Assignment 1: Introduction to Web Applications

Report by: Tonny Odhiambo, CS-CNS06-24028

Introduction

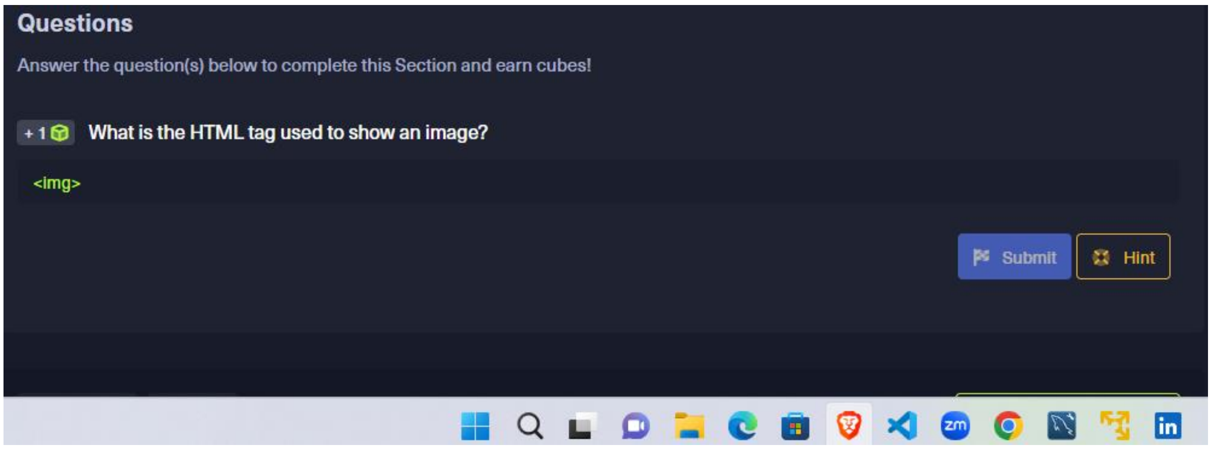
In the modern digital era, web applications have become integral to our daily life, facilitating everything from communication and commerce to education and entertainment. Understanding their architecture, functionality, and security is essential for my aspirations in cybersecurity. This module on Hack The Box has provided me with an immersive and practical introduction to web applications, equipping me with foundational knowledge and hands-on experience. By exploring real-world scenarios and challenges, I've gained insights into the various components of web applications, common vulnerabilities, and best practices for securing these critical systems.

Through a combination of theoretical concepts and practical exercises, this module has bridged the gap between academic learning and practical application, fostering a comprehensive understanding of web application security.

1. **Front End Components.**

a) **HTML**

In HTML, the <img> tag is used to display images on a web page. It is a self-closing tag, meaning it does not have a closing tag. The <img> tag requires two essential attributes: src, which specifies the path to the image file, and alt, which provides alternative text for accessibility purposes and appears if the image cannot be loaded. Optionally, the width and height attributes can be used to define the image dimensions in pixels, helping to control the layout of the page. The <img> tag is crucial for enriching web pages with visual content and plays a significant role in enhancing user experience by incorporating graphics seamlessly into HTML documents.


Questions

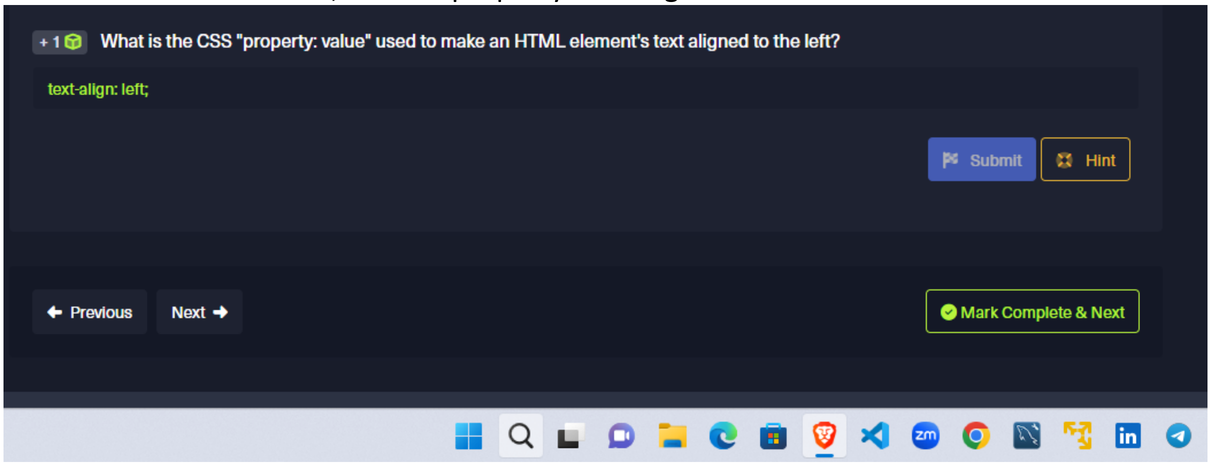
Answer the question(s) below to complete this Section and earn cubes!

+1 What is the HTML tag used to show an image?

<img>

b) **Cascading Style Sheets (CSS)**

CSS (Cascading Style Sheets) is a stylesheet language used in conjunction with HTML to format and style HTML elements. Different versions of CSS have been released over time, each introducing new features for formatting HTML. Browsers are continually updated to support these features. CSS allows defining the style of HTML elements by specifying properties such as font family, font size, background colour, text colour, and alignment**.**
For the question in this section, the answer is **text-align: left;** To align the text of an HTML element to the left, the CSS property text-align is used with the value left.


+1 What is the CSS "property: value" used to make an HTML element's text aligned to the left?

text-align: left;

