**Lab7 – Cross-Site Request Forgery (CSRF) attack lab**

Due: Wednesday, Nov 24, 5:00pm

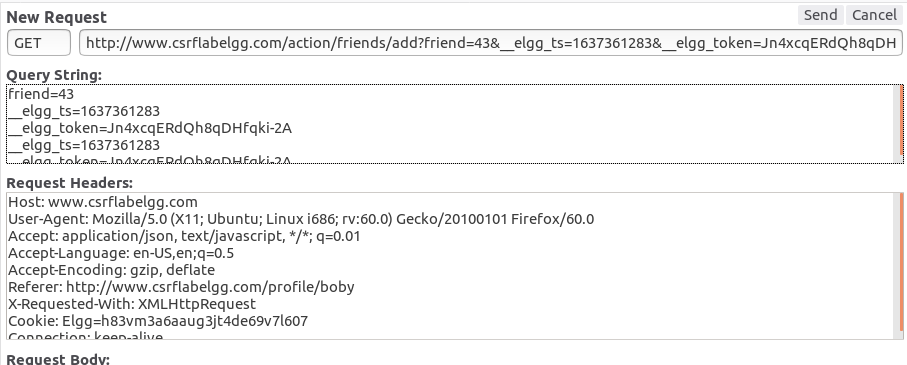
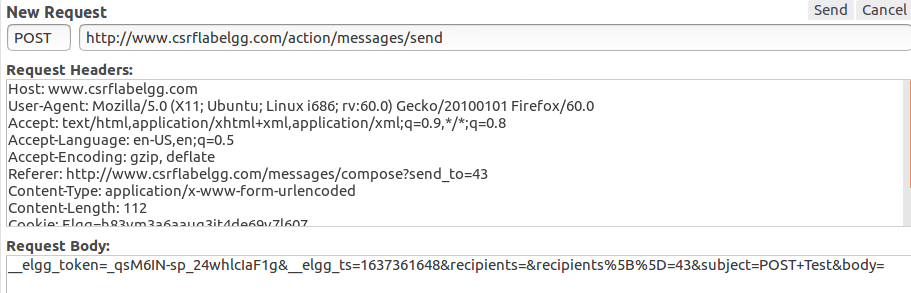
Turn in: lab report

Points: 60 pts

Note: In this lab, the countermeasure to defend against CSRF attack has been disabled. Without using the the \_\_elgg\_ts and \_\_elgg\_token, we can launch the attack easily.

