

CSCI 5234 Web Security

Lab1

Cross Site Request Forgery (CSRF) Attacks

Configure the Virtual Machines:

1. Follow the instructions given in the [Lab Setup](#) page to download and install the virtual machines (VMs).
2. Configure the Virtual Machines after having installed the VMs:
NOTE: A VM must be off in order to be configured.
Click and open the Settings (Figure 1).

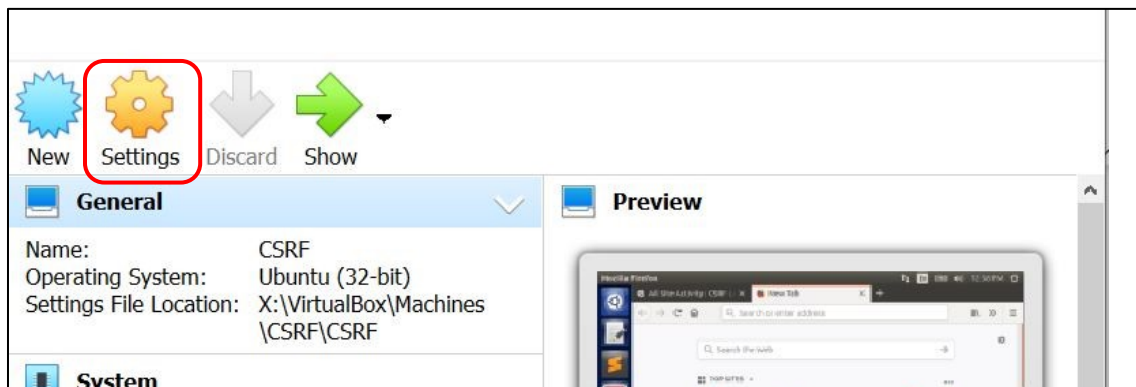


Figure 1: Configuring the VMs

3. Open the settings for the Server virtual machine.
 - a. In the General:
Advanced tab, set Shared Clipboard and Drag'n'Drop to Bidirectional.
 - b. In the Network:
Adapter 1 tab, set the "Attached to" field to Bridged Adapter. Click the arrow to show advanced settings and change "Promiscuous Mode" to Allow VMs.
4. Right click on the Server VM and select Clone. (Note: the VM must be Powered Off). Name the new virtual machine "Attacker", select "Reinitialize the MAC address", and create a Full Clone.
5. Lab environment setup:
In the Victim VM, modify the /etc/hosts file to map the domain name of www.csrfattack.com and www.csrflabelgg.com to the attacker machine's IP address. (modify 192.168.0.165 to the attacker machine's IP address)

```
192.168.0.165 www.csrflabelgg.com
192.168.0.165 www.csrfattack.com
```
6. Apache configuration: Restart apache

Task 1: Observing HTTP Request in Victim VM

1. Open the Firefox and click the button on the right corner and click add-ons. On the search bar, enter "HTTP Header Live" (Figure 2).

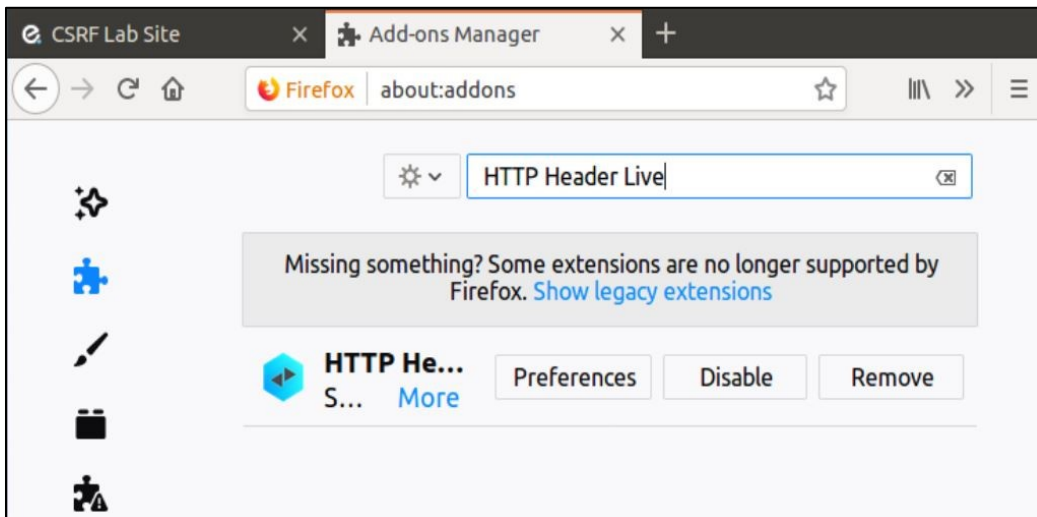


Figure 2: Configuring the Firefox

2. Click HTTP Header Live to add it to the Firefox. HTTP Header Live displays the HTTP header and allows the user to edit the header and send it (Figure 3).

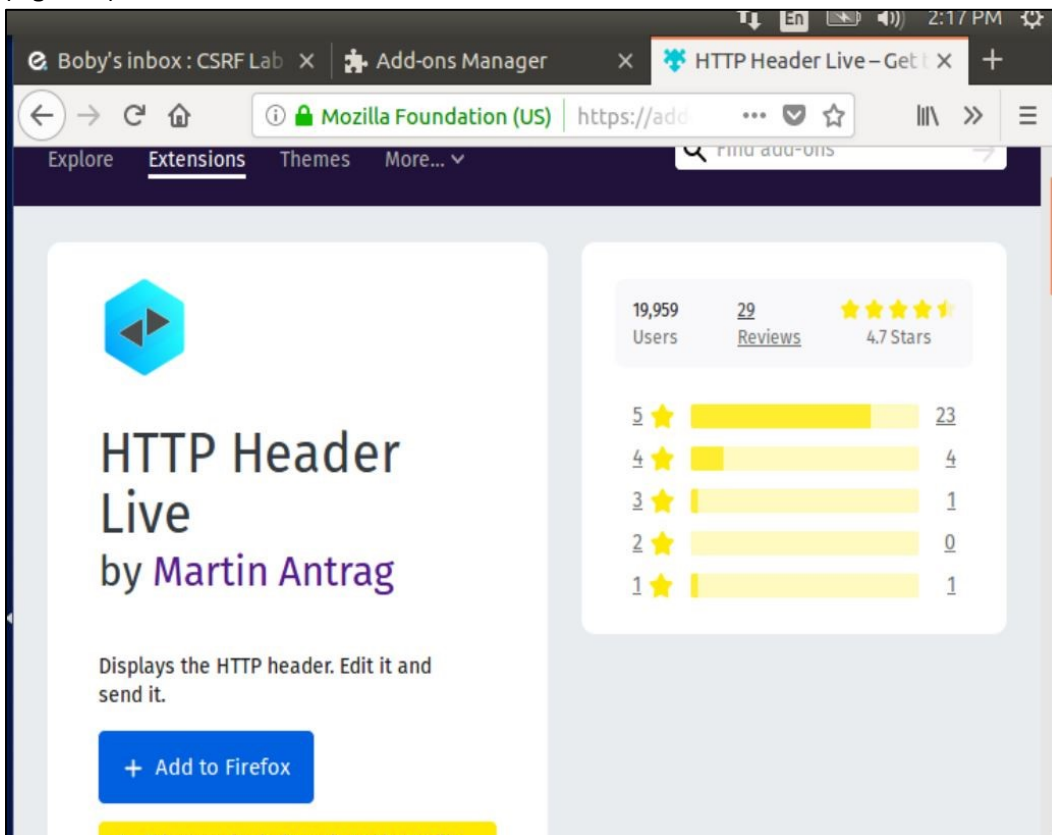


Figure 3: HTTP Header Live in Firefox

3. In the Firefox, observe <http://www.csrflabelgg.com/> (Figure 4).



Figure 4: Observing the HTTP traffic in the Firefox

4. Open a browser window and login as Alice on the csrflabelgg.com page (Figure 5). You will be able to view the report produced by the HTTP Header Live page about that login session (Figure 6).

