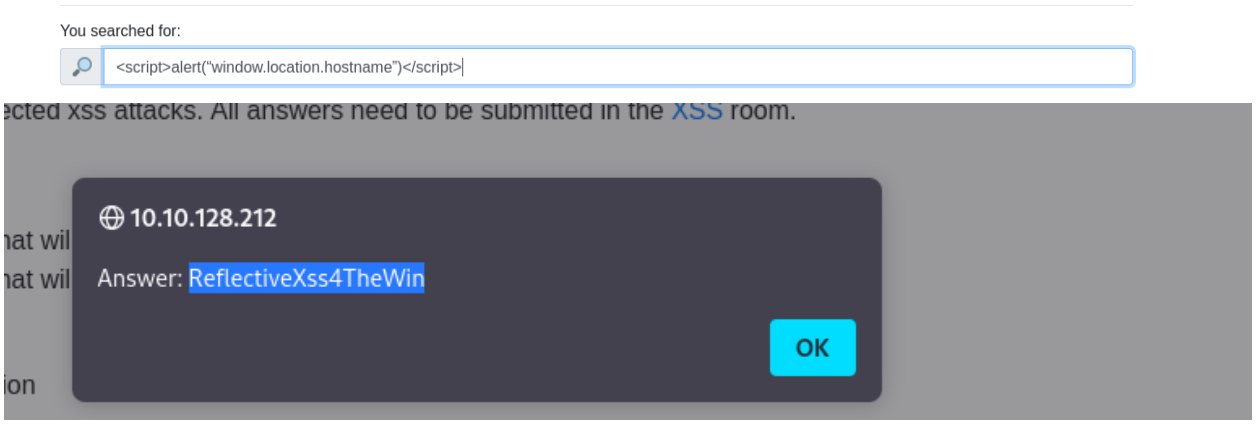


Task 20: [Severity 7] Cross-site Scripting (XSS)

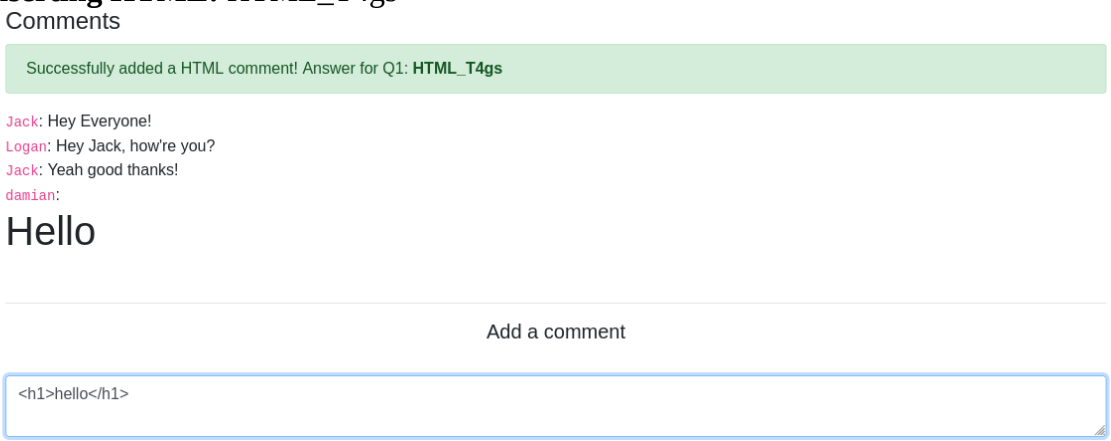
- Understanding XSS, where attackers inject malicious scripts into content viewed by other users.
- Reflected XSS Practical: Crafting payloads to create popups.
  - **Payload for "Hello" popup:** ThereIsMoreToXSSThanYouThink



