

Task 3: Secure a Web Application

OBJECTIVE:

Task: Implement advanced security measures on a web application.

Details:

Apply security headers like Content Security Policy (CSP) and HTTP Strict Transport Security (HSTS). Conduct a thorough review of code for common vulnerabilities such as SQL injection or Cross-Site Scripting (XSS).

STEPS:

- **Deploy a Simple Website on GitHub**

1. **Create an HTML File:**

- Create a simple HTML file.

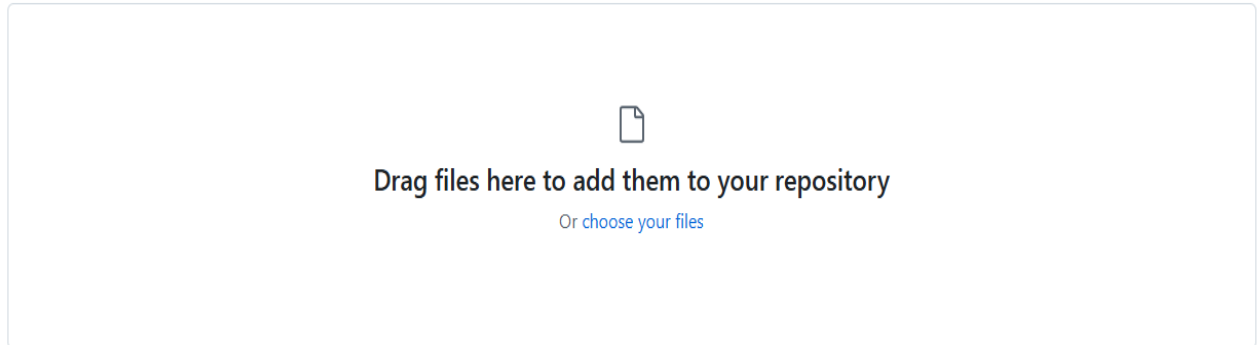
```
File Edit Format View Help
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Website</title>
</head>

<body>
  <h1>Hello, World!</h1>
  <p>Welcome to my website hosted on GitHub Pages!</p>
</body>
</html>
```

Upload the HTML File to GitHub:

- Go to your GitHub account and create a new repository (e.g., secure web app).
- Upload the index.html file to the repository.

YoungDevInterns_CyberSecurity_Tasks /



Commit changes

Add files via upload

Add an optional extended description...

- Enable GitHub Pages in the repository settings to host your website.

Farayha / YoungDevInterns_CyberSecurity_Tasks

Q Type to search

+ -

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

main

Go to file

- Basic Task 1 youngdev.docx
- Basic Task 2 YoungDev.docx
- Basic Task 3 YoungDev.docx
- expert task 1 youngdev.docx
- expert task 2 youngdev.docx
- index.html.txt**
- inter Task 1 youngdev.docx
- inter task 2 youngdev.docx
- inter task 3 youngdev.docx

YoungDevInterns_CyberSecurity_Tasks / index.html.txt

Farayha Add files via upload a59cb5f · now History

Code Blame 14 lines (13 loc) · 360 Bytes Code 55% faster with GitHub Copilot

Raw

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Website</title>
8 </head>
9
10 <body>
11   <h1>Hello, World!</h1>
12   <p>Welcome to my website hosted on GitHub Pages!</p>
13 </body>
14 </html>
```

➤ Apply Security Headers

2.1 Content Security Policy (CSP)

- **Overview:** CSP is a security feature that helps prevent various attacks, including Cross-Site Scripting (XSS) and data injection attacks.
- **Implementation:** To apply for CSP, add the following HTTP header to your web server configuration:

```
Content-Security-Policy: default-src 'self'; script-src 'self'; object-src 'none';
```

2.2 HTTP Strict Transport Security (HSTS)

- **Overview:** HSTS is a web security policy mechanism that helps protect websites against man-in-the-middle attacks such as protocol downgrades.
- **Implementation:** To enable HSTS, add the following header:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

The screenshot displays a GitHub repository interface for 'Farayha / YoungDevInterns_CyberSecurity_Tasks'. The 'Files' tab is active, showing a list of files on the left, including 'index.html.txt' which is selected. The main area shows the content of 'index.html.txt' with 14 lines of HTML code. The code includes a DOCTYPE declaration, HTML lang attribute, charset, X-UA-Compatible meta tag, viewport meta tag, and a title 'My Website'. The body contains a 'Hello, World!' heading and a paragraph 'Welcome to my website hosted on GitHub Pages!'. The interface also shows a search bar, navigation tabs (Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings), and a sidebar with file explorer and search options.

YoungDevInterns_CyberSecurity_Tasks / index.html.txt

Farayha Add files via upload a59cb5f · now History

Code Blame 14 lines (13 loc) · 360 Bytes Code 55% faster with GitHub Copilot Raw Copy Download Edit

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Website</title>
8 </head>
9
10 <body>
11   <h1>Hello, World!</h1>
12   <p>Welcome to my website hosted on GitHub Pages!</p>
13 </body>
14 </html>
```

➤ **Conduct a Thorough Review of the Code for Common Vulnerabilities**

3.1 Review for SQL Injection Vulnerabilities

1. Understand SQL Injection:

- SQL injection occurs when an attacker manipulates SQL queries by injecting malicious SQL code into input fields.

2. Identify User Input Points:

- Look for places in the code where user input is processed (e.g., login forms, search bars).

3. Check for Dynamic SQL Queries:

- Review the code for any dynamic SQL queries that concatenate user input directly into the query string.

4. Mitigation Techniques:

- **Use Prepared Statements:** Ensure the application uses prepared statements or parameterized queries to prevent SQL injection.
- **Input Validation:** Validate user input by using whitelisting techniques to accept only expected characters.

3.2 Review for Cross-Site Scripting (XSS) Vulnerabilities

1. Understand XSS:

- XSS vulnerabilities occur when an attacker injects malicious scripts into web pages viewed by other users.

2. Identify Output Points:

- Review the code for areas where user input is rendered on web pages.

3. Check for Unsafe Output Encoding:

- Look for code that outputs user input without proper encoding.

4. Mitigation Techniques:

- **Output Encoding:** Ensure that user input is properly encoded before rendering.
- **Use CSP:** Implement a Content Security Policy to restrict which scripts can be executed on your web pages.

➤ **Code Review Checklist**

- **For SQL Injection:**
 - Are all user inputs validated?
 - Are prepared statements or parameterized queries used for database interactions?
 - Are database queries logged and monitored for suspicious activities?
- **For XSS:**
 - Is user-generated content properly encoded before being displayed on the page?
 - Are there any functions that directly manipulate the DOM without encoding?
 - Is a Content Security Policy (CSP) in place to limit the execution of scripts?

Conclusion

Implementing advanced security measures such as CSP and HSTS, along with conducting a thorough code review for common vulnerabilities, is essential for securing web applications. By following best practices for input validation, output encoding, and using prepared statements, developers can significantly reduce the risk of exploitation. Regular security assessments should be an integral part of the development process.