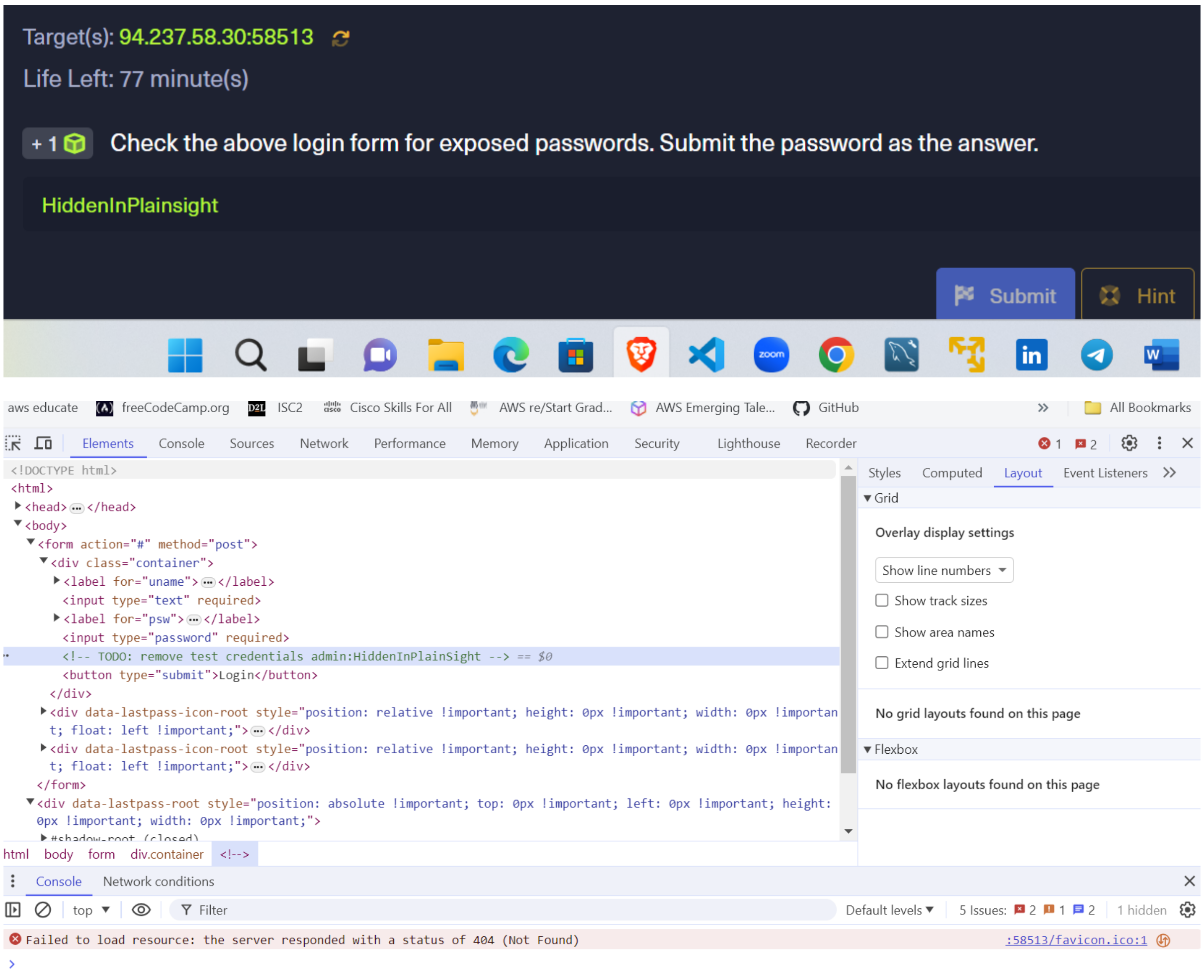
2. Front End Vulnerabilities

a) Sensitive Data Exposure

Front-end vulnerabilities in web applications can pose significant risks to end-users by exposing sensitive data through the source code accessible via the client's browser. Sensitive Data Exposure occurs when clear-text sensitive information, such as credentials, hidden links, or debugging parameters, is inadvertently included in the HTML or JavaScript of a web page.

This data can be easily viewed using browser tools or proxies, potentially leading to unauthorized access and exploitation of the application. It is crucial for developers to meticulously review and sanitize front-end code, remove unnecessary comments, and employ techniques like JavaScript obfuscation to protect sensitive information. Regular testing and code reviews are essential to ensure that no sensitive data is exposed inadvertently, thus maintaining the application's security and integrity.

In this section of the exercise, I spawned the target system and used developer tools to inspect and find the exposed password.



Target(s): 94.237.58.30:58513

Life Left: 77 minute(s)

+ 1 Check the above login form for exposed passwords. Submit the password as the answer.

HiddenInPlainsight

```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body>
  ▼<form action="#" method="post">
    ▼<div class="container">
      ▶<label for="uname">…</label>
       <input type="text" required>
      ▶<label for="psw">…</label>
       <input type="password" required>
       <!-- TODO: remove test credentials admin:HiddenInPlainSight --> == $0
       <button type="submit">Login</button>
     </div>
    ▶<div data-lastpass-icon-root style="position: relative !important; height: 0px !important; width: 0px !importan
     t; float: left !important;">…</div>
    ▶<div data-lastpass-icon-root style="position: relative !important; height: 0px !important; width: 0px !importan
     t; float: left !important;">…</div>
   </form>
  ▼<div data-lastpass-root style="position: absolute !important; top: 0px !important; left: 0px !important; height:
   0px !important; width: 0px !important;">
     ▶#shadow-root (closed)
```