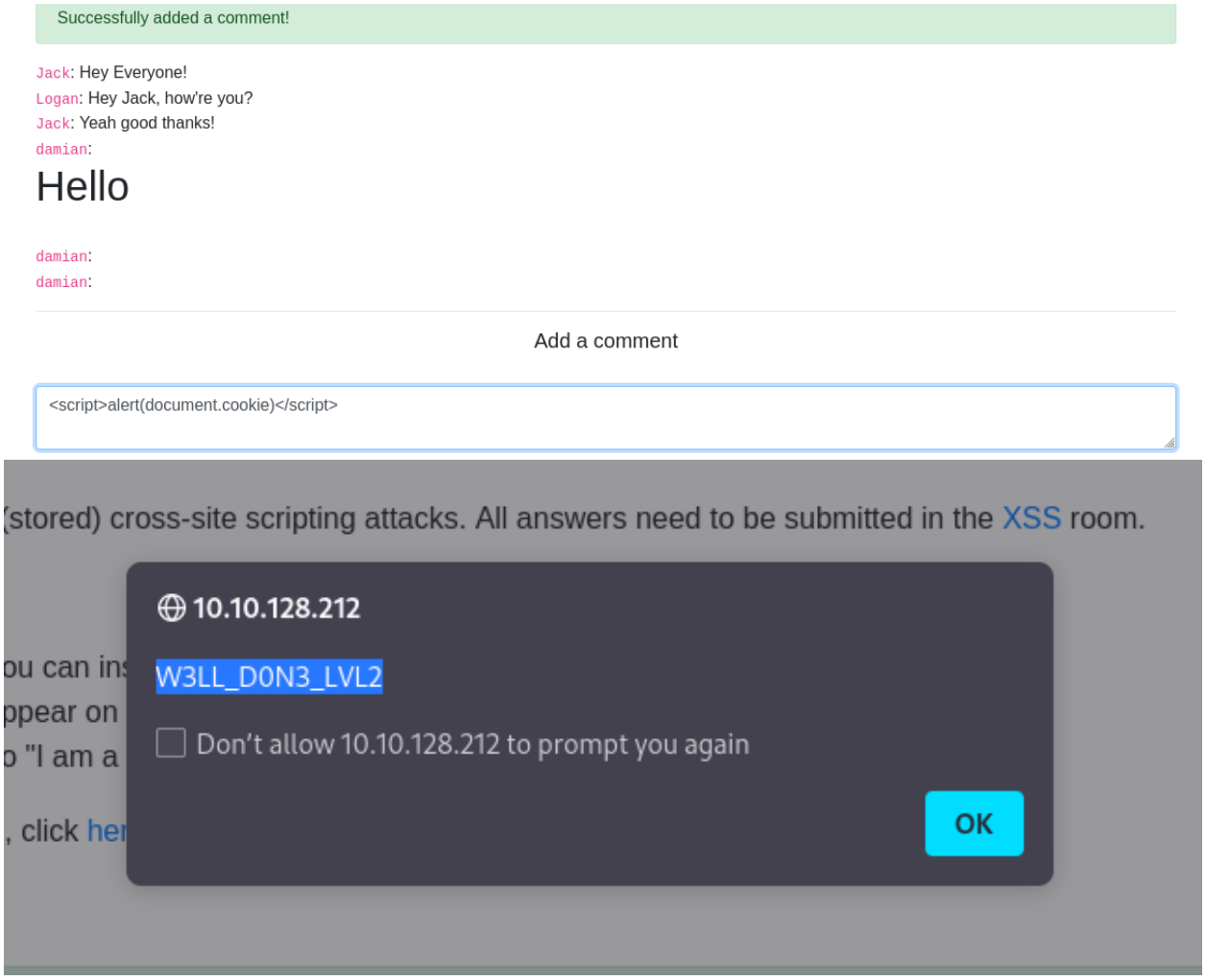
  - **Payload for IP address popup:** ReflectiveXss4TheWin



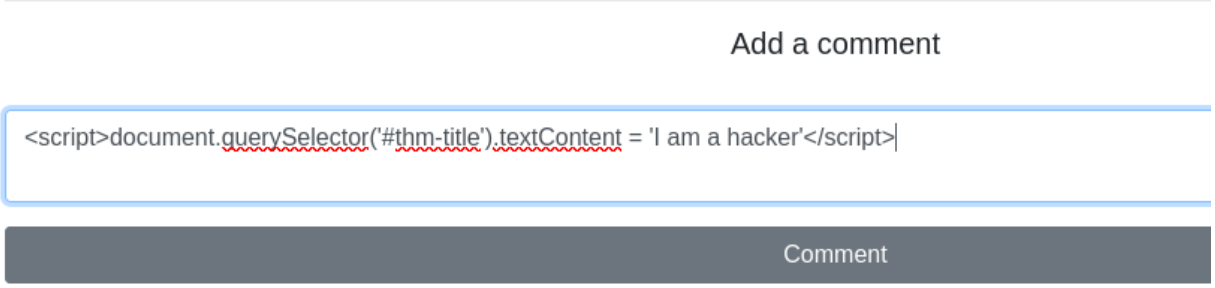
- Stored XSS Practical: Inserting HTML and JavaScript into comments**.**
  - **Inserting HTML:** HTML_T4gs



