## Lab 9: Conducting a Cross Site Scripting (XSS) Attack

### Scenario

CyberSecure Labs, a cybersecurity consulting firm, has been engaged by TechBlog Solutions, an online publishing platform, to conduct a comprehensive security assessment of its web applications. The primary objective is to identify vulnerabilities that could allow attackers to compromise user accounts or manipulate website content. During the assessment, a high-risk concern is raised regarding the blog comment functionality, which stores user-submitted content in a backend database and displays it to all visitors. Since the application does not appear to have proper input sanitization or output encoding, there is a risk of a Persistent Cross Site Scripting (XSS) attack. This type of vulnerability could allow an attacker to inject malicious JavaScript code into the blog comments, which would be executed automatically whenever any user views the infected page, potentially leading to credential theft, session hijacking, or the installation of malicious scripts on the victim’s browser.

### Solution

As a certified security practitioner, your task is to simulate a persistent XSS attack on the blog comment system to verify the vulnerability. You can test this by submitting a script payload, such as <script>alert('XSS Test')</script>, into the comment section. If the payload executes upon viewing the page, it confirms that the application is not properly sanitizing inputs or encoding outputs. To mitigate this risk, recommend the implementation of secure coding practices, including validating and sanitizing all user input, applying proper output encoding, and restricting script execution through a Content Security Policy (CSP). These measures will help ensure that malicious scripts cannot be stored or executed within the application, thereby safeguarding user data and maintaining the integrity of the platform. In this lab, we will be using Mutillidae to demonstrate an XSS vulnerability.

|  |
| --- |
| 1. Turn on **ParrotOS** virtual machine. Open a **Terminal**, and execute the **sudo su** command to run programs with root privileges.    2. Execute the following command: **https://github.com/webpwnized/mutillidae-dockerhub.git** to clone the Mutillidae DockerHub GitHub repository.    3. Execute the **ls** command to list all the contents inside a directory. Then execute **cd** **mutillidae-dockerhub** to go inside a directory.    4. Execute the following command: **docker-compose up** to build, create, start, and attach to containers for a service or a set of services defined in a docker-compose.yml file.    5. Mutillidae Docker container is successfully created. It takes a few minutes to complete a process.    6. Open **Firebox** browser in ParrotOS. In the search bar, type **127.0.0.1** localhost IP address and then press **Enter** to access the Mutillidae web GUI.    7. The database page appears; you need to setup a database for Multillidae. Click on the **Click here** link to reset the database.    8. A pop-up appears: **Database reset successfully**. Click on the **OK** button to go to the Mutillidae GUI webpage.    9. Now we need to create a dummy user. Click on the **Login/Register** to register a new user account.    10. Click on the **Please register here** link.    11. In the registration form page, enter your information: **Username**, **Password**, **Confirm Password**, **First Name**, **Last Name,** and **Signature**. After that, click on the **Create Account** button.    12. The account is successfully created in Multillidae. Click on the **Login/Register** to login to a new user account.    13. Enter your **Username** and **Password** at the time of created your account. Click on the **Login** button.    14. We will test a Cross Site Scripting (XSS) attack on a blog website to access it click on the **OWASP 2017** > **A7 - Cross Site Scripting (XSS)** > **Persistent (Second Order)** > **Add to your blog**.    15. Upon accessing this page, you will find a text box that allows you to submit content as a new entry to your blog. The objective in this scenario is to insert a script into the blog that will trigger a pop-up on the user’s screen each time the page is loaded. To assess whether the page is susceptible to a Persistent XSS vulnerability, input the following script **<script>alert("Hello, you have been hacked!")</script>** into the textbox. Then click on the **Save Blog Entry** button.    16. You will notice that a pop-up will appear on your screen with the message we entered into the textbox. Click on the **OK** button to close the pop-up.    17. We have executed an XSS attack, but we want to make sure that this attack will occur every time the page is loaded. Navigate to **OWASP 2017** > **A7 - Cross Site Scripting (XSS)** > **Persistent (Second Order)** > **View someone’s blog**.    18. Click on the **Please Choose Author** dropdown and then select **Show All**. Click on the **View Blog Entries** button.    19. You will observe that the pop-up reappears with the same message previously entered. This demonstrates that any user accessing this page will encounter the same pop-up, confirming the successful execution of a persistent XSS attack. |