html body form div.container <!-->

Grid

Overlay display settings

Show line numbers

☐ Show track sizes

☐ Show area names

☐ Extend grid lines

No grid layouts found on this page

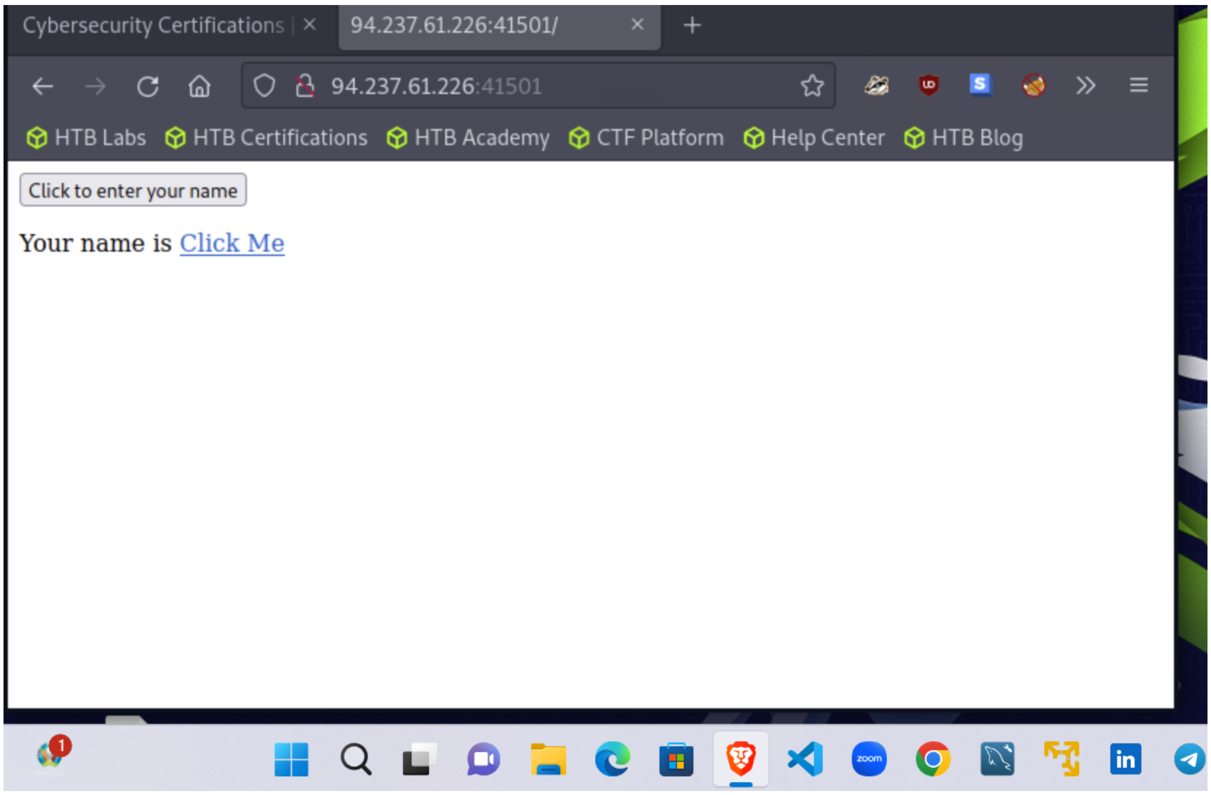
Flexbox

No flexbox layouts found on this page

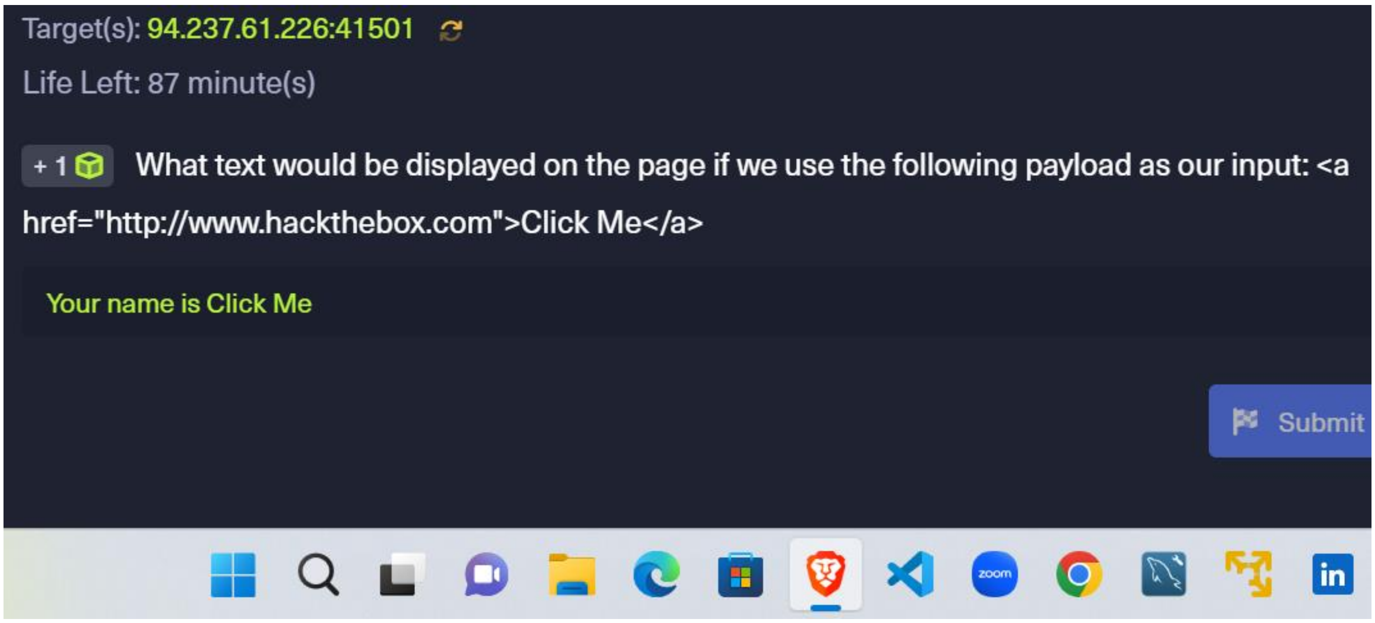
```
Failed to load resource: the server responded with a status of 404 (Not Found)          :58513/favicon.ico:1
```

b) HTML Injection

HTML Injection is a critical front-end security issue that occurs when user input is not properly validated or sanitized, allowing malicious HTML code to be executed by the browser. This can result in various attacks, such as displaying deceptive login forms, altering the appearance of the web page, or embedding malicious ads. For example, if a webpage takes user input and directly displays it without sanitization, an attacker can inject HTML code to manipulate the content. To test for HTML Injection, one can input a payload like **<a href="http://www.hackthebox.com">Click Me</a>**, which would display as a clickable link labelled "Click Me" on the web page. It is crucial to implement input validation and sanitization on both the front-end and back-end to prevent such vulnerabilities.



Your name is Click Me


Target(s): 94.237.61.226:41501

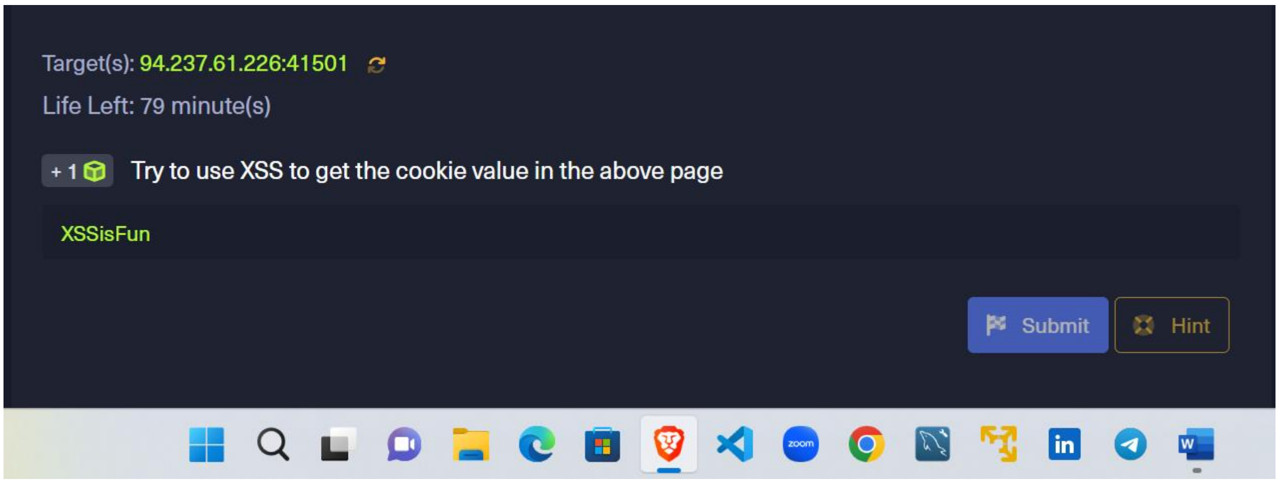
Life Left: 87 minute(s)