  - **Creating alert popup:** W3LL_D0N3_LVL2

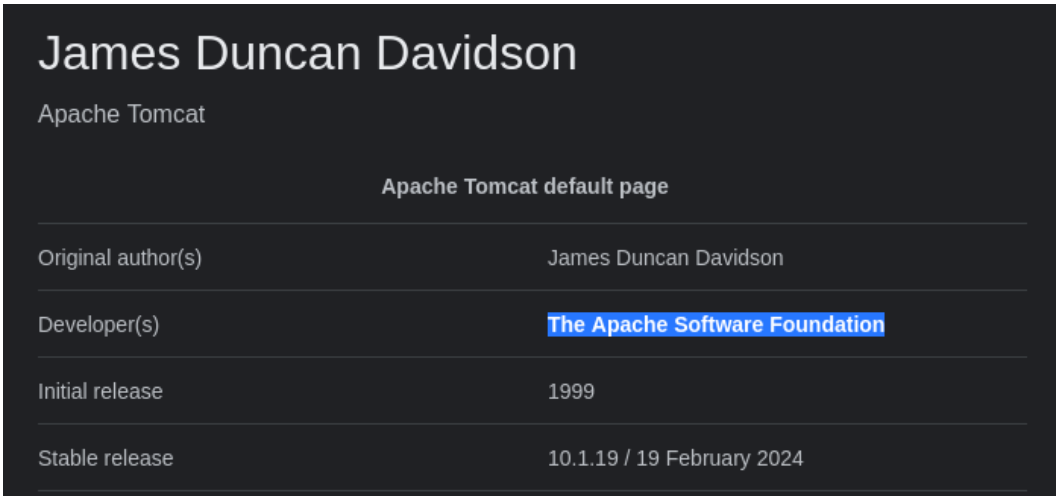


- **Defacing website text:** websites_can_be_easily_defaced_with_xss



Task 21: [Severity 8] Insecure Deserialization

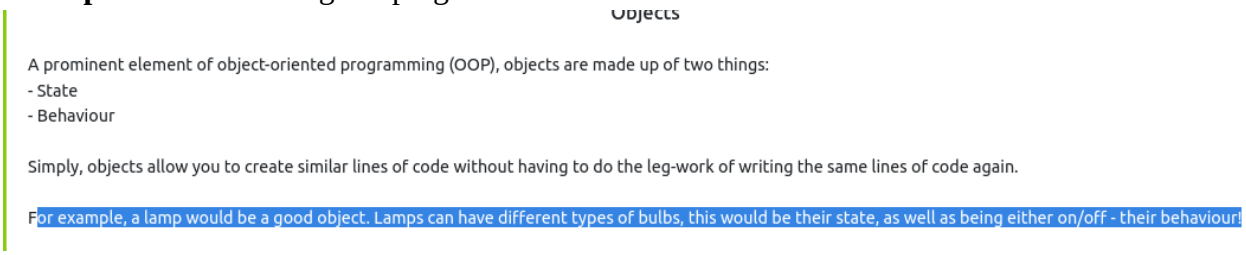
- Understanding how deserialized data can be manipulated to carry out attacks like denial of service.
- Tomcat developed by The Apache Software Foundation.



- Insecure deserialization can lead to Denial-of-Service attacks.

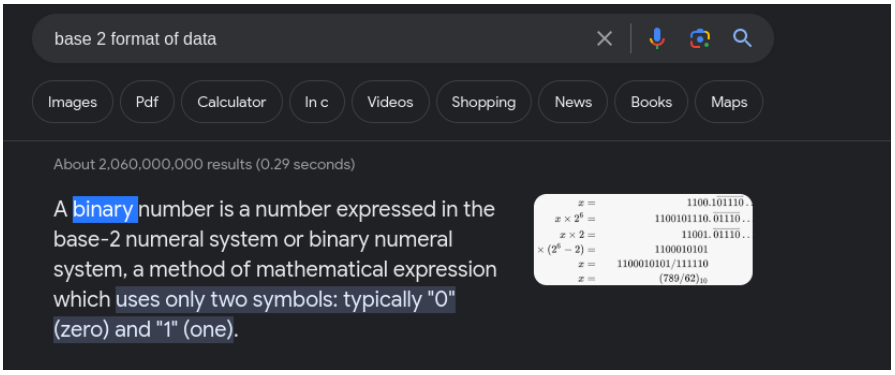
Task 22: [Severity 8] Insecure Deserialization - Objects

- Concept of object states and behaviors.
- **Example Answer:** A dog sleeping is a **Behaviour.**



Task 23: [Severity 8] Insecure Deserialization - Deserialization

- Understanding deserialization, converting data from a binary format back into an object.
- Base-2 formatting for data sent across networks is called Binary.



Task 24: [Severity 8] Insecure Deserialization - Cookies

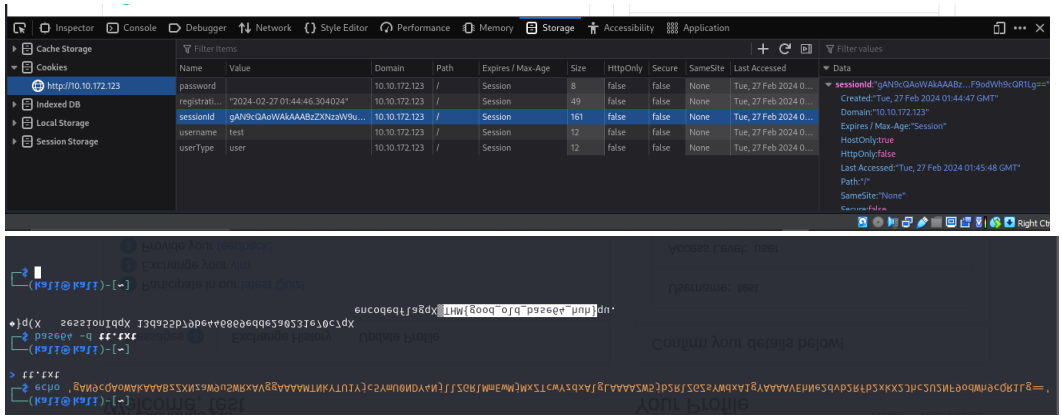
- Exploring how cookies can be insecurely deserialized.
- **Example Answers:**
  - **URL for cookie path webapp.com/login:** webapp.com/login

    If a cookie had the path of *webapp.com/login*, what would the URL that the user has to visit be?

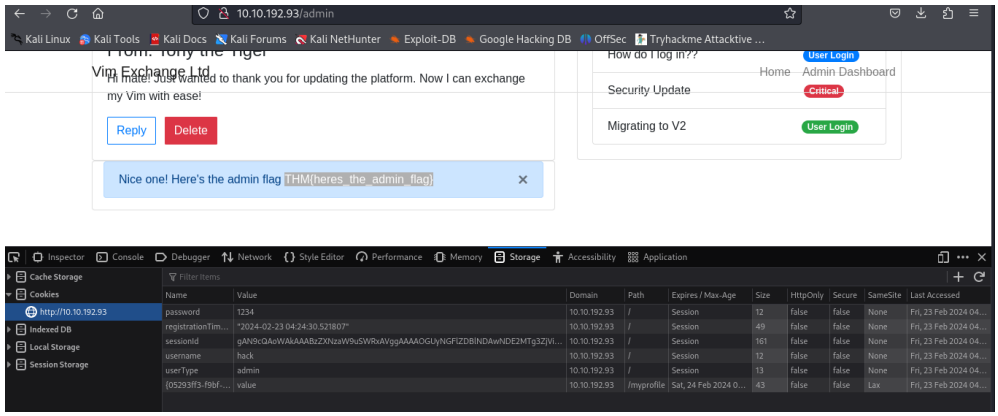
  - **Secure cookies technology acronym:** HTTPS