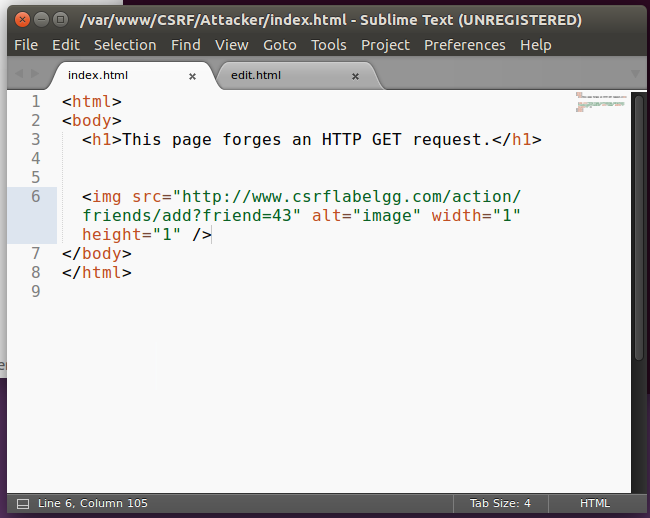
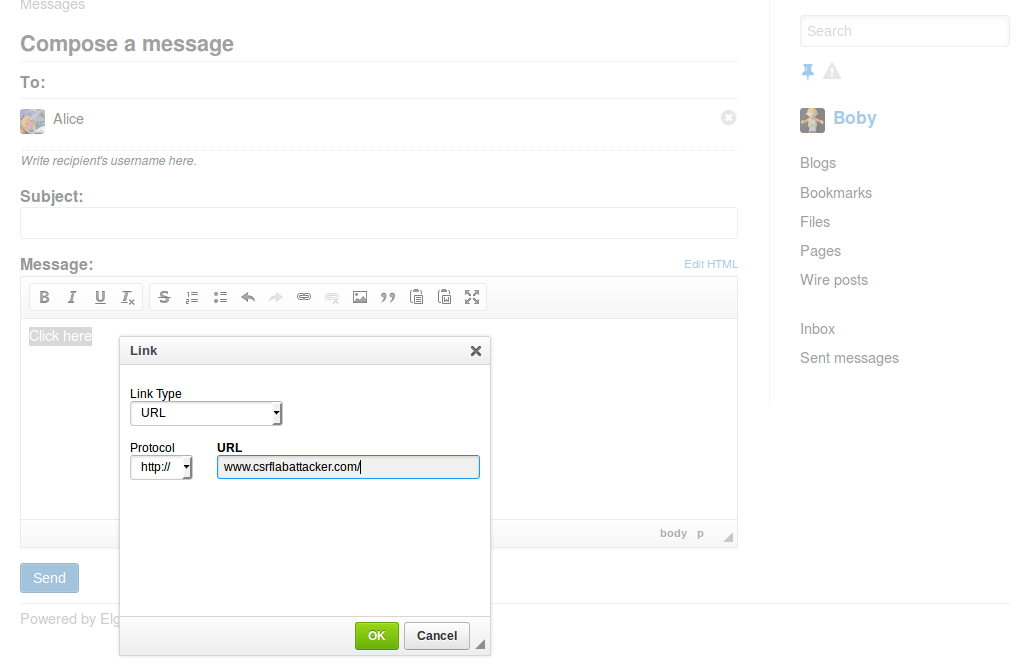
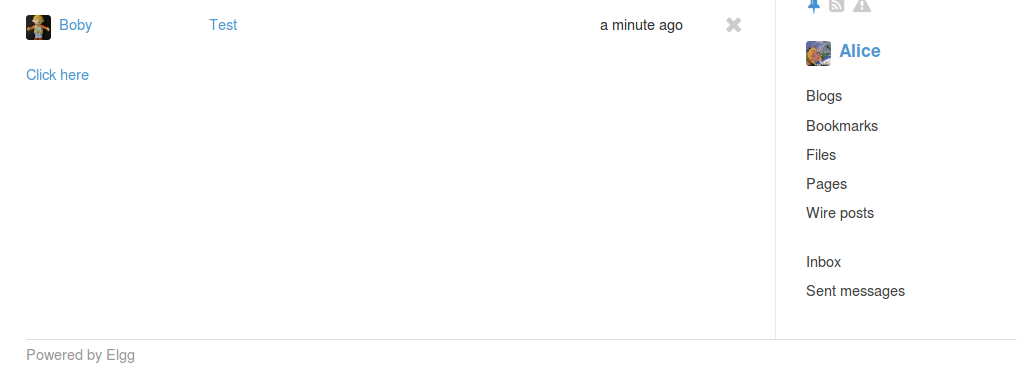
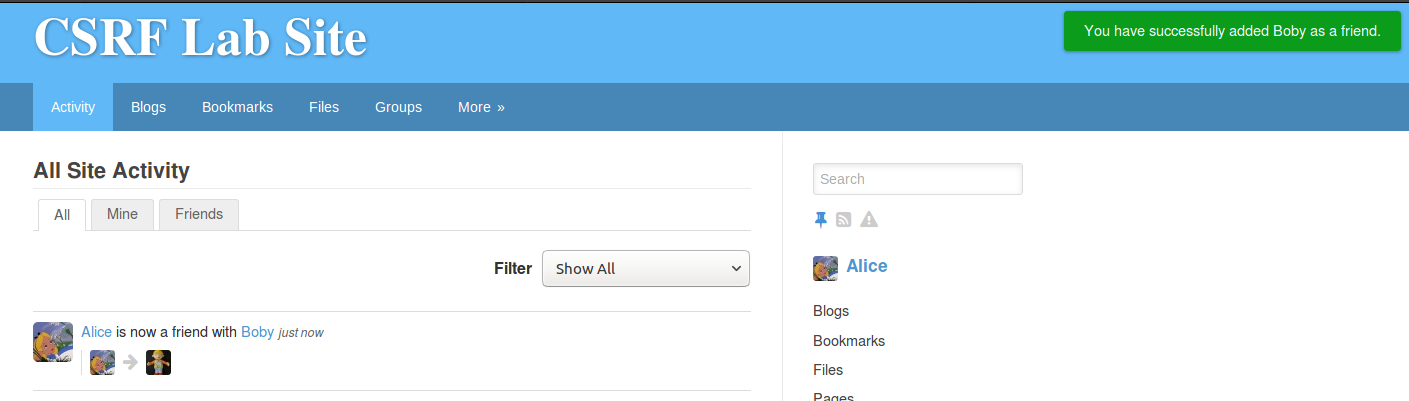
(10 pts) Task 1: Observing HTTP Request

What does a legitimate HTTP request look like?