+ 1 What text would be displayed on the page if we use the following payload as our input: <a href="http://www.hackthebox.com">Click Me</a>

Your name is Click Me

c) Cross-Site Scripting (XSS)

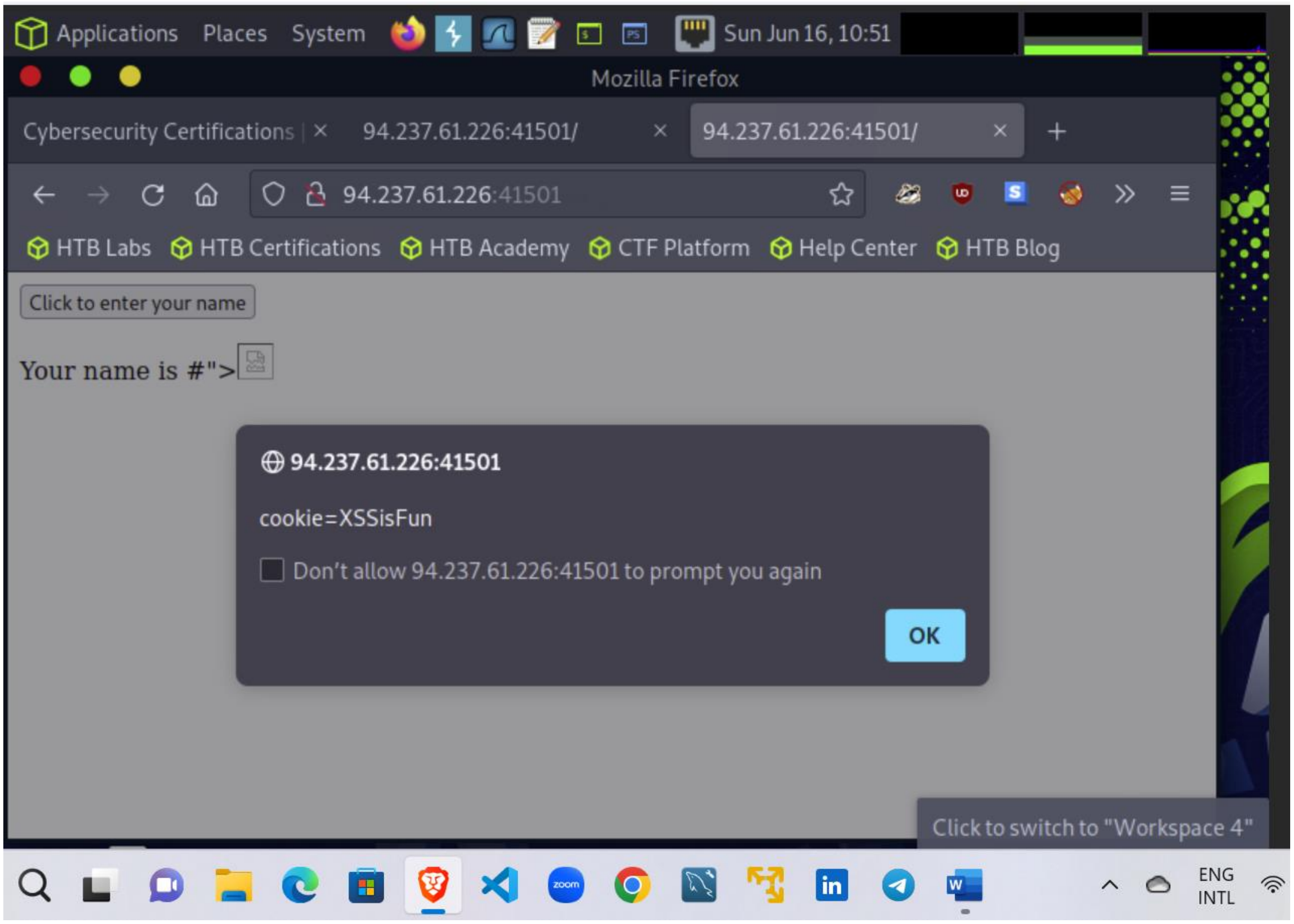
Cross-Site Scripting (XSS) is a significant web security vulnerability that involves injecting malicious JavaScript code into a web page, which is then executed on the client's browser. XSS attacks can be used to steal sensitive information, such as cookies, session tokens, or other credentials. There are three primary types of XSS: Reflected XSS, where the malicious input is reflected back from the server; Stored XSS, where the malicious input is stored on the server and served to users; and DOM XSS, where the malicious input directly manipulates the Document Object Model (DOM) in the browser. In an HTML Injection example, the lack of input sanitization can also make the page vulnerable to XSS. By injecting the payload **#"><img src=/ onerror=alert(document.cookie)>**, an attacker can trigger an alert displaying the user's cookie, potentially allowing the attacker to hijack the user's session as shown below.


Target(s): 94.237.61.226:41501

Life Left: 79 minute(s)

+1 Try to use XSS to get the cookie value in the above page

XSSisFun



Your name is #">

94.237.61.226:41501

cookie=XSSisFun

Don't allow 94.237.61.226:41501 to prompt you again

d) Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is a front-end vulnerability where an attacker tricks an authenticated user into executing unwanted actions on a web application. This can involve using XSS vulnerabilities to perform queries or API calls on the user's behalf. A common CSRF attack might change a user's password through a crafted JavaScript payload, exploiting the user's session to perform the action.

CSRF can target admins to gain higher privileges, potentially compromising the server. Prevention includes both front-end and back-end measures: sanitizing and validating user input, implementing web application firewalls (WAFs), and using anti-CSRF tokens and HTTP headers.

