LAB EXERCISE

8

COMP3134

Introduction to Cyber Security

Week: 12

Objective(s):
Attacking the web applications and applying mitigation techniques

Learning Outcome(s):
Implement various tactics to attack a network or application
Critique and execute mitigation techniques

Table of Contents

Contents

Summary

Goal: Attacking the web applications and applying mitigation techniques

In Effort To: Implement various tactics to attack a network or application, as well as to critique and execute mitigation techniques

A. Clone GitHub Repo

Clone course GitHub repo to any location on your local machine
Navigate to the location above and create a folder named **wk12**
Use this local folder created above to create all the files necessary for this Lab Exercise

B. Connect to Droplet Server

Using GitBash (on Windows) or the Terminal (on Mac), connect to your droplet server by executing the following command

```
ssh droplet_username@droplet_ip_address
```

When prompted, user your droplet password

Navigate to your web root of your droplet

```
cd /var/www/html
```

You will create scripts in the steps below in the web root

C. Cross-Site Request Forgery (CSFR)

Cross-site request forgery is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the web application trusts.

Let's take the steps to set-up a CSFR scenario:

1) Create page **csfr.php**
2) Create a form with
    a. username and password fields
    b. submit button
    c. div splash message (only displays once form submission occurs)
    d. Using the POST method

3) Once user submits, output success or failure message (to the div splash message)
   Depending on whether user enters credentials:
   username = host
   password = pass

4) Create page **csfr.html** that forges form submission
    a. Using JavaScript, once the user opens the page, a form submission is automatically sent to csfr.php page with a valid username/password combo

D. Reflected XSS

Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data.

Let's take the steps to set-up a Reflected XSS scenario:

1) Create a page **reflectedxss.html** that has the following elements
    a. An H1 header with a page title
    b. A form with
        i. 1 text field & 1 submit button
    c. A div element that will display output

2) Using JS, disable default form behavior
3) Once user submits, the div element should display/execute the value of the text field.
4) Use the form on the page and input your first name
    a. Create a screenshot named **reflectedxss_good.png**
5) Use the form on the page and input JavaScript code. Be as creative as you like
    a. Create a screenshot named **reflectedxss_bad.png**

E. Stored XSS

Stored XSS occurs when the user input is ran immediately from a storage source (such as file storage or database storage) without that data being made safe to render in the browser

Let's take the steps to set-up a Reflected XSS scenario:

1) Create a file named **storedxss.txt**
    a. Store the following code in the text file
        i. <script>document.location.href="/directory_traversal_part1.php"</script>

2) Create a file named **storexss.php**
    a. In this file, read and output directly the contents of the text file without sanitizing

3) In a file named **storedxss_issue.txt**, outline what issue(s) you see?

F. Mitigating Cross-Site Forgery Request

In order to mitigate CSFR, we need to ensure that the requests are coming from valid sessions

1) Create a file named **csfr_action.php**. Copy the contents of **csfr.php** as your starting point.
    a. GET a session variable named confirmation
    b. Retrieve a post value named confirmation
    c. Ensure the session variable and confirmation value are equal (have same value)

2) Create a file named **csfr_form.php**. Copy the contents of **csfr.html** as your starting point.
    a. Create a PHP session variable called confirmation.
        i. Give it a random value.
    b. Send session variable value as a hidden form field value named confirmation.

3) Use form of PHP page: **csfr_form.php**

4) Use form HTML page **csfr.html**.

5) In a file named **csfr_tests.txt**, outline what you noticed.

G. Commit and Upload Changes to GitHub repo

Commit the changes to your repo by:

1. Opening a GitBash window and ensure that it is connected to your local machine
2. Navigate to local repository directory location
3. Add all the files completed in this Lab Exercise
4. Commit the changes
5. Push the changes to your GitHub course repo

Please refer to the instructions in the last section of Lab Exercise 1