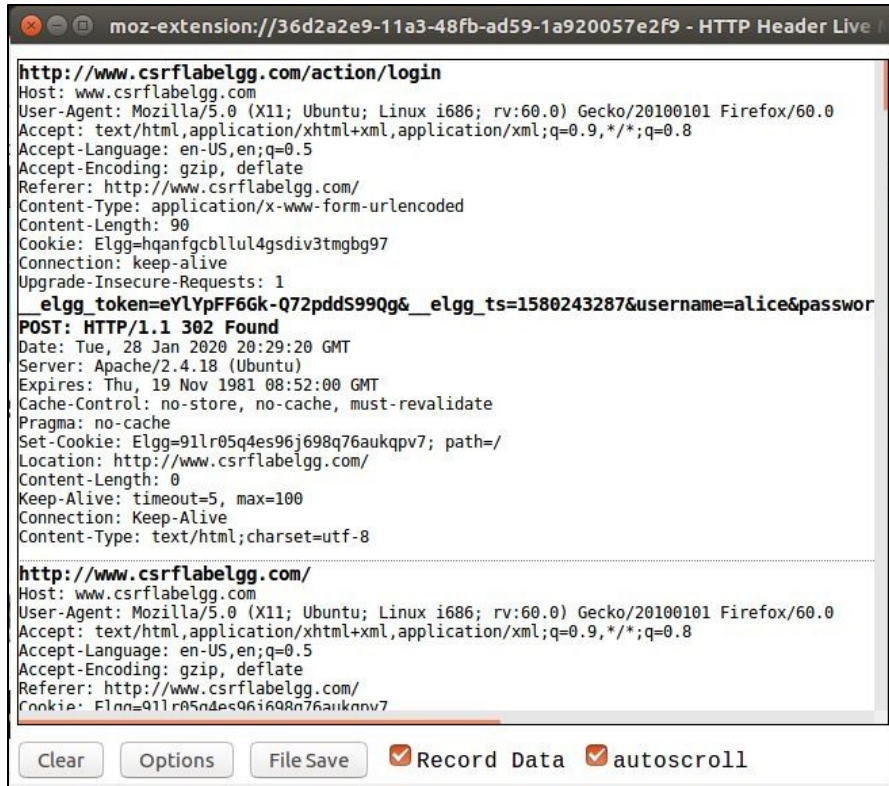


Figure 5. Observing the login session using HTTP Header Live

You should be able to see the username and password captured by the HTTP Header Live (Figure 6).

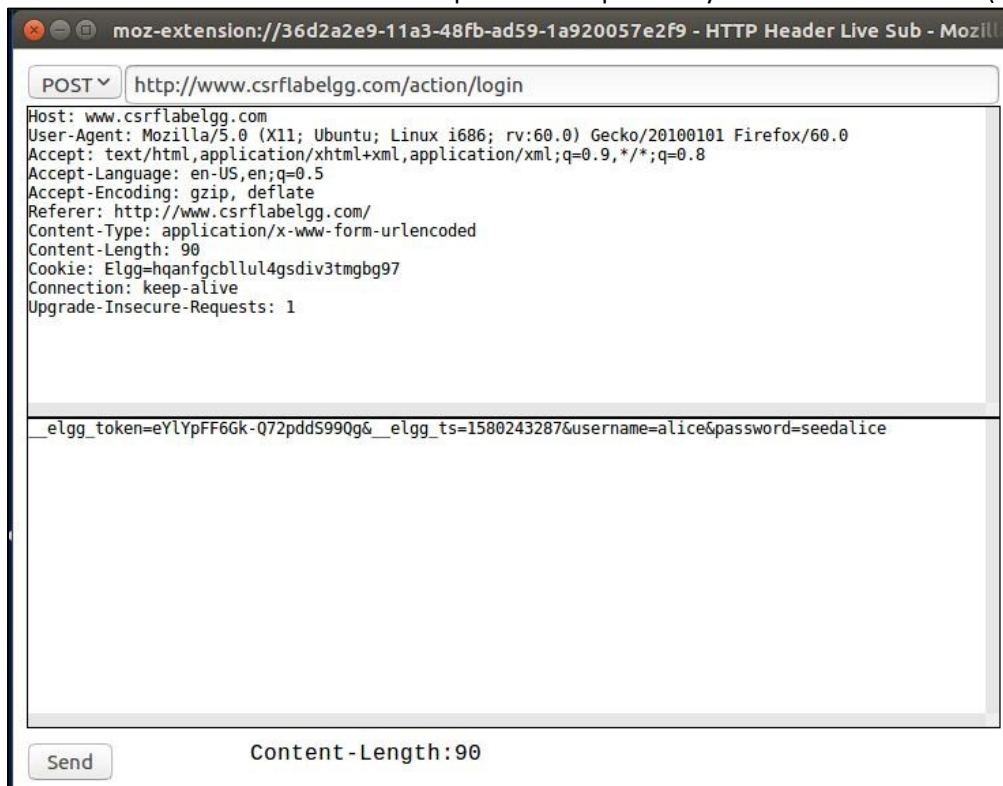


Figure 6. username and password captured by the HTTP Header Live

Task 2: CSRF Attack using GET Request

In this task, we need two people in the Elgg social network: Alice and Boby. You can use only one VM login and out or clone the other VM as Boby.