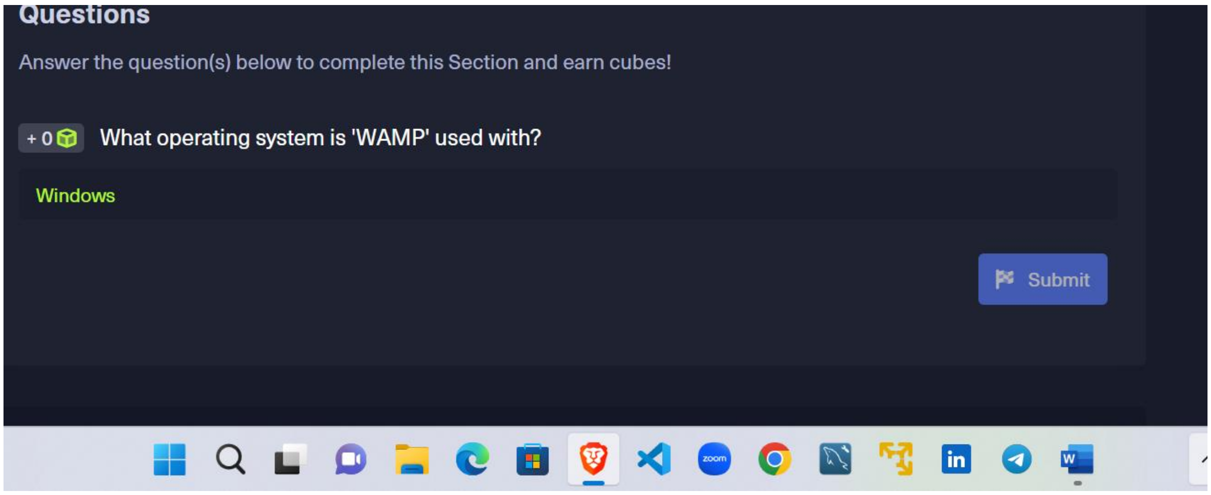
Additionally, modern browsers and applications have built-in anti-CSRF features, but developers should not solely rely on these and should ensure their code is robust against such attacks. For detailed prevention strategies, the OWASP CSRF Prevention Cheat Sheet is a valuable resource.

3. Back End Components

a) Back End Servers

A back-end server is the core system hosting essential applications for web applications, residing in the Data access layer. It includes components like web servers (e.g., Apache), databases (e.g., MySQL), and development frameworks (e.g., PHP or .NET). Common configurations (stacks)

include LAMP (Linux, Apache, MySQL, PHP), WAMP (Windows, Apache, MySQL, PHP), and others tailored to specific operating systems like Windows (WINS). These setups determine performance and stability. Hardware-wise, the server's capabilities influence application responsiveness, with scalable options using data centers or cloud services.


Questions

Answer the question(s) below to complete this Section and earn cubes!

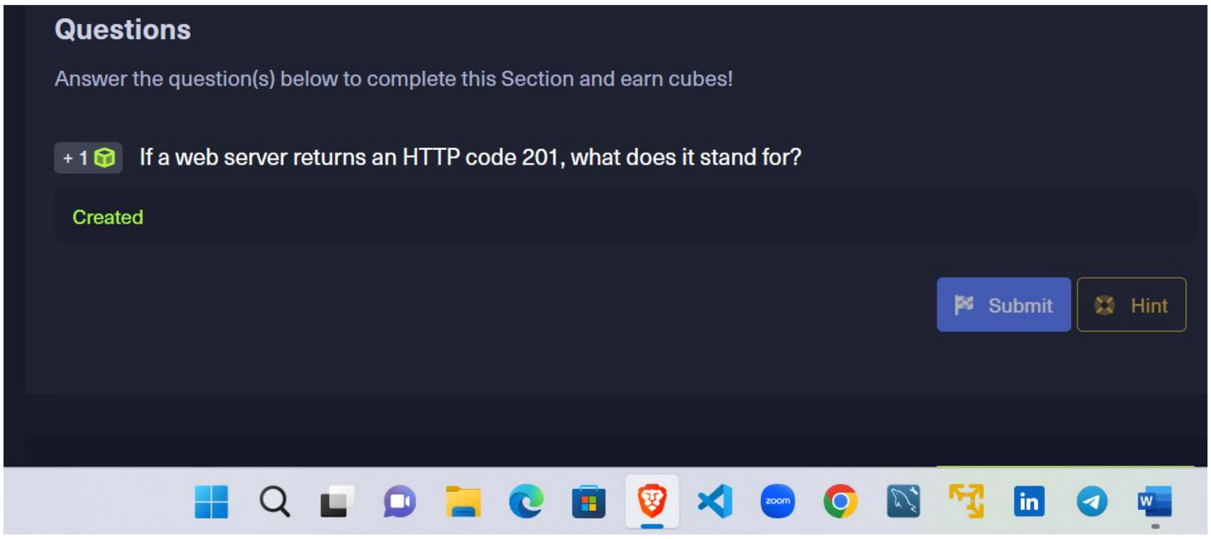
+ 0 What operating system is 'WAMP' used with?

Windows

b) Web Servers

A web server is a crucial component of the back-end server that handles HTTP traffic from client-side browsers, routing requests to requested pages and returning responses. It operates on ports 80 or 443, managing various HTTP response codes like 200 OK for success, 404 NOT FOUND for missing pages, and others for errors and redirects. Common web servers include Apache, known for its versatility with languages like PHP and support for modules; NGINX, favoured for its efficient handling of concurrent requests; and IIS, tailored for Windows environments and integration with Active Directory. Each serves distinct needs based on scalability, security, and compatibility with different web applications.

For the question in this section, the HTTP code 201 stands for **created.**



Questions

Answer the question(s) below to complete this Section and earn cubes!

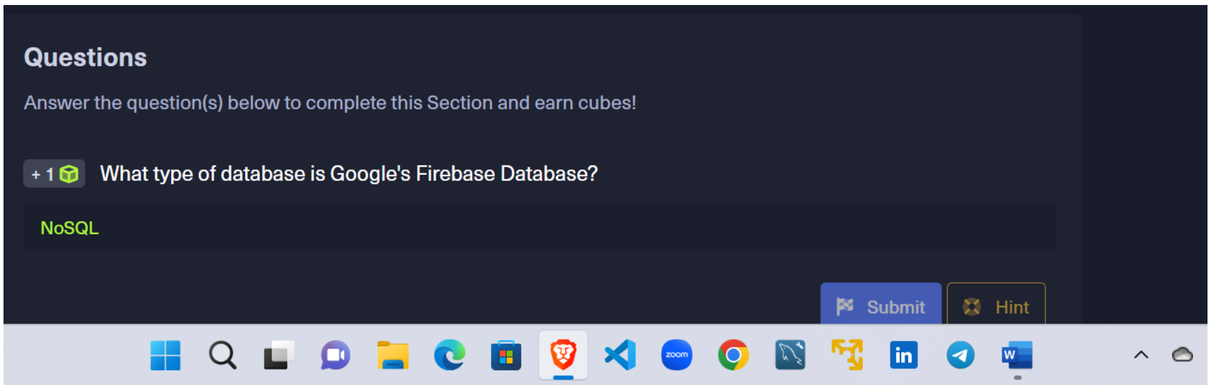
+1 If a web server returns an HTTP code 201, what does it stand for?