Task 25: [Severity 8] Insecure Deserialization - Cookies Practical

- Practical challenge with cookie manipulation.
- **Example Flags:**
  - **First flag:** THM{good_old_base64_huh}

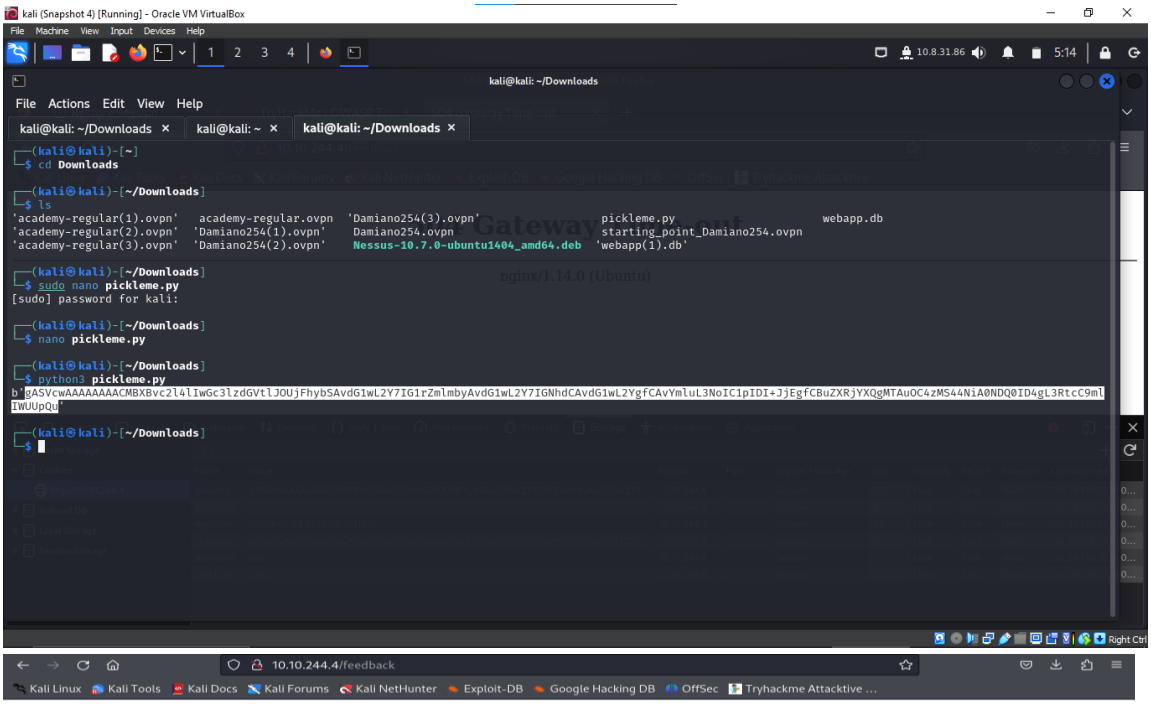


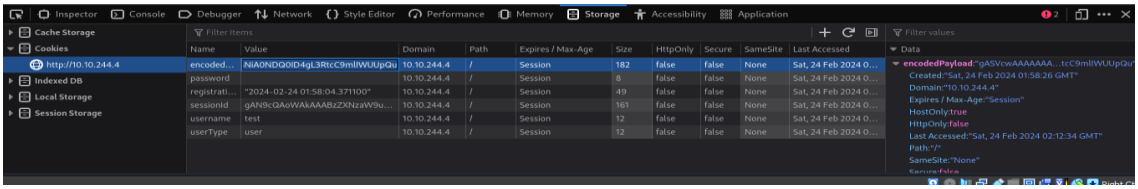
- **Second flag (admin dashboard):** THM{heres_the_admin_flag}