1. On the attacker VM, open the Firefox and login to www.csrflabelgg.com as Boby and add Alice to friend (Figure 7).

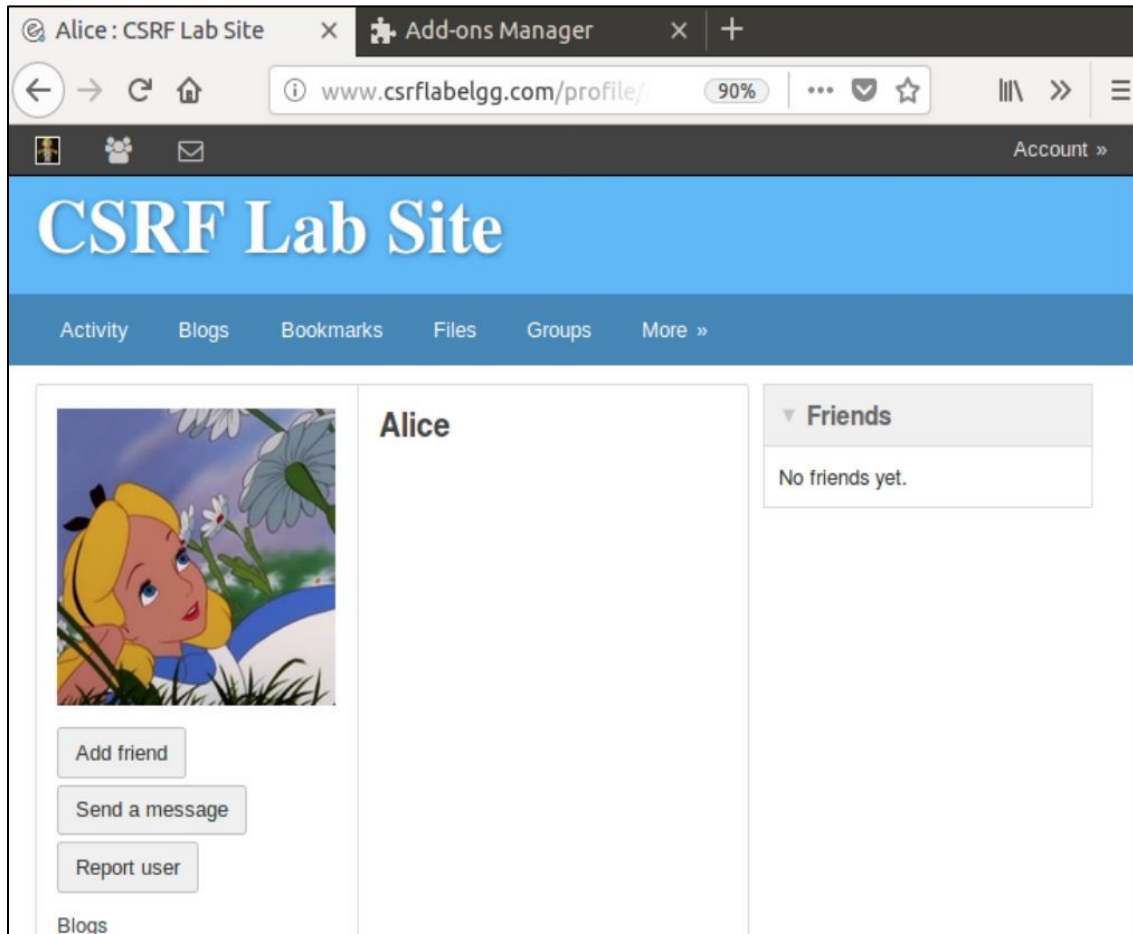


Figure 7: CSRF Lab Site

2. Observing HTTP request (Figure 8)

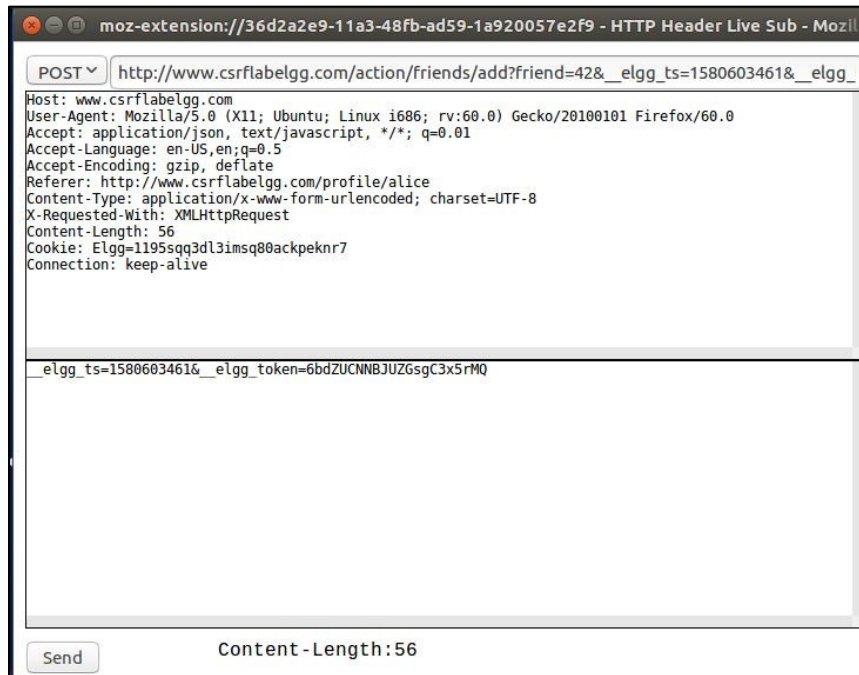


Figure 8: HTTP POST request

3. Now, login as Alice to www.csrflabelgg.com on the other VM, we can see that Bobby is not on Alice's friend list (Figure 9).

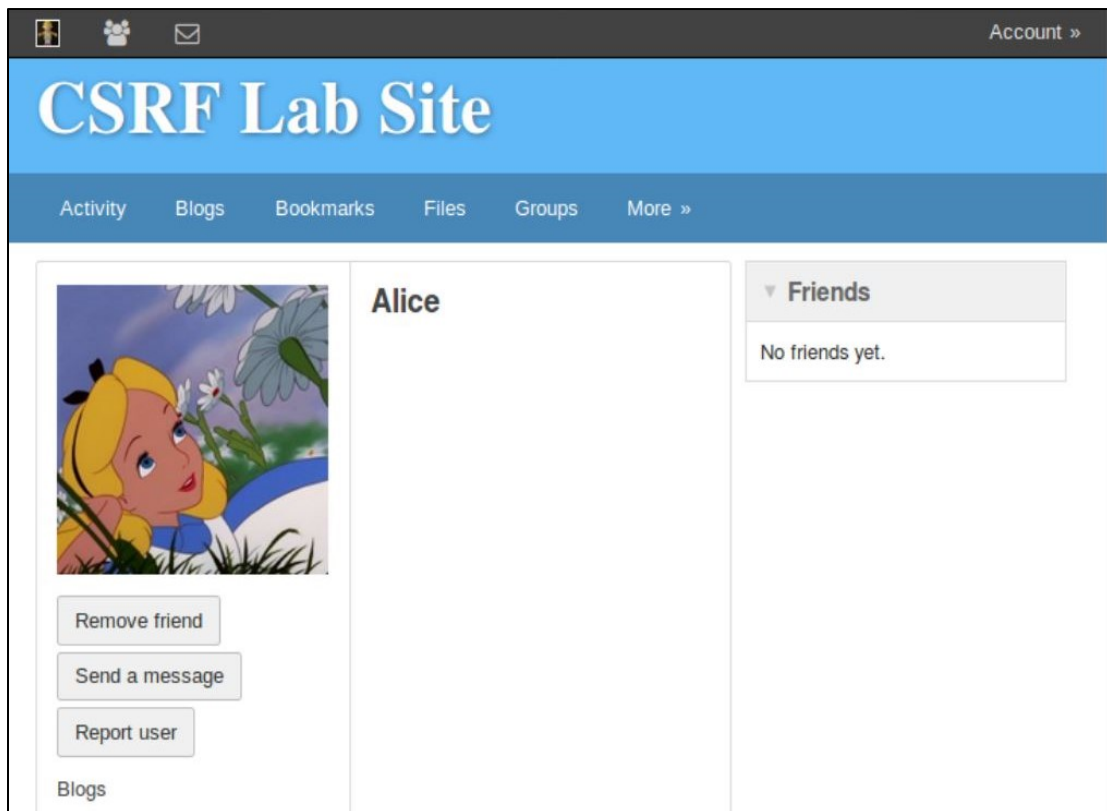
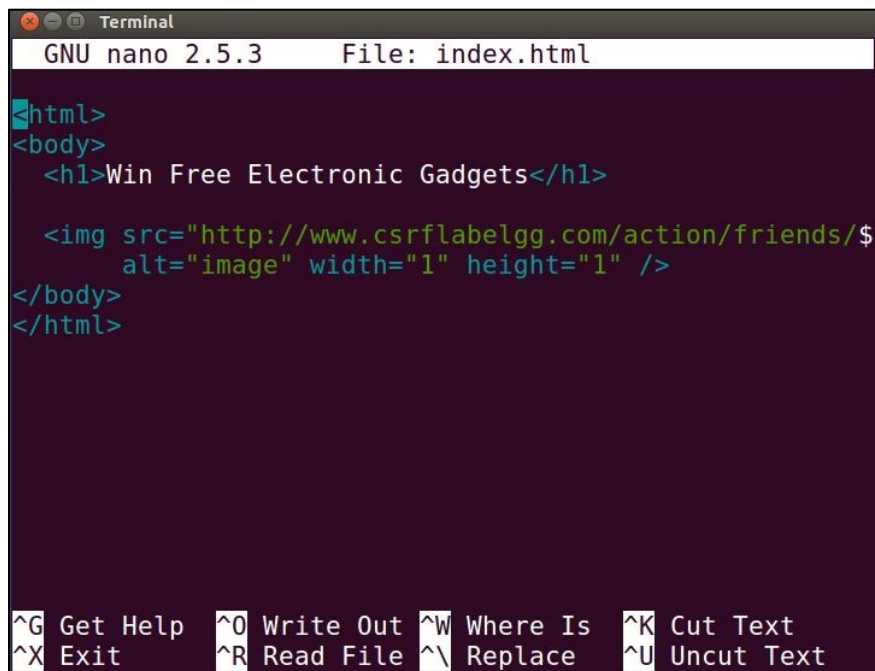