Created

C) Databases

Web applications rely on back-end databases to store and manage various types of data, such as user information, content, and application assets. Relational databases like MySQL and PostgreSQL organize data into structured tables with relationships, ideal for structured data management. Non-relational (NoSQL) databases like MongoDB and Elasticsearch offer flexibility with models like Key-Value and Document-Based, suited for unstructured and scalable data storage. Modern web development frameworks seamlessly integrate these databases, enabling efficient data retrieval and manipulation via web applications.

For the question in this section, Firebase is a type of NoSQL database.



Questions

Answer the question(s) below to complete this Section and earn cubes!

+1 What type of database is Google's Firebase Database?

NoSQL

d) Development Frameworks & APIs

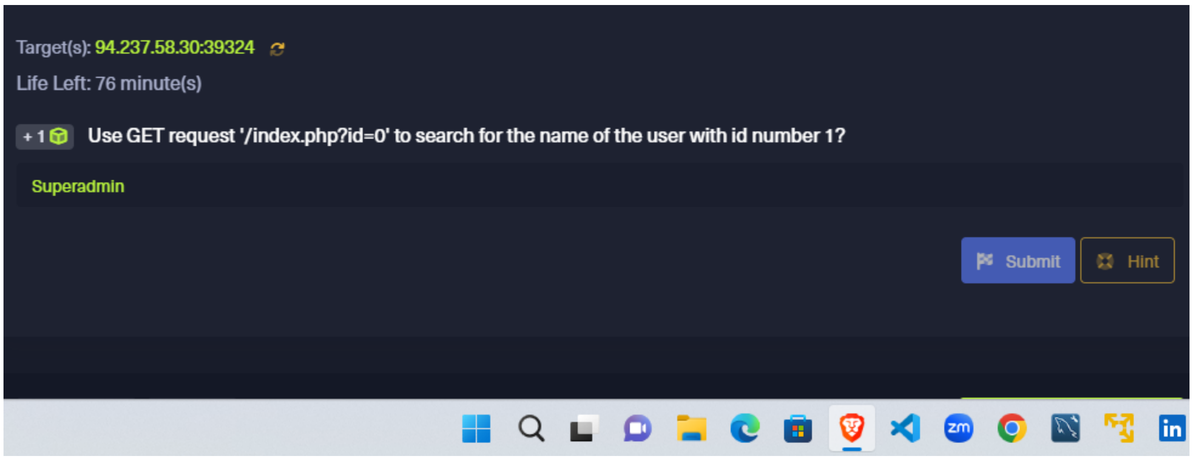
Development frameworks play a crucial role in modern web application development by providing tools to quickly build and deploy robust applications. Popular frameworks like

Laravel (PHP), Express (Node.js), Django (Python), and Rails (Ruby) streamline common functionalities such as user registration, enhancing development efficiency and ensuring consistency across projects.

For the question on this section, i followed the bellow steps to find the name of the user with id number 1.

- **Accessing the Endpoint**: We accessed the specific endpoint /index.php?id=1 on the target server using the browser's address bar.
- **Sending the Request**: By entering the URL http://94.237.58.30:39324/index.php?id=1 into the browser, we initiated an HTTP GET request to fetch details associated with user ID 1.
- **Interpreting the Response**: The server responded by delivering data related to the user identified by ID 1, which was displayed in the browser window.

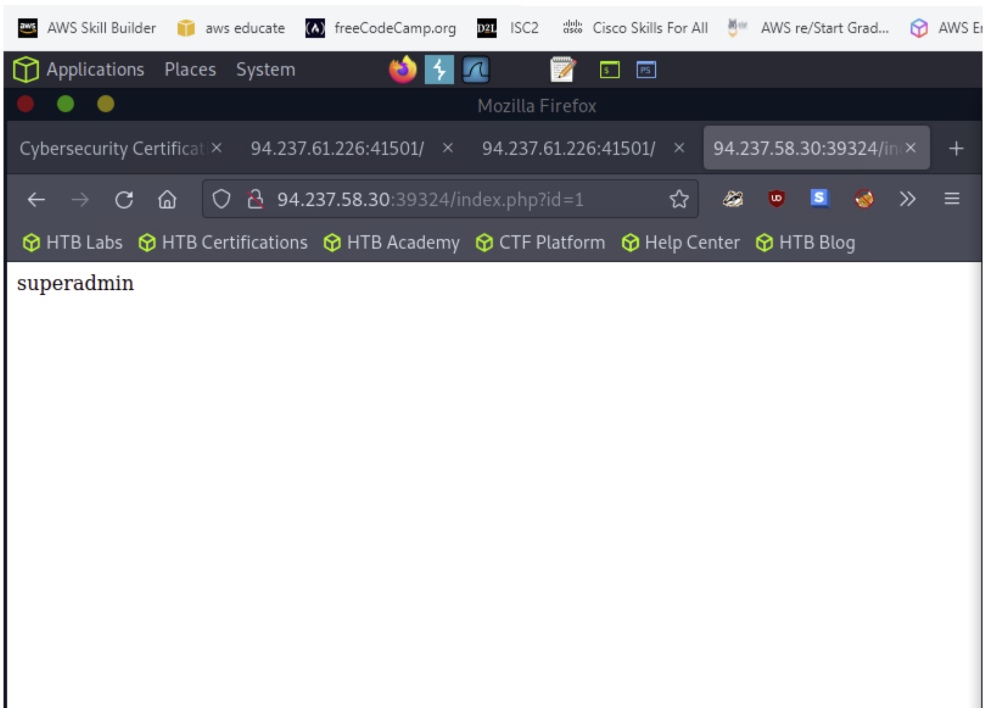
This approach showcased the simplicity of using a web browser to interact with web applications, highlighting its role in accessing and visualizing data from remote servers without requiring additional software or command-line tools.


Target(s): 94.237.58.30:39324

Life Left: 76 minute(s)

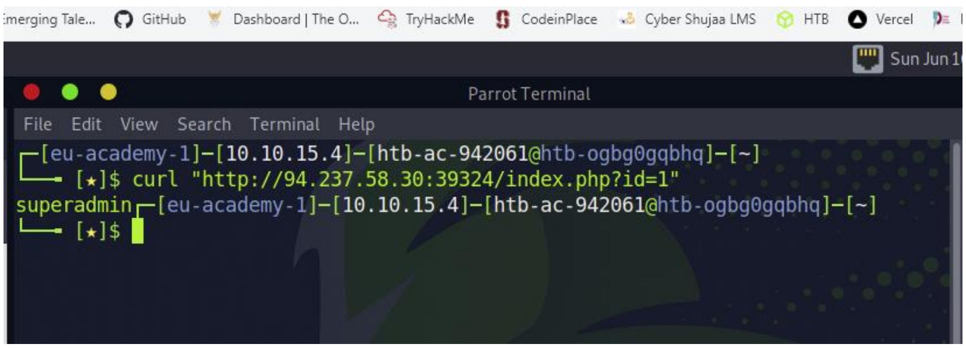
+1 Use GET request '/index.php?id=0' to search for the name of the user with id number 1?

