

ASSIGNMENT 1: INTRODUCTION TO WEB APPLICATIONS

In this writeup, we will explore the steps involved in completing the module covering essential topics that form the foundation of understanding the structure and security of web applications to understand the most common web applications and their environment to proceed with web application attacks.

The module begins with defining web applications in which they interactive applications that run on web browsers. We then looked at its comparison to websites. The following are some of the major differences: Websites are static and cannot be changed in real-time while web apps are dynamic, interactive, responsive, and functional.

Attacking Web Applications

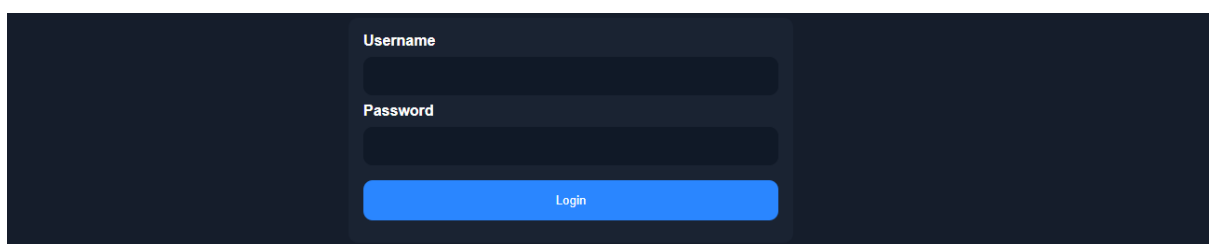
In this section, discussion of Front-End and Back-End components vulnerabilities takes place. To comprehend the web application's inner workings, we familiarized ourselves with the concepts of front-end and back-end, unraveling their respective components and their roles. The module extensively covered HTML, CSS, and JavaScript, explaining their usage, syntax, and providing practical examples. On the back-end side, we learned about the components comprising a web application's back-end and the distinction between a back-end server and a web server, various web servers and their advantages, along with different database types and their typical usage scenarios. Additionally, we touched upon popular web application development frameworks and the significance of APIs in modern web applications.

Front End Vulnerabilities

The following are some of the common front-end vulnerabilities:

Sensitive Data Exposure which refers to the availability of sensitive data in clear-text to the end-user. It can be achieved through viewing any website's page source in our browser by right-clicking anywhere on the page and selecting View Page, ctrl + u or view the page source through a web proxy such as Burp Suite.

The challenge:



By selecting view page on the above site provided we discovered an admin user and its corresponding password.

```

<label for="psw">Password</label>
<input type="password" required>
<!-- TODO: remove test credentials admin:HiddenInPlainSight -->
<button type="submit">Login</button>

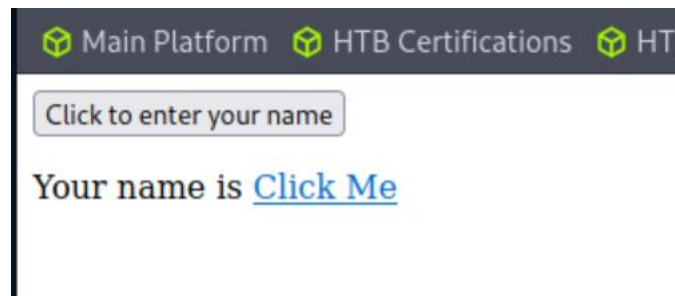
```

HTML Injection: occurs when unfiltered user input is displayed on the page. To test for HTML Injection, input a small amount of HTML code as our name and see if it is displayed as part of the page.

The challenge:

What text would be displayed on the page if we use the following payload as our input: `Click Me`

When user inputs the above it will showcase the username is Click Me.



Cross-Site Scripting (XSS): This is when there is an attack by injecting JavaScript code to be executed on the client-side. There are 3 types of XSS: Reflected XSS, Stored XSS and DOM XSS.

Type	Description
Reflected XSS	Occurs when user input is displayed on the page after processing (e.g., search result or error message).
Stored XSS	Occurs when user input is stored in the back end database and then displayed upon retrieval (e.g., posts or comments).
DOM XSS	Occurs when user input is directly shown in the browser and is written to an HTML DOM object (e.g., vulnerable username or page title).

The Challenge:

Try to use XSS to get the cookie value in the above page?

```
#"><img src=/ onerror=alert(document.cookie)>
```

Use the above code as an input to get the XSS of the cookie value.

165.232.42.79:32462 says

cookie=XSSisFun

OK

Cross-Site Request Forgery (CSRF): This is the type of attack where an attacker tricks a victim into unknowingly performing unwanted actions on a web application. By exploiting the trust relationship between the victim's browser and the target website, the attacker can execute malicious actions on behalf of the victim, potentially leading to unauthorized actions, data manipulation, or account compromise.

Preventing CSRF attacks involves:

Type	Description
Sanitization	Removing special characters and non-standard characters from user input before displaying it or storing it.
Validation	Ensuring that submitted user input matches the expected format (i.e., submitted email matched email format)

Backend Vulnerabilities

Back-end server is in the data access layer in the architecture overview. It consists of the web server which runs on port ports 80 or 443 and connects end-users to parts of web applications, databases in which are categorized into two relational (SQL) and non-relation (NoSQL) and development framework.

The following are some of the common Web vulnerabilities:

Broken Authentication/Access Control: refers to vulnerabilities that allow attackers to bypass authentication functions.

Malicious File Upload: refers to when web application has a file upload feature and does not properly validate the uploaded files, we may upload a malicious script (i.e., a PHP script), which will allow us to execute commands on the remote server.

Command Injection: Command Injection is a security vulnerability that occurs when an attacker can execute arbitrary system commands on a target server or application. It happens when untrusted user input is improperly handled and directly incorporated into system commands, allowing the attacker to execute malicious commands with the privileges of the application or server.

SQL Injection (SQLi): SQL Injection (SQLi) is a type of web application vulnerability where an attacker can manipulate SQL queries to execute unauthorized database operations. By injecting malicious SQL code into user input fields, the attacker can bypass input validation and gain unauthorized access to sensitive data or modify database contents.

Question:

To which of the above categories does public vulnerability 'CVE-2014-6271' belongs to? [Command Injection](#).

In conclusion, by completing this course module I got a deeper understanding of the structure and security of web applications as well as understanding the common web applications (frontend and backend) vulnerabilities such as Sensitive Data Exposure, Cross-Site Request Forgery (CSRF), Command Injection, and SQL Injection (SQLi), how to mitigate them and why they are crucial steps in safeguarding the integrity and confidentiality of sensitive information.