Figure 9: Bobby on Alice's profile

4. In order to add Alice to Bobby's friend list automatically, we need to prepare an attractive malicious webpage. Create a file "index.html" in /var/www/CSRF/attacker and make a webpage as shown below (Figure 10).

```
<html>
<body>
  <h1>Win Free Electronic Gadgets</h1>
  
</body>
</html>
```



```
Terminal
GNU nano 2.5.3 File: index.html

<html>
<body>
  <h1>Win Free Electronic Gadgets</h1>
  
</body>
</html>

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text
```

Figure 10: index.html

5. Now, Bobby sent a message to Alice with the malicious webpage's URL to attract Alice (Figure 11).

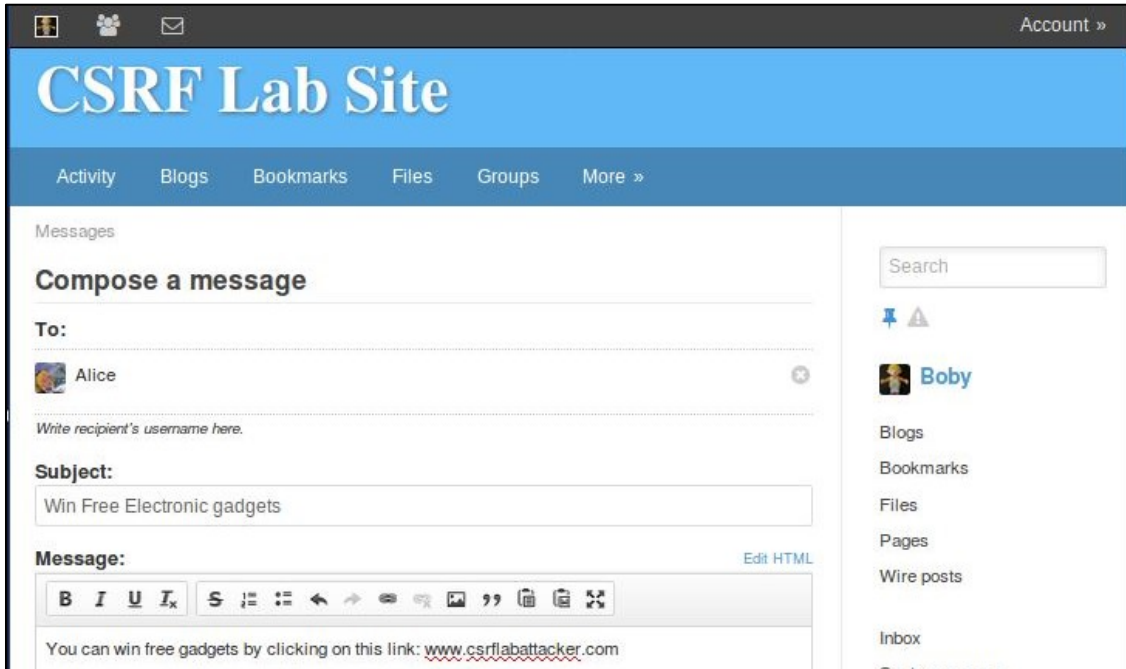


Figure 11: Bobby sends a message to Alice

6. When Alice sees the message, she curies and visits the URL given in the message (Figure 12).

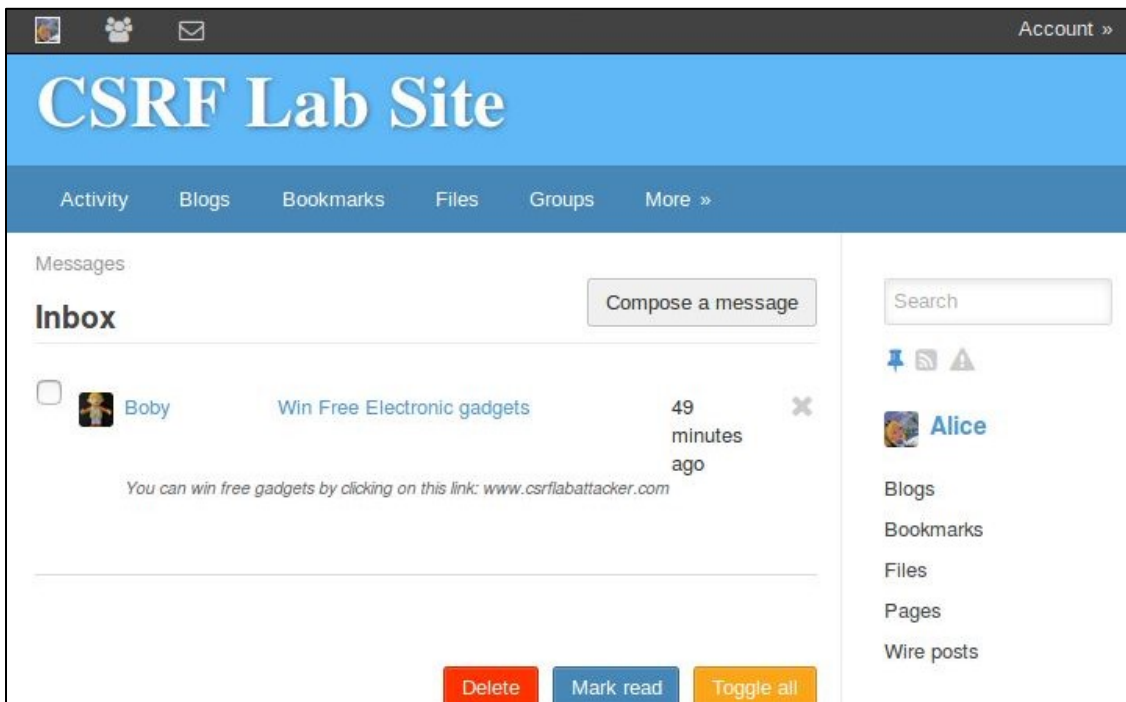


Figure 12: Alice's inbox

7. Once Alice clicks on the URL, she is redirected to the web page we created on step 4 (Figure 13).



Figure 13: www.csrfbattacker.com

8. After redirected to the www.csrfbattacker.com, Alice added Bobby automatically (Figure 14).

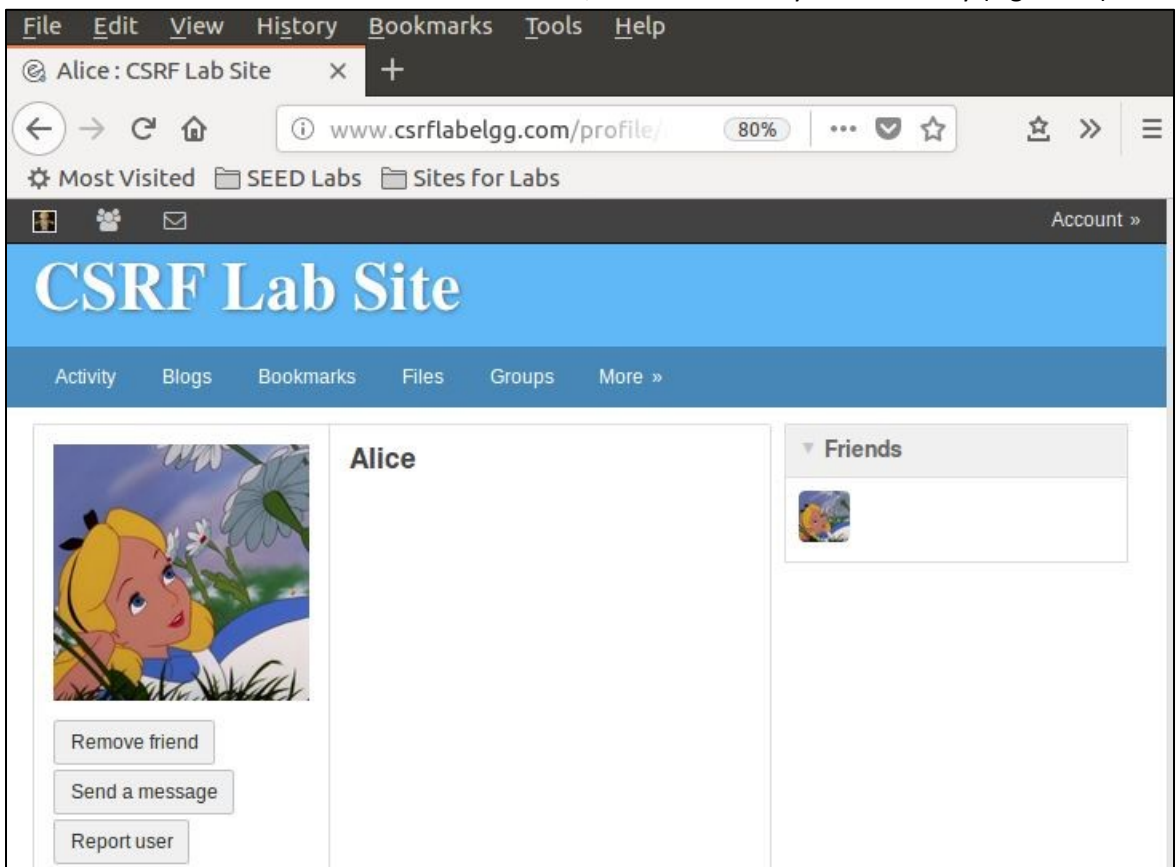


Figure 14: Alice redirected to CSRF Lab Site

9. When we observe HTTP request, we can see the HTTP request referrer to www.csrfbattacker.com (Figure 15).

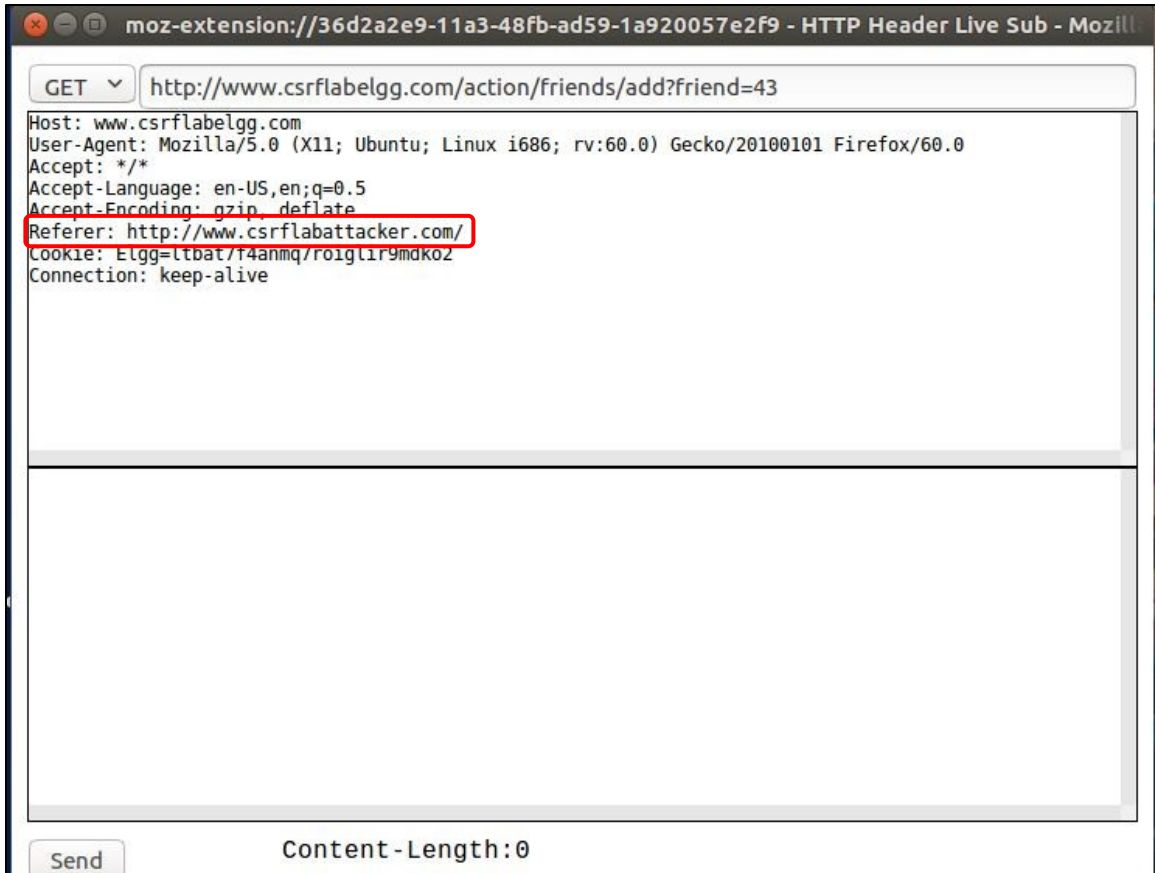