Superadmin



superadmin

Alternatively, you could also use the curl command on the terminal to achieve the same goal by running the command **curl** http://94.237.58.30:39324/index.php?id=1

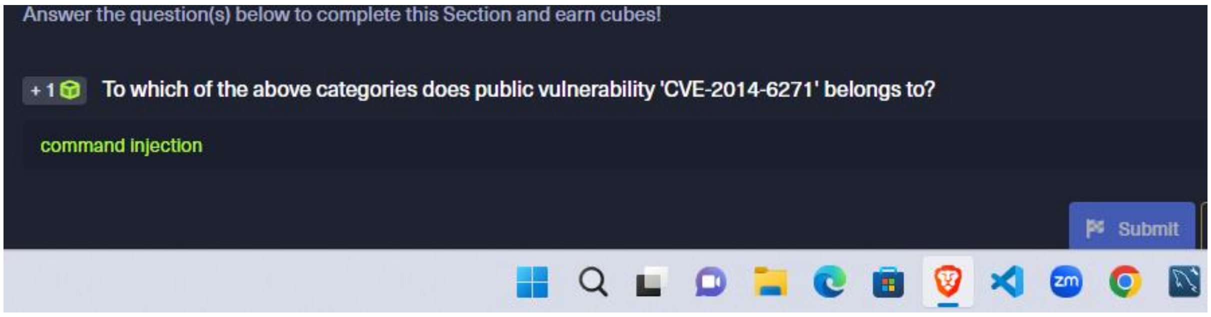


```
┌─[eu-academy-1]─[10.10.15.4]─[htb-ac-942061@htb-ogbg0gqbhq]─[~]
└──╼ [★]$ curl "http://94.237.58.30:39324/index.php?id=1"
superadmin┌─[eu-academy-1]─[10.10.15.4]─[htb-ac-942061@htb-ogbg0gqbhq]─[~]
└──╼ [★]$ 
```

4. Back End Vulnerabilities

a) Common Web Vulnerabilities

In my assessment, I encountered CVE-2014-6271, a Command Injection vulnerability affecting certain web applications. This vulnerability enables attackers to execute malicious commands directly on the server hosting the web application, potentially leading to unauthorized access, data leakage, or even complete server compromise**.** This also answers the question in this section as shown below.


Answer the question(s) below to complete this Section and earn cubes!

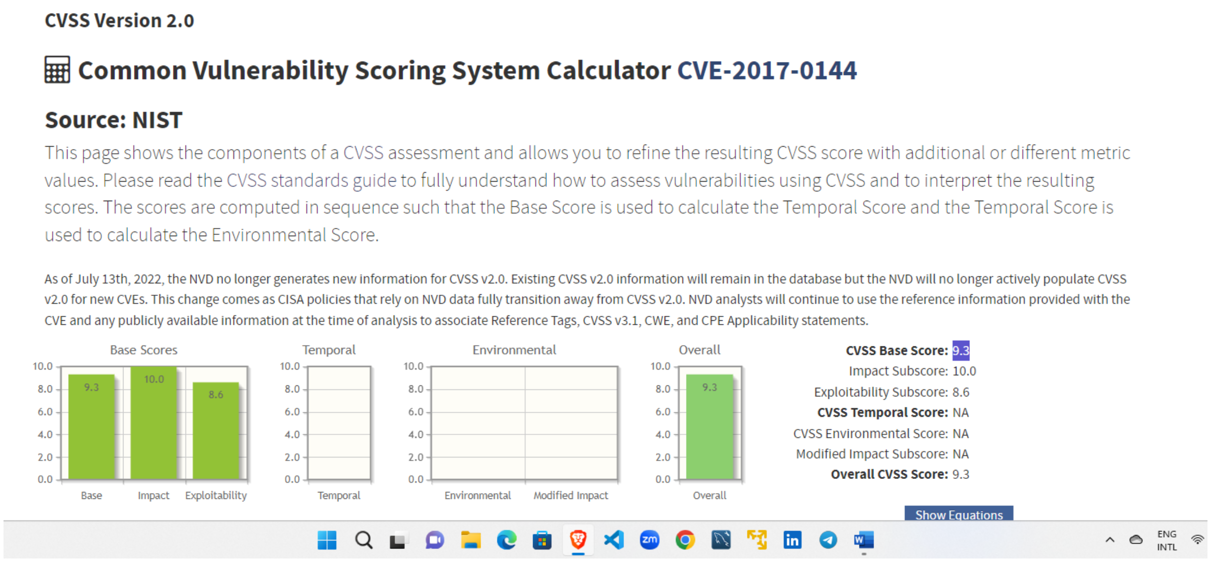
+1 To which of the above categories does public vulnerability 'CVE-2014-6271' belongs to?

command injection

b) Public Vulnerabilities

Publicly deployed web applications, both open-source and proprietary, are subject to extensive scrutiny by security professionals worldwide, often leading to the discovery of numerous vulnerabilities. These vulnerabilities are documented and assigned CVE (Common Vulnerabilities and Exposures) identifiers, accompanied by CVSS (Common Vulnerability Scoring System) scores that assess their severity.