* Take a screenshot of the HTTP header analyzer to show a HTTP GET request and its content.
* Here’s a GET request, the Request Body is cut off but it is empty:
* 
* Take a screenshot of the HTTP header analyzer to show a HTTP POST request and its content.
* Here is a POST request, there is no Query String; instead the data is in the Request Body:
* 

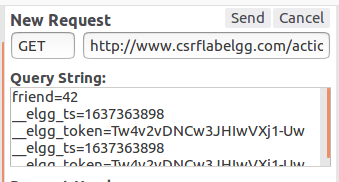
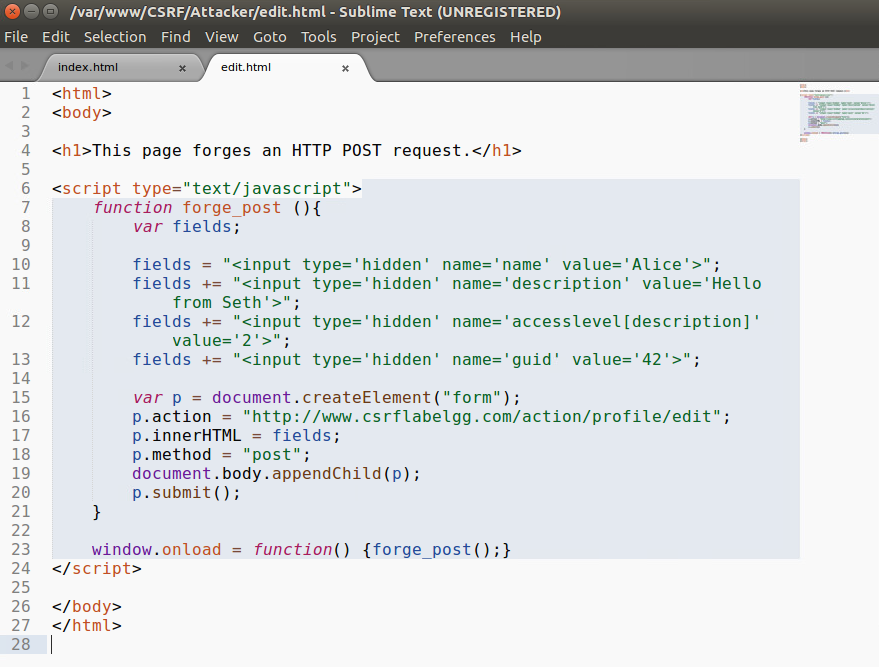
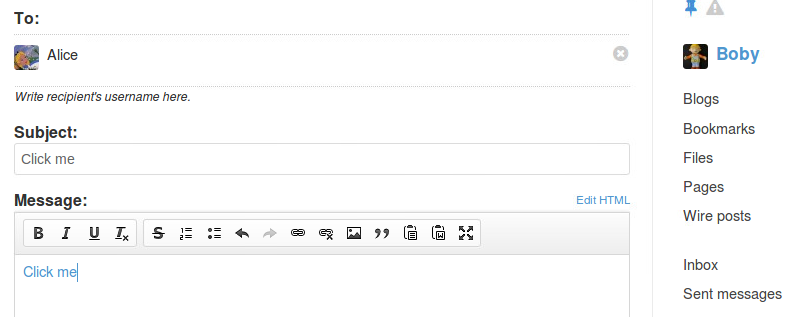
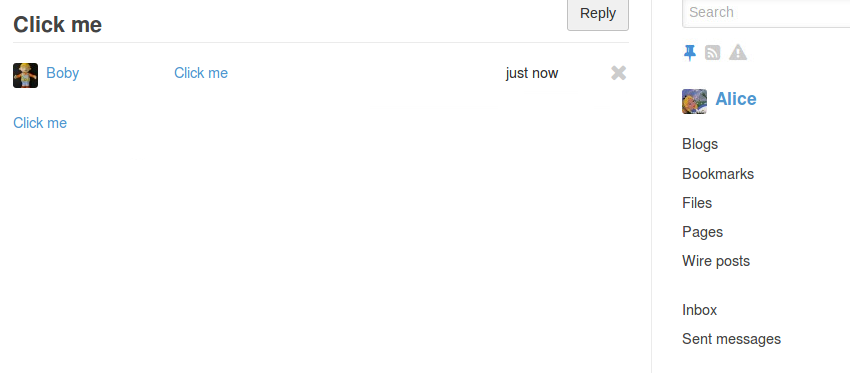
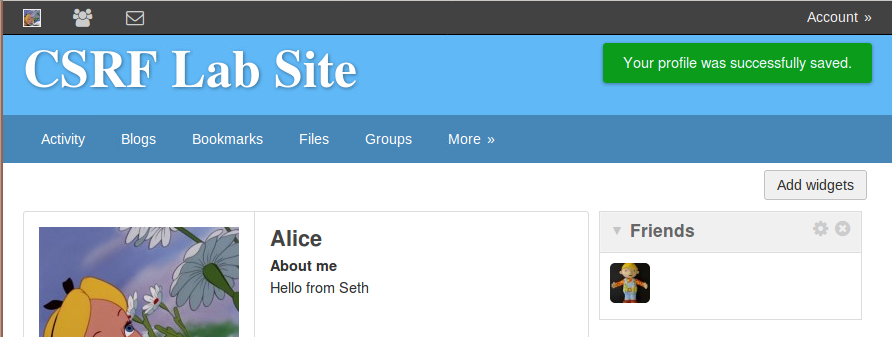
(15 pts) Task 2: CSRF Attack using GET request