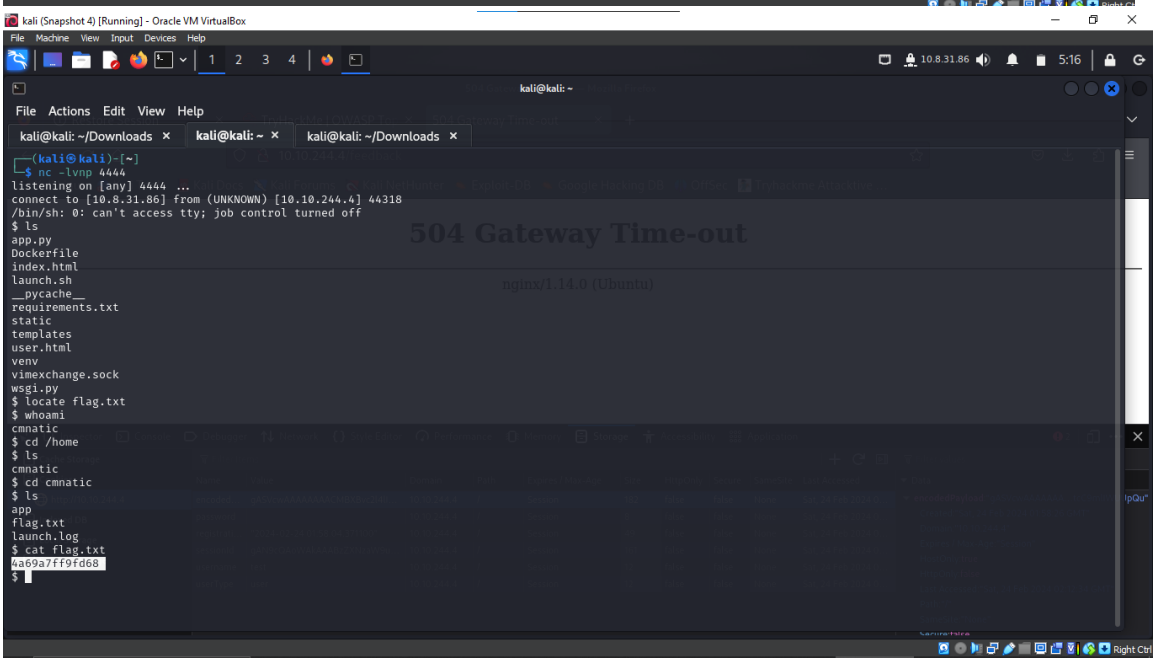


Task 26: [Severity 8] Insecure Deserialization - Code Execution

- Exploring how deserialization vulnerabilities can lead to code execution.
- **Example Answer:**
  - **flag.txt:** 4a69a7ff9fd68