For the question on this part asking the CVSS score of the public vulnerability CVE-2017-0144, I used the CVSS Version 2.0 calculator to find out the base score to be **9.3**


CVSS Version 2.0

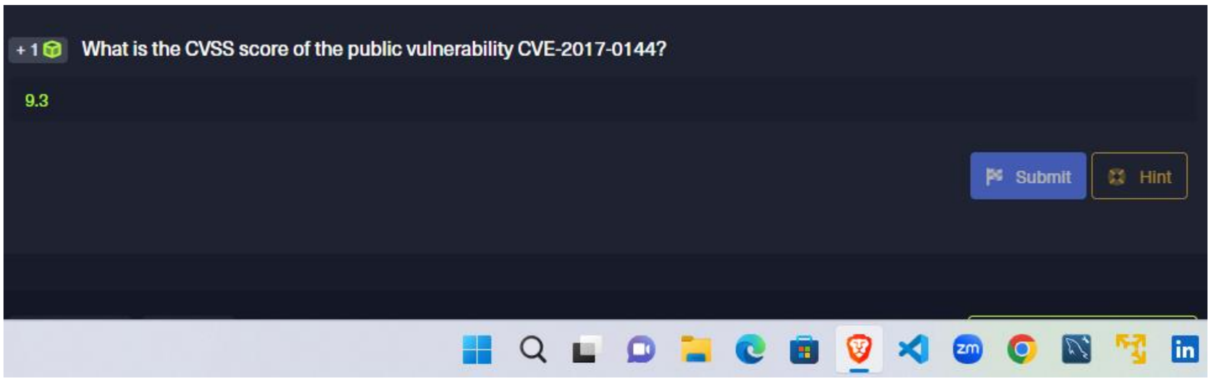
Common Vulnerability Scoring System Calculator CVE-2017-0144

Source: NIST

This page shows the components of a CVSS assessment and allows you to refine the resulting CVSS score with additional or different metric values. Please read the CVSS standards guide to fully understand how to assess vulnerabilities using CVSS and to interpret the resulting scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.

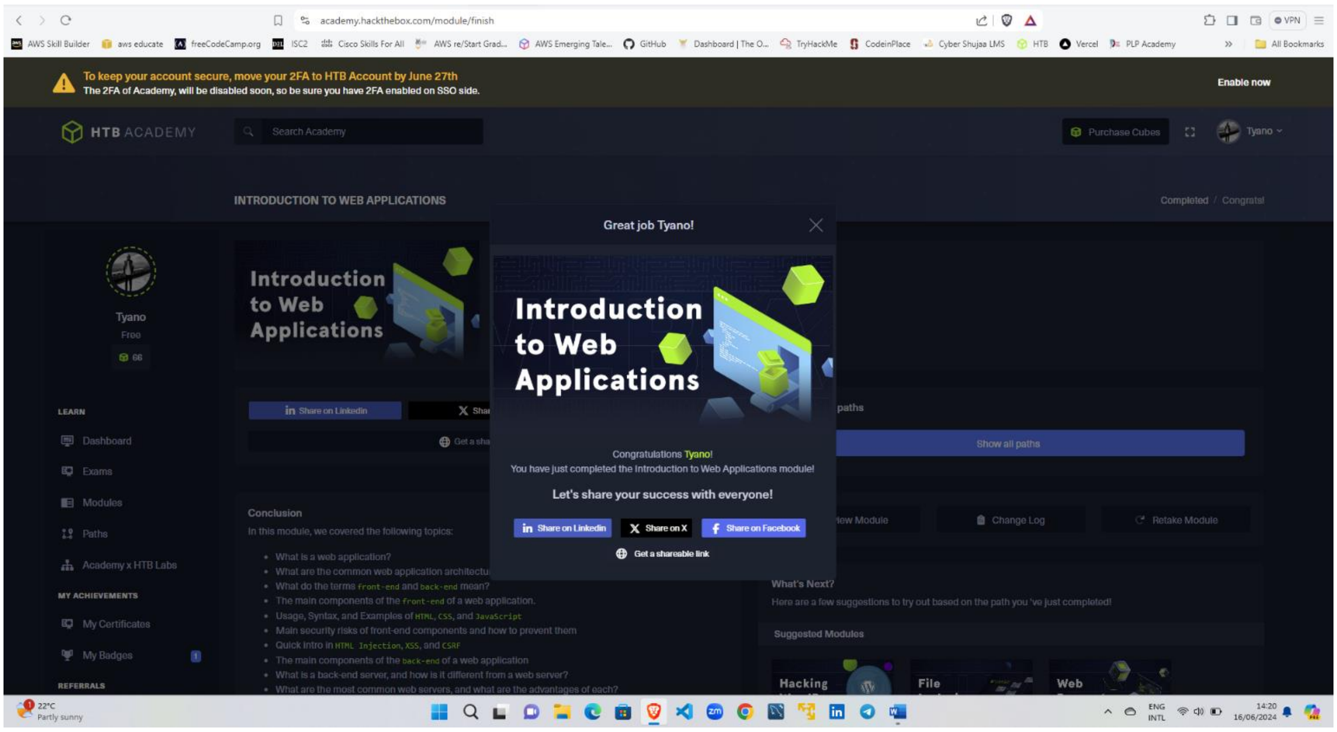
As of July 13th, 2022, the NVD no longer generates new information for CVSS v2.0. Existing CVSS v2.0 information will remain in the database but the NVD will no longer actively populate CVSS v2.0 for new CVEs. This change comes as CISA policies that rely on NVD data fully transition away from CVSS v2.0. NVD analysts will continue to use the reference information provided with the CVE and any publicly available information at the time of analysis to associate Reference Tags, CVSS v3.1, CWE, and CPE Applicability statements.

CVSS Base Score: 9.3
Impact Subscore: 10.0
Exploitability Subscore: 8.6
CVSS Temporal Score: NA
CVSS Environmental Score: NA
Modified Impact Subscore: NA
Overall CVSS Score: 9.3


+1 What is the CVSS score of the public vulnerability CVE-2017-0144?

9.3

Here is a screenshot to prove my completion of the module on Hack The Box and the link to the completed module https://academy.hackthebox.com/achievement/942061/75



To keep your account secure, move your 2FA to HTB Account by June 27th
The 2FA of Academy, will be disabled soon, so be sure you have 2FA enabled on SSO side.

INTRODUCTION TO WEB APPLICATIONS

Great job Tyano!

Congratulations Tyano!
You have just completed the Introduction to Web Applications module!

Let's share your success with everyone!

Conclusion
In this module, we covered the following topics:

- What is a web application?
- What are the common web application architectu
- What do the terms front-end and back-end mean?
- The main components of the front-end of a web application.
- Usage, Syntax, and Examples of HTML, CSS, and JavaScript
- Main security risks of front-end components and how to prevent them
- Quick intro in HTML Injection, XSS, and CSRF
- The main components of the back-end of a web application
- What is a back-end server, and how is it different from a web server?
- What are the most common web servers, and what are the advantages of each?

What's Next?
Here are a few suggestions to try out based on the path you 've just completed!

Suggested Modules

Conclusion

In this module, I explored the intricacies of web application architecture, delved into common security vulnerabilities, and developed strategies to mitigate these risks. The hands-on challenges not only reinforced my theoretical knowledge but also honed my practical skills, preparing me for real-world scenarios.

As web applications continue to evolve and become more sophisticated, the skills and insights I gained from this module will be invaluable in ensuring their security and integrity. The world of cybersecurity is dynamic and ever-changing; I am committed to building on this foundation, staying curious, and continuing to learn to stay ahead in this critical field.