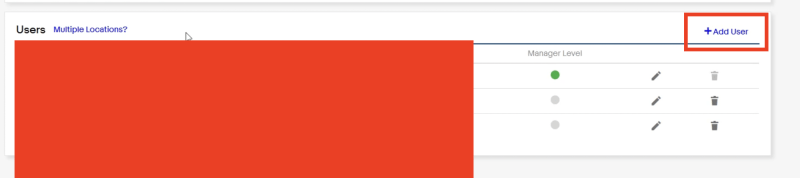


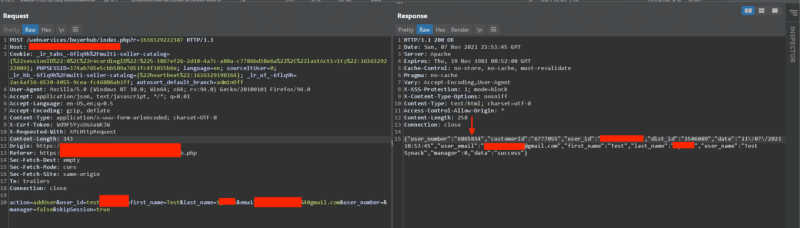
So, I became a part of a new program. Upon visiting the scope URL,it turned out to be an ecommerce application that had two user roles.

1. Seller
2. User

The seller role had permission to invite new users. So I tried to test this feature as it looked intresting.



I quickly sent an invitation to my email id and checked the response in my Burp proxy. I saw that the send invitation response contained a numeric parameter named **user\_number**

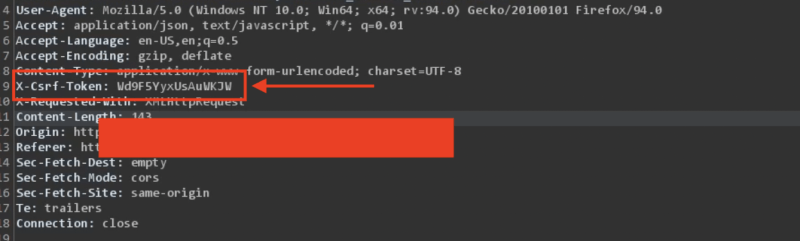


To check whether the userID was incremental or not, I sent another invitation and as suspected it was in the incremental order. So, if the previous ID was 6865834, then the next ID would be 6865835, 6865836 and so on.

The first thing that came to my mind was to test for IDOR on delete account feature. To do that, I quickly intercepted the delete user request and sent it to Burp Repeater.

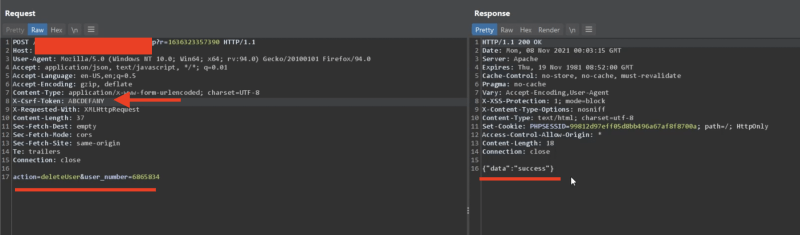
Next thing, was to remove the cookie header completely and check whether it worked or not, and yes it was working. This meant any attacker could delete anyone’s account without being authenticated by the application.

But the problem was this particular request that had a **X-Csrf-token** header, which contained a random CSRF token in the request. Without that header, the response was ‘403 forbidden’.



After a while I understood that there was no need to put an actual CSRF token. You could put any random string like ABCD in the header, so you could put like **X-Csrf-Token: ABCD.** and it would get accepted by the server.

The final request for the account deletion was



This issue was resolved by enforcing the validation on **cookie** header, CSRF token validation on the delete user endpoint and by using a random user ID.

Takeaway

* Always try to play with request headers
* Use GUIDs for referencing the objects.
* Always impose server side validation on CSRF tokens

[wpcode id="7804"]

* Tags: [RedTeam](https://payatu.com/tag/redteam/), [WebSecurity](https://payatu.com/tag/websecurity/)