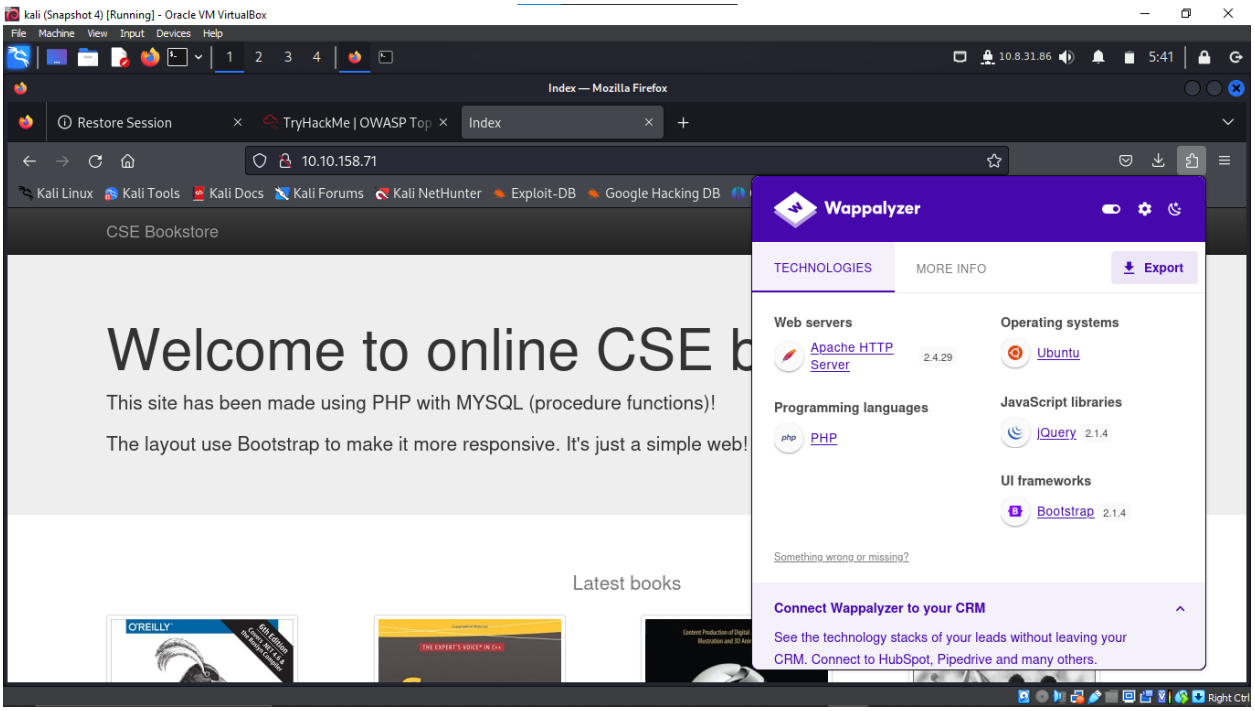


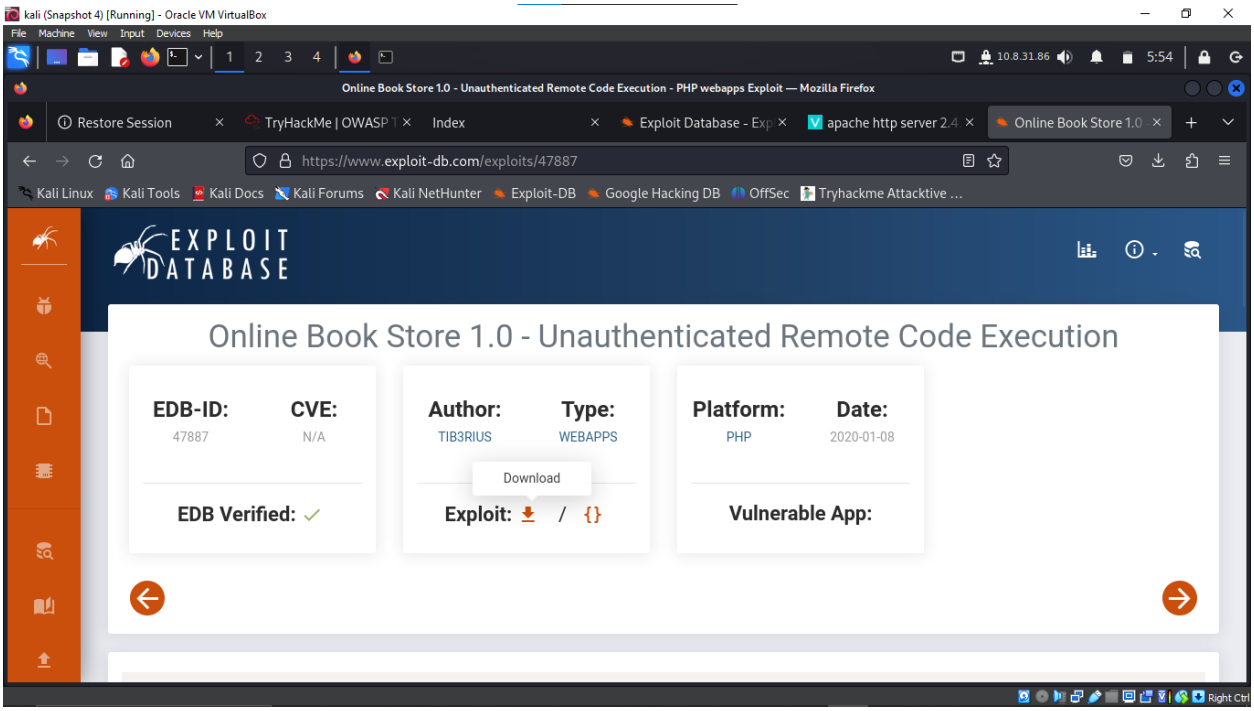


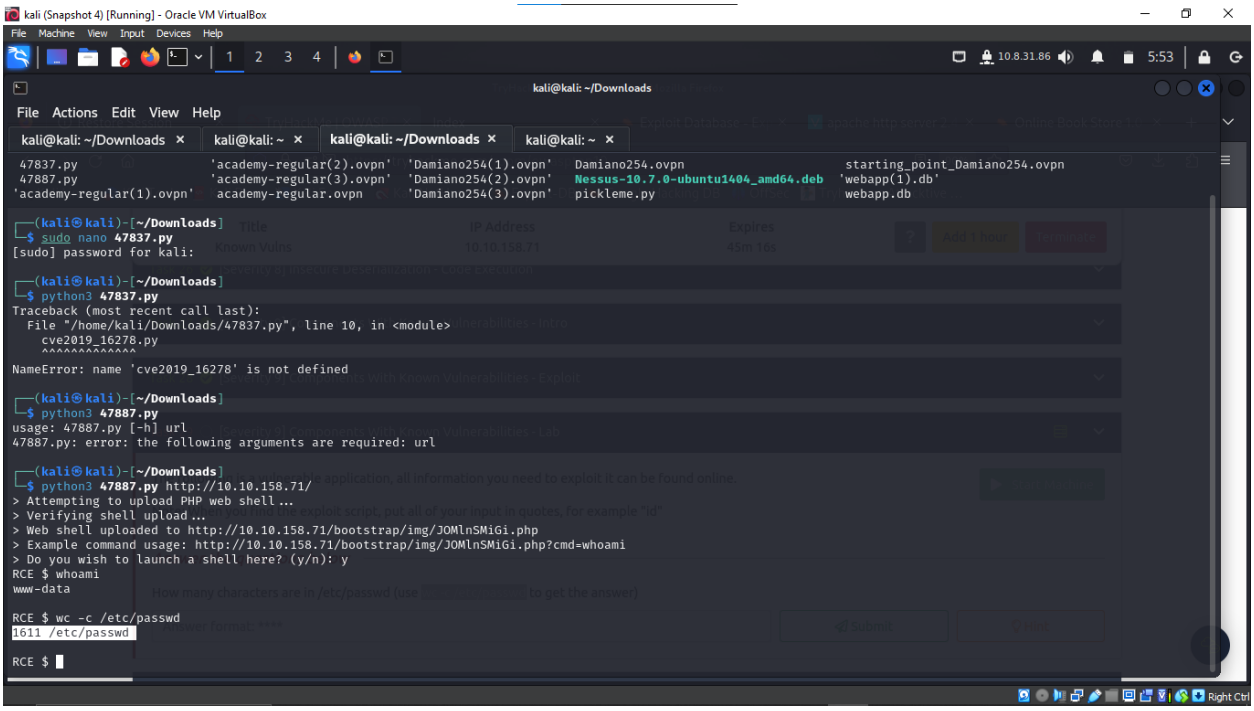


Task 27-29: [Severity 9] Components With Known Vulnerabilities

- Discussing risks associated with using software components with known vulnerabilities.
- Practical lab tasks involving exploitation of these vulnerabilities.
- **Example Answer:**
  - **Characters in /etc/passwd:** 1611