Figure 15: HTTP GET request

Task 3: CSRF Attack using POST Request

1. Boby sends the other message to Alice to change Alice's profile (Figure 16).

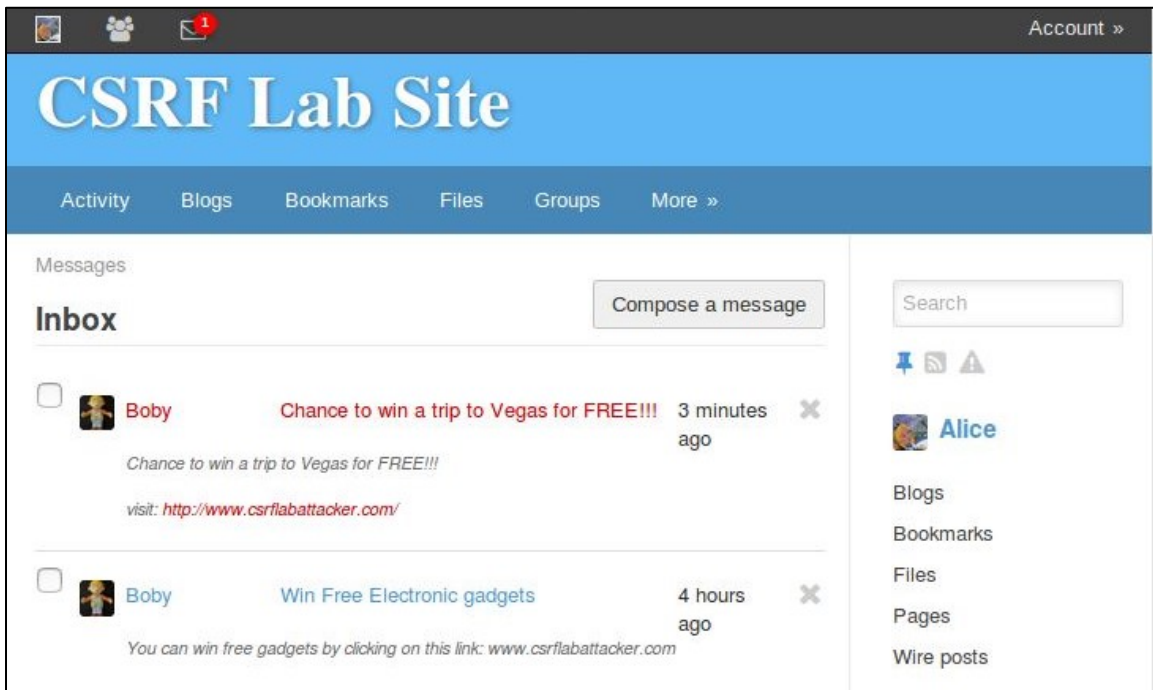


Figure 16: Boby sends a message to Alice

2. In /var/www/CSRF/attacker, we modify the script and add JavaScript (Figure 17).

```
GNU nano 2.5.3 File: index.html
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
var fields;
// The following are form entries need to be filled ou$
// The entries are made hidden, so the victim won't be$
fields += "<input type='hidden' name='name' value='Ali$
fields += "<input type='hidden' name='description' val$
fields += "<input type='hidden' name='accesslevel[desc$
fields += "<input type='hidden' name='briefdescription$
fields += "<input type='hidden' name='accesslevel[brie$
fields += "<input type='hidden' name='location' value=$
fields += "<input type='hidden' name='accesslevel[loca$
fields += "<input type='hidden' name='guid' value='42'$
}
</script>

```

[Read 38 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit ^R Read File ^\ Replace ^U Uncut Text

Figure 17: Added JavaScript on index.html

3. After we created the web page, Alice clicks the web page and you should see it connect the malicious page and then it redirected back to Alice's profile automatically (Figure 18, Figure 19).



