1 CSRF Lab Writeup

Ethical Hacking 2022/23, University of Padua

Eleonora Losiouk, Alessandro Brighente, Gabriele Orazi, Francesco Marchiori

---

1.1 Task 1

We can intercept a GET and a POST request. To generate a POST request, we can, for instance, create a new blog post. Then, the POST request generated will generally have the following parameters:

```
POST /action/blog/save HTTP/1.1
Host: www.seed-server.com
...
Cookie: Elgg=rh9bshn1ken1pigkgcd7vjci7r
...

__elgg_token=ehtSovuALQdjfwAMlQ9EKQ&__elgg_ts=1642425536&title=test&excerpt=test&
    description=%3cp%3etest%3c%2fp%3e%0d%0a&tags=&comments_on=On&access_id=2&
    status=published&container_guid=56&guid=&save=Save
```

1.2 Task 2

For this task, you simply have to insert the correct parameters and the correct URL on the HTML code already provided:

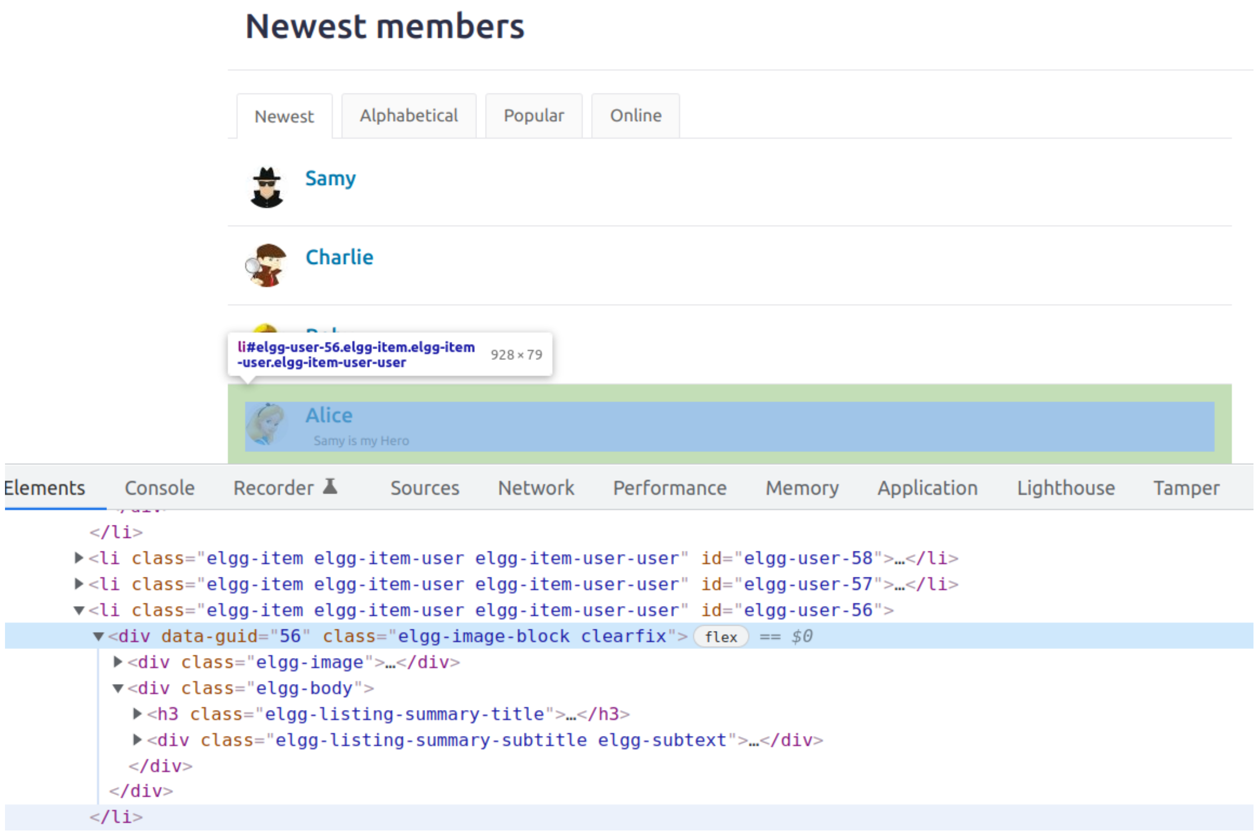
```
...
fields += "<input type='hidden' name='name' value='Alice'>";
fields += "<input type='hidden' name='briefdescription' value='Samy is my Hero'>";
fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
fields += "<input type='hidden' name='guid' value='56'>";

...

// Construct the form
p.action = "http://www.seed-server.com/action/profile/edit";
...
```

The answers to the questions are: 1. Boby can intercept the request used to add Alice as a friend. By doing so, he can see the `friend=56` parameter in the request, which value is the Alice's guid. Another way to find the guid is to look at the source of the member page as follows:

1. Alice cannot launch this attack on anybody who visits her malicious webpage since the guid of every user is different, and only when the user id of the logged-in user and the user id specified in the webpage matches, the attack can be successful. Furthermore, the attack takes place only if the user id specified in the webpage has an active session with Elgg when visiting the malicious webpage. So, by changing the guid, we can perform the attack on other users too.

1.3 Task 3

For this task, you have to activate the CSRF protection as presented in the Readme, and verify that the edit profile attack is not working anymore.



```
  </li>
 ▶<li class="elgg-item elgg-item-user elgg-item-user-user" id="elgg-user-58">…</li>
 ▶<li class="elgg-item elgg-item-user elgg-item-user-user" id="elgg-user-57">…</li>
 ▼<li class="elgg-item elgg-item-user elgg-item-user-user" id="elgg-user-56">
   ▼<div data-guid="56" class="elgg-image-block clearfix"> flex == $0
     ▶<div class="elgg-image">…</div>
     ▼<div class="elgg-body">
       ▶<h3 class="elgg-listing-summary-title">…</h3>
       ▶<div class="elgg-listing-summary-subtitle elgg-subtext">…</div>
      </div>
    </div>
  </li>
```

Figure 1: guid

The countermeasure is to send the two extra fields, timestamp and a unique token, along with each request. When the countermeasure is turned on, it compares these values and checks if they are valid in the current session with the user. The secret token validation fails if we perform the attack when the countermeasure is turned on because it identifies a cross site request and not a request from the user.

1.4 Task 4

1. Normal cookies are always sent. `Lax` allows the cookie to be sent on some cross-site requests, whereas `Strict` never allows the cookie to be sent on a cross-site request.

   The situations in which `Lax` cookies can be sent cross-site must satisfy both of the following:

   - The request must be a top-level navigation. You can think of this as equivalent to when the URL shown in the URL bar changes, e.g., a user clicking on a link to go to another site.
   - The request method must be safe (e.g., GET or HEAD, but not POST).

   Therefore, in Experiment A, cookies are always sent since all the requests are in the same domain. Instead, on Experiment B, the `Strict` cookies is not sent when pressing the link or when performing a GET request. Finally, with the POST request, no cookies are sent.

2. Since SameSite cookie cannot be sent to another domain, if you use a session token that has the SameSite attribute, it will not be sent if the origin of the request is another (possibly not trusted) website.

3. By using a SameSite session cookies, you can avoid requests coming from malicious websites that can perform CSRF attacks since they will not be authenticated.

More info on SameSite cookies against CSRF can be found here.