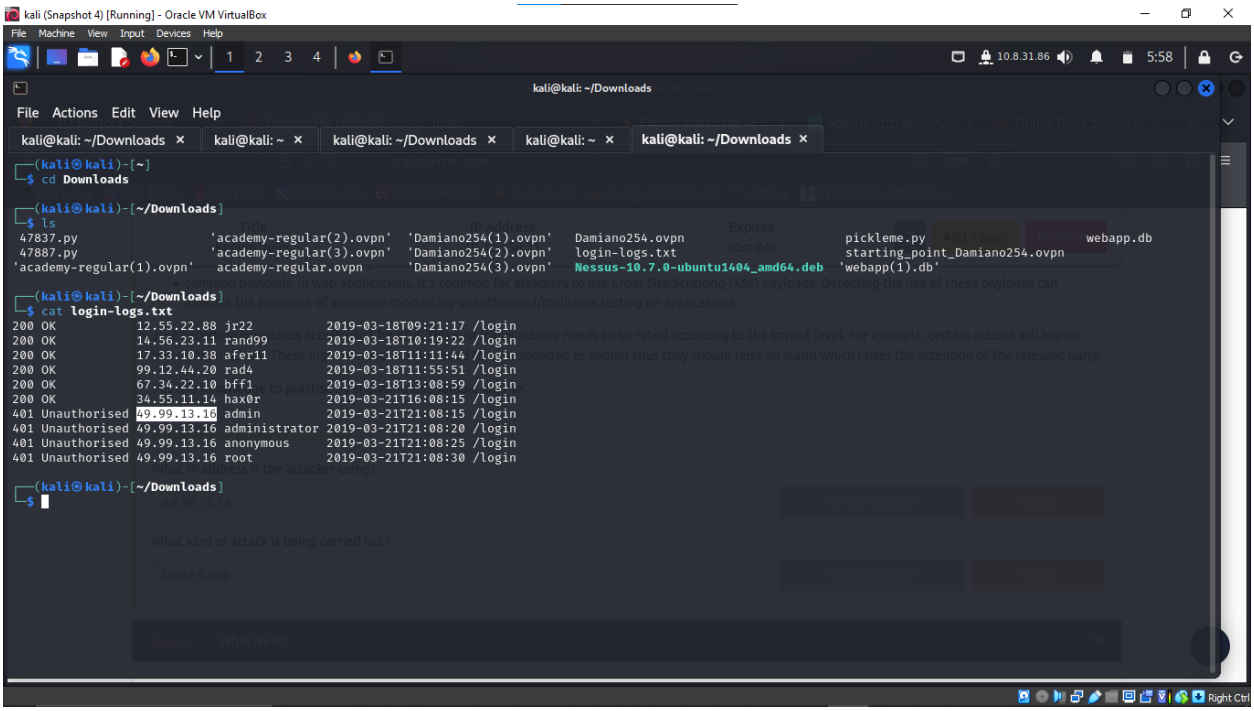

kali (Snapshot 4) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Index — Mozilla Firefox
Restore Session
TryHackMe | OWASP Top
Index
10.10.158.71
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
CSE Bookstore
Welcome to online CSE b
This site has been made using PHP with MYSQL (procedure functions)!
The layout use Bootstrap to make it more responsive. It's just a simple web!
Latest books
Wappalyzer
TECHNOLOGIES
MORE INFO
Export
Web servers
Apache HTTP Server 2.4.29
Operating systems
Ubuntu
Programming languages
PHP
JavaScript libraries
jQuery 2.1.4
UI frameworks
Bootstrap 2.1.4
Something wrong or missing?
Connect Wappalyzer to your CRM
See the technology stacks of your leads without leaving your CRM. Connect to HubSpot, Pipedrive and many others.



