Task 2 XForge

Task 1

After logging in as Alice and figuring out the Add Friend `GET` request, I copied and pasted the `GET` request message and removed everything except the friend ID number, which was `40`. I removed the other values since they would be filled out by the browser due to the fact that user already logged in, as we learned in class, because it attaches any required cookie values for us. Then I went to the malicious site's HTML code and created an `img` tag with the `src` set to `"http://csrflabelgg.com/action/friends/add?friend=40"`, which is the request to add Boby as a friend. Then I pretended to be Boby and posted a blog about a cool website, and poor Alice happened to click on it and opened the website, where the `img` tag with that `src` ran the GET request for us. It even crashed Alice's browser for some reason, but I checked on Boby's tab, and Alice is now our friend, albeit forced to be.

Task 2:

After logging in as Boby, I tried a test message by editing the description of himself, which generated a `POST` request. I then copied the request parameters and manually matched those fields with the `POST` request script input form, as shown at the bottom of the lab. I left out the tokens and cookies as usual since those would be filled by the browser due to the fact that the user is already logged in. Then I copied the script to the `index.html` page and used Boby's browser to visit the malicious website that Alice had made a blog post about. Boby's description is now `"I support SEED project!"`.

1. We can always obtain someone else's friend ID with the Add Friend `GET` request, as the other person's ID is within that request. We can then just copy that over to the malicious website and target someone else.

2. No, we cannot launch the attack since we are only able to bypass the SOP for tokens/cookies. If we need to obtain a visitor's GID by simply visiting the

malicious website, it is not possible, as our website cannot read items from the actual Elgg website and therefore cannot forge a request.

Task 3

After enabling the countermeasure mentioned above, I attempted the CSRF attack again. This time, it did not work, as I was not redirected back to the Elgg page. Instead, the request kept being sent repeatedly. When I returned to the Elgg page, I saw numerous red dialog boxes stating that the form was missing the `__token` or `__ts` field. This occurred because the secret token needed to validate the request was not present. The reason we cannot access this token is that it is protected by the SOP, and the attacker does not have access to those values as these tokens are not automatically included in the `POST` request. Therefore, the attacker cannot obtain or bypass them like they could with the session token previously. If we look at a legitimate request, something like `__elgg_token=258df318b4458a912f0b0d24230666e2&__elgg_ts=1729729955` will be pass along the request.