Let’s pick Boby to be the bad guy who wants Alice to add him as a friend. So after this attack, Alice added Boby as a friend after clicking on the bad link sent from Boby. Sample code is on slide page 12.

* Show me how you got the correct friend id.
  + I logged into Charlie’s account and added Boby
  + 
  + The friend ID is 43
* Take a screenshot of the content of your index.html.
  + 
* Did you/Boby send the bad link to Alice? have a screenshot.
* 
* Compose the message with the malicious link
* 
* Open the message as Alice
* 
* 
* Success! Alice has ‘friended’ Boby.
* Explain how the CSRF attack works in this task.
  + Boby finds the GET request that adds himself as a friend.
  + He puts the link in the src of an img, so that the browser will send a GET request to the src. Since the link is to the csrflabelgg.com site, the browser also sends the associated cookies from that site, since Alice is logged in when she clicks it.

(25 pts)Task 3: CSRF Attack using POST Request

Boby wants to change Alice’s profile page to display a message 🡨 pick your own message, include your name in it. Sample code can be found on Slide page 19.

* Show me how you got the correct friend id.
* 
* Take a screenshot of the content of your html.
* 
* Did you/Boby send the bad link to Alice? have a screenshot.
* 
* Boby sends the malicious link to Alice
* 
* Alice clicks the link
* Take a screenshot to show Alice’s profile page was updated successfully.
* 
* Success!

(10 pts) Conclusion:

1. What is a CSFR attack? What is/are the reasons causing the problem?
   1. A Cross-Site Request Forgery attack can occur when a victim has an active session on the site you want to attack. It is called Cross-Site because the attacking site sends a request to the victim site. Since the person is logged in, the browser sends their session information along the attack code, so the website *thinks* the request came from them.
2. What are the countermeasures we can use to defeat the attack? (You should continue reading the later texts on the handout and slides to understand the countermeasures.)
   1. There’s a few options:
      1. Put a HTTP header field that identifies the address of the webpage where the request is generated
         1. Poses browsing history concerns
      2. Same Site Cookies
         1. Some browsers allow special types of cookies to be set that tells the browser whether the cookies should be attached to cross-site requests
      3. Secret Token
         1. Embed a random secret value inside each page
         2. Include that secret value in requests
         3. Pages from a different origin won’t be able to access the secret value. The value is randomly generated and is different for different users.
3. What have you learned? Anything fun or not fun?
   1. I generally avoid clicking links from any source I wasn’t expecting a link from, and this lab has reinforced that. This seemed like an easier way to accomplish some of the attacks we did in the last XSS lab. I’ve enjoyed the website attacks; I’ve worked on HTML/CSS/JS projects before so it’s nice to work on something I’m familiar with.