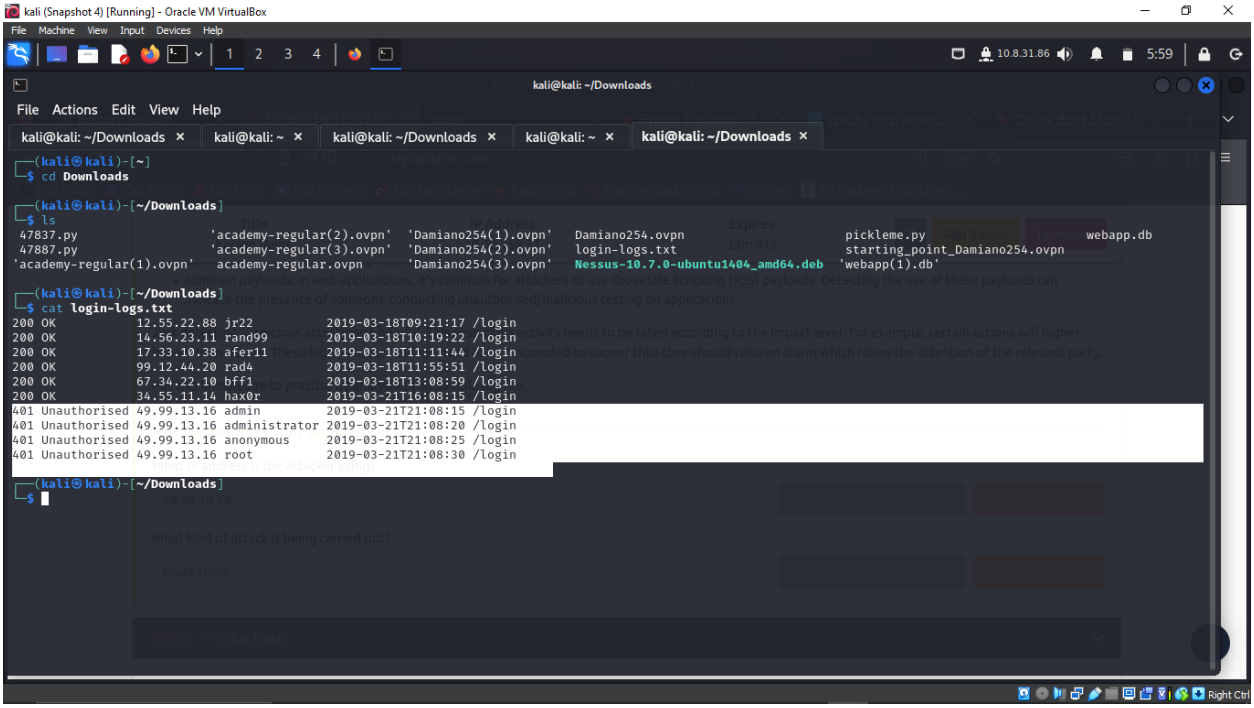


Task 30: [Severity 10] Insufficient Logging and Monitoring

- Understanding risks due to inadequate logging, monitoring, and alerting.
- Practical analysis of logs to identify attack details.
- **Example Answers:**
  - **Attacker's IP address:** 49.99.13.16

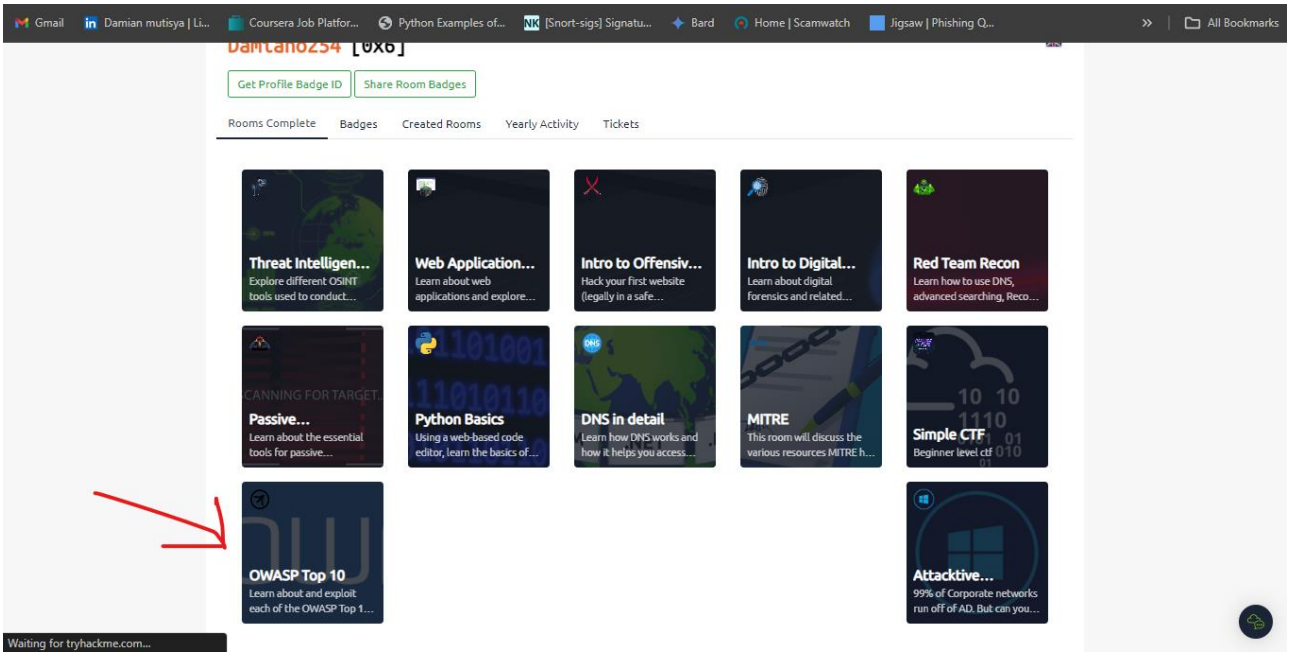


- **Type of attack:** Brute Force



Conclusion

Completing the TryHackMe OWASP Top 10 module equips learners with vital skills in identifying and addressing web vulnerabilities. It's a journey from foundational knowledge to advanced security practices, emphasizing the importance of ongoing learning in the dynamic field of cybersecurity.


Gmail
Damian mutisya | Li...
Coursera Job Platfor...
Python Examples of...
[Snort-sigs] Signatu...
Bard
Home | Scamwatch
Jigsaw | Phishing Q...
All Bookmarks
Damian0254 [0x6]
Get Profile Badge ID
Share Room Badges
Rooms Complete
Badges
Created Rooms
Yearly Activity
Tickets
Threat Intelligen...
Explore different OSINT tools used to conduct...
Web Application...
Learn about web applications and explore...
Intro to Offensiv...
Hack your first website (legally in a safe...
Intro to Digital...
Learn about digital forensics and related...
Red Team Recon
Learn how to use DNS, advanced searching, Reco...
Passive...
Learn about the essential tools for passive...
Python Basics
Using a web-based code editor, learn the basics of...
DNS in detail
Learn how DNS works and how it helps you access...
MITRE
This room will discuss the various resources MITRE h...
Simple CTF
Beginner level ctf
OWASP Top 10
Learn about and exploit each of the OWASP Top 1...
Attacktive...
99% of Corporate networks run off of AD. But can you...
Waiting for tryhackme.com...