Figure 18: www.csrlabattacker.com

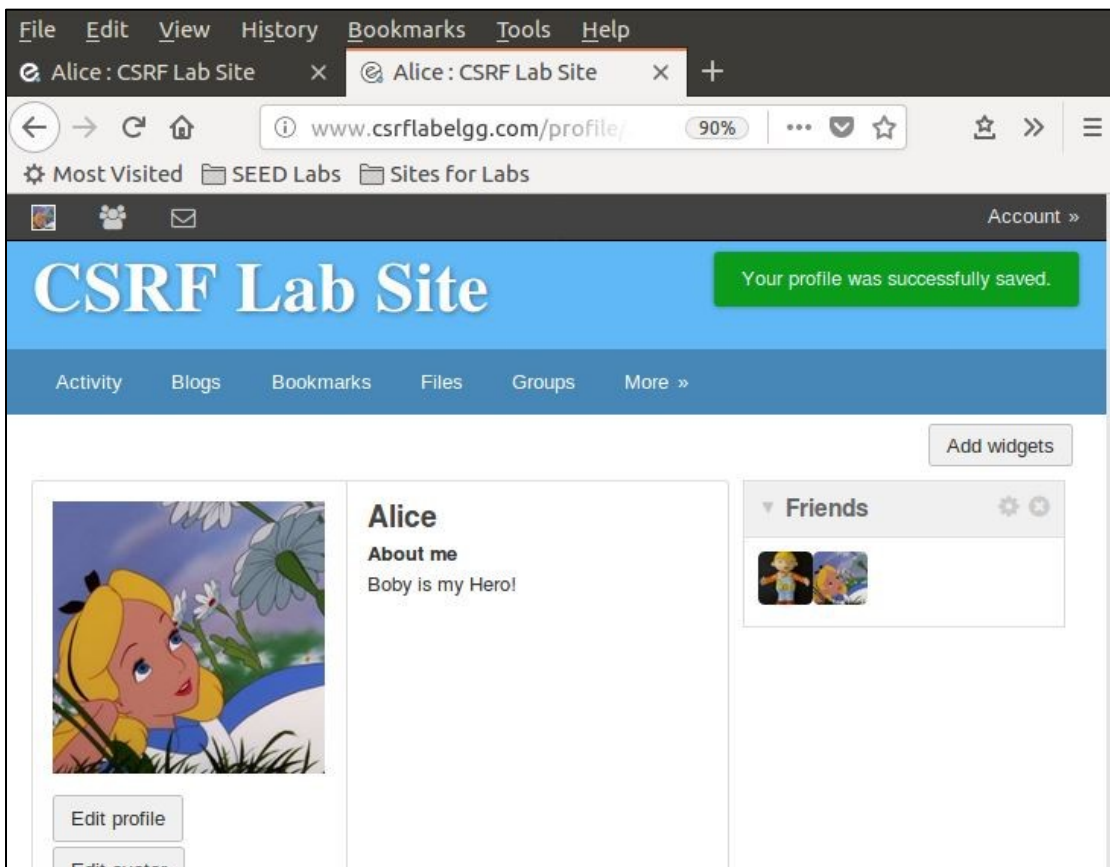


Figure 19: The malicious website redirected to Alice's profile

4. In HTTP request, you should see the HTTP request referrer to www.csrfbattacker.com (Figure 20).

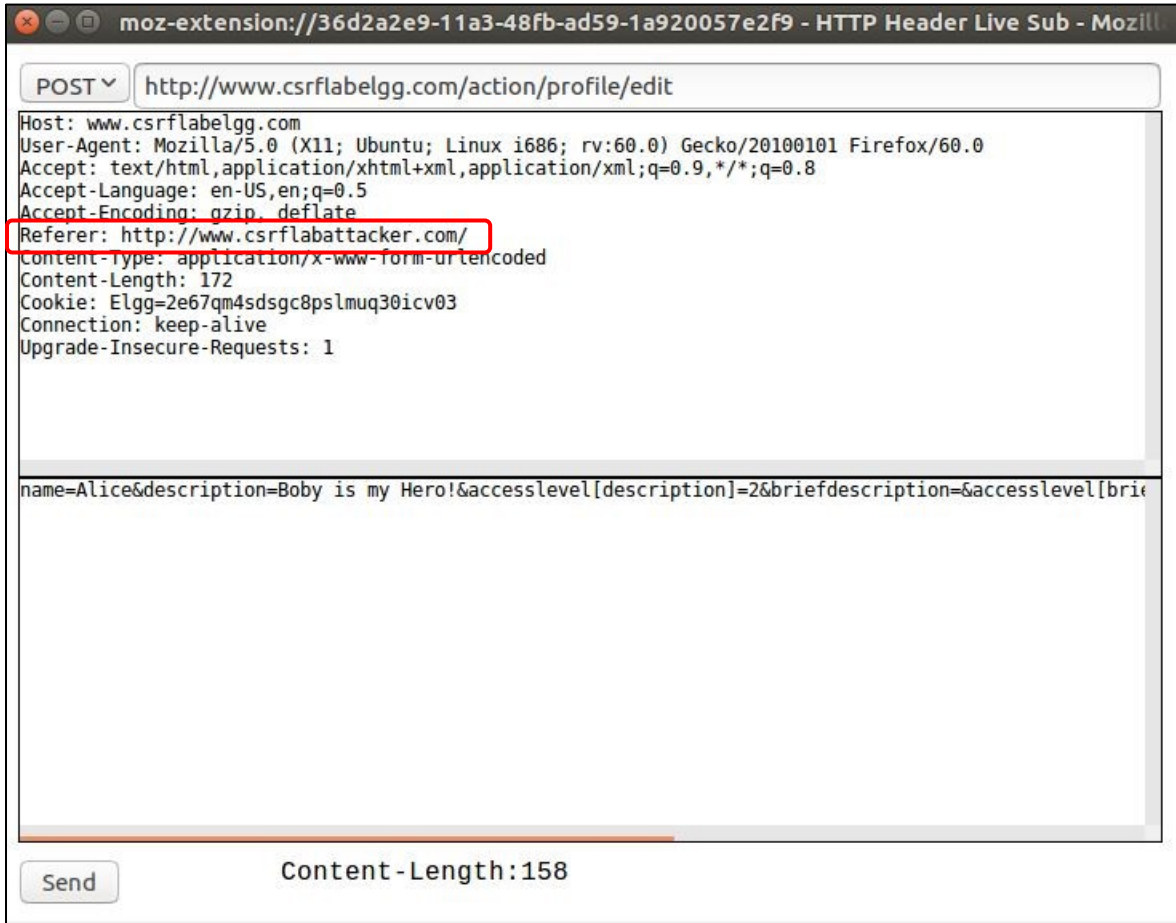
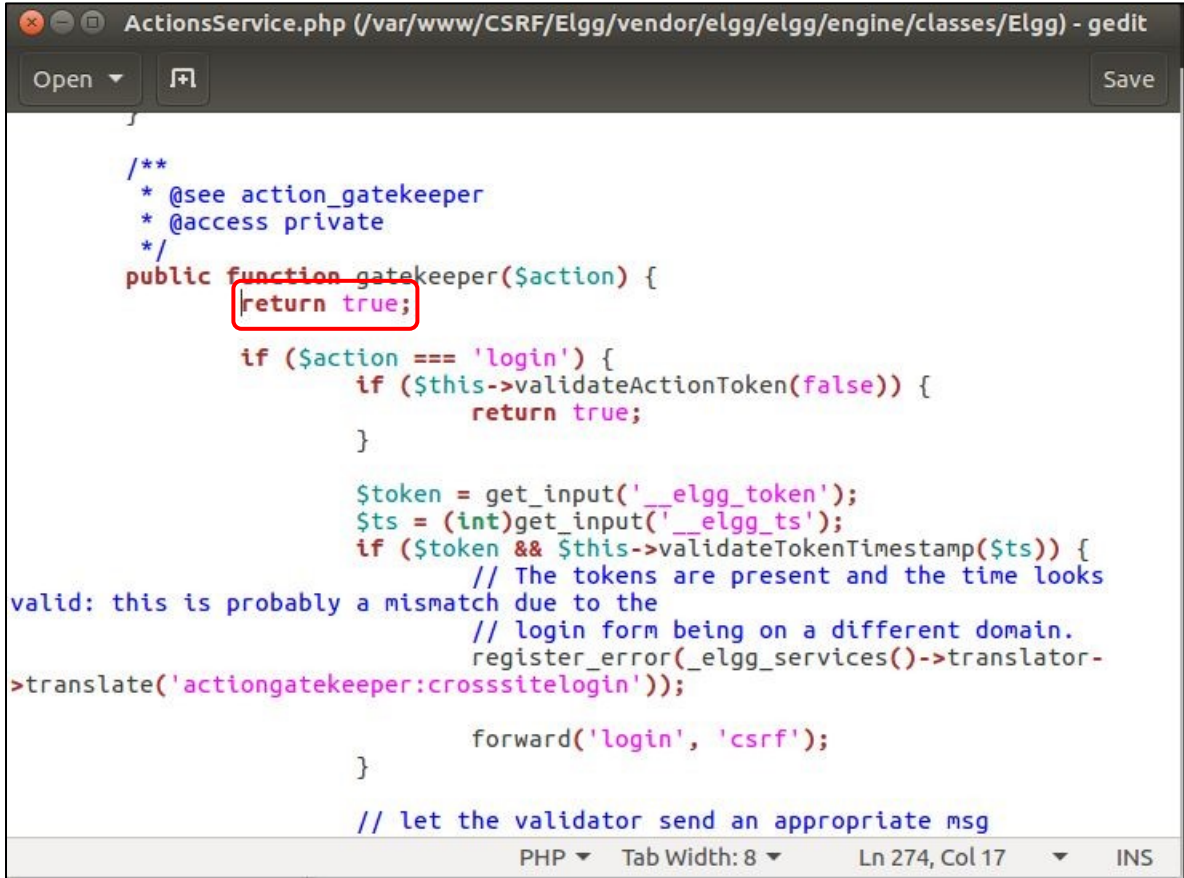


Figure 20: HTTP POST request

Task 4: Implementing a countermeasure for Elgg

1. In this task, we need to turn countermeasure on and off by modifying the file "ActionsServices.php" in /var/www.CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg/ (Figure 21).



```
ActionsService.php (/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg) - gedit

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

    if ($action === 'login') {
        if ($this->validateActionToken(false)) {
            return true;
        }

        $token = get_input('__elgg_token');
        $ts = (int)get_input('__elgg_ts');
        if ($token && $this->validateTokenTimestamp($ts)) {
            // The tokens are present and the time looks
            // login form being on a different domain.
            register_error(_elgg_services()->translator-
>translate('actiongatekeeper:crosssitelogs'));
            forward('login', 'csrf');
        }

        // let the validator send an appropriate msg
    }
}
```

Figure 21: ActionsServices.php

2. Clear cookies on the victim's browser and click www.csrfattacker.com, you should see the given below (Figure 22).



Figure 22: www.csrfattacker.com

Because we turn off the countermeasure, the attacker cannot modify Alice's profile anymore (Figure 23).

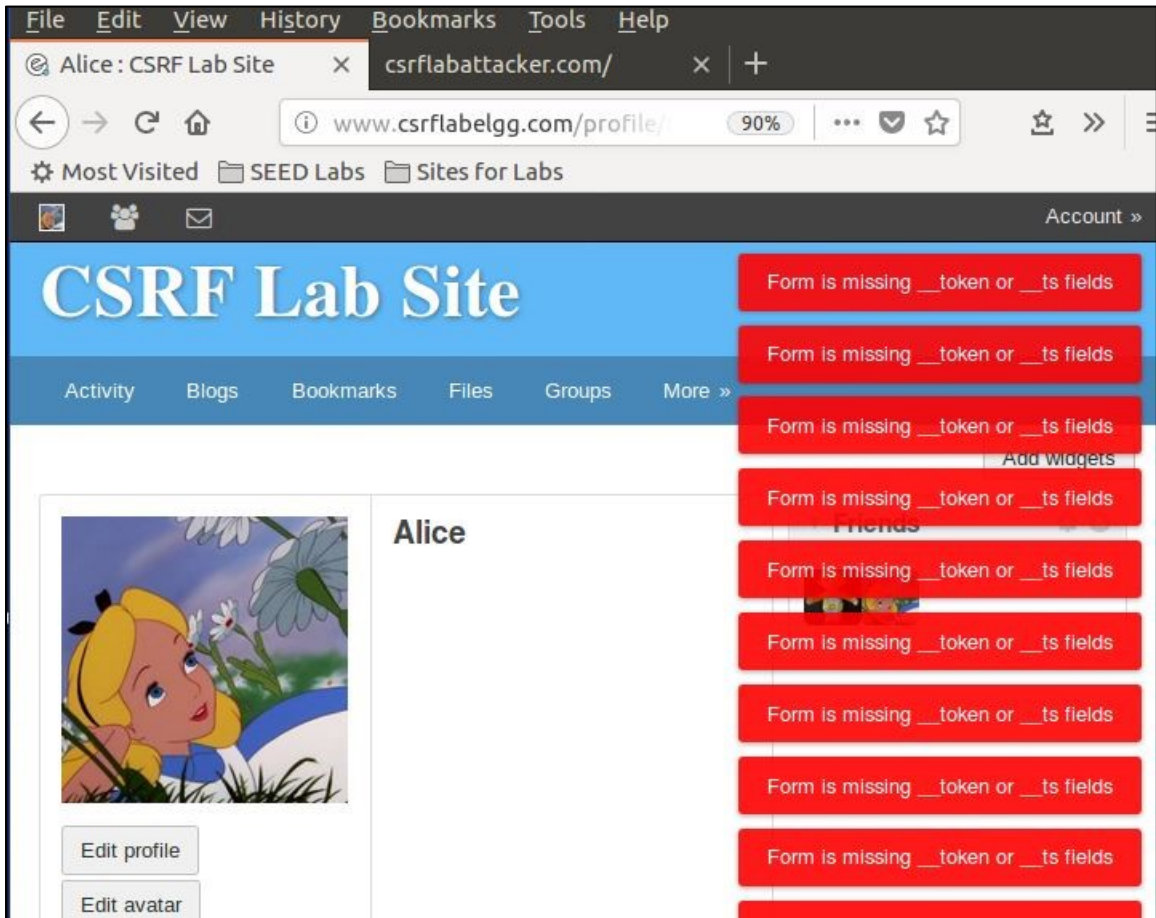


Figure 23: The malicious website redirected to Alice's profile

3. Observing HTTP request, you should see POST and GET as given below (Figure 24, Figure 25).



Figure 24: HTTP GET request

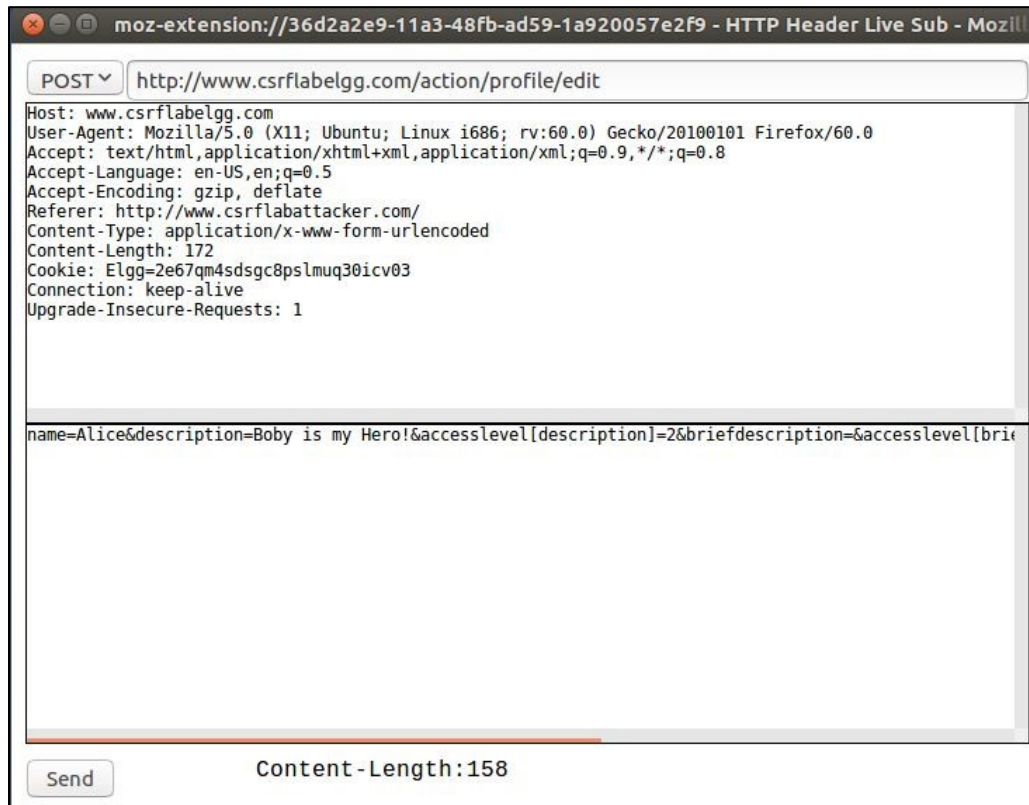


Figure 25: HTTP POST request