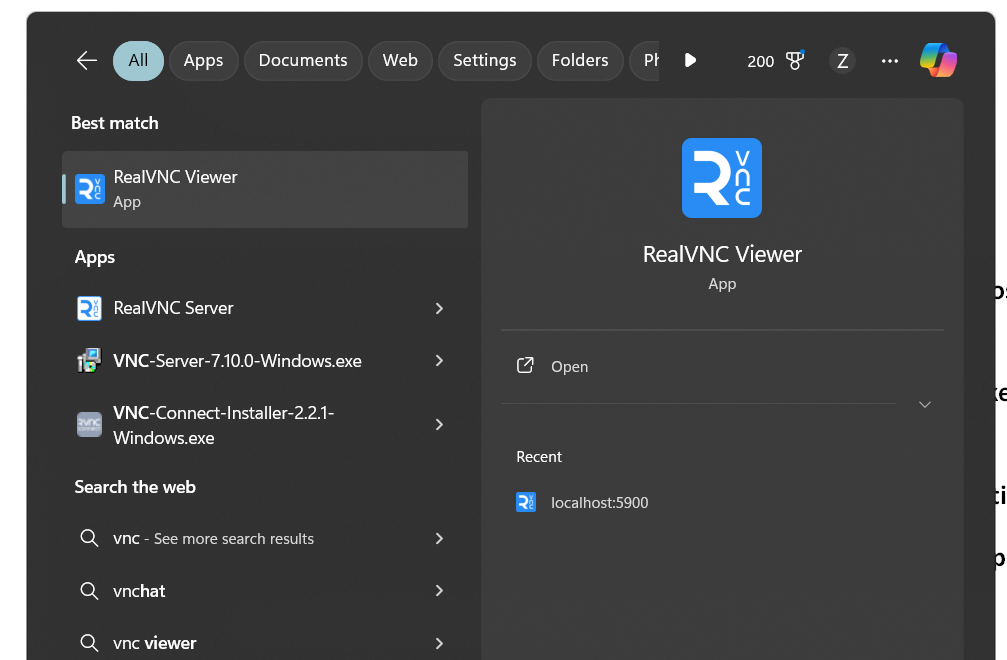
Web security – CSRF ( Cross-Site-Request-Forgery)

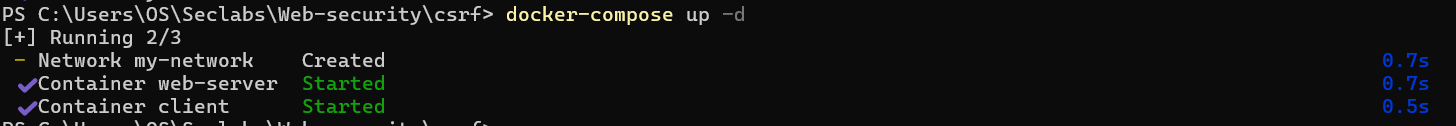
**Tasks**

**1. Setup**

**a) Install VNC viewer on the host computer.**

****

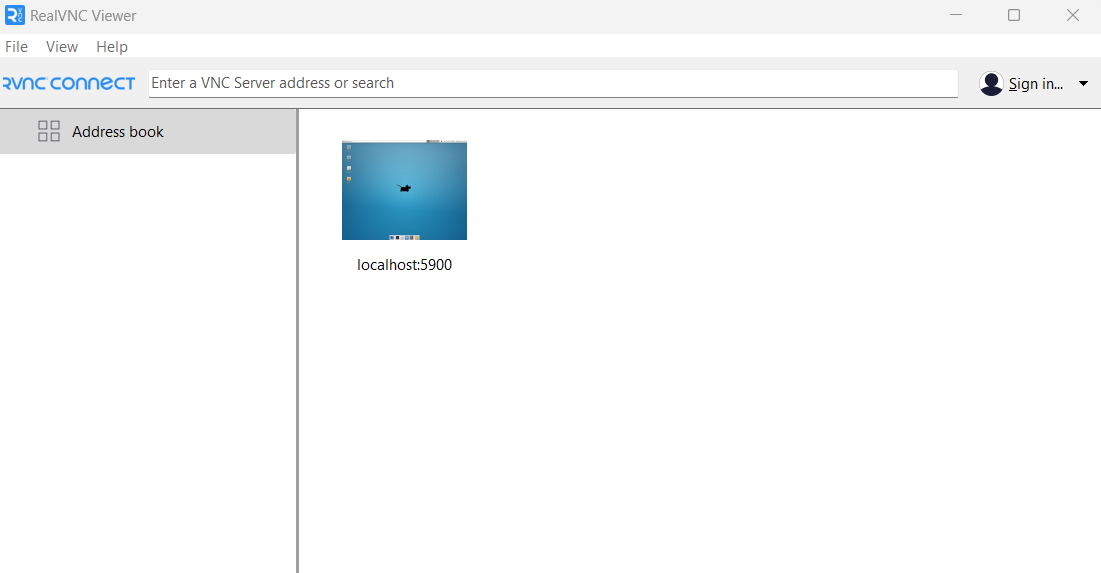
**b) In windows Terminal, change working directory to CSRF. Fire up docker-compose to create the docker-container set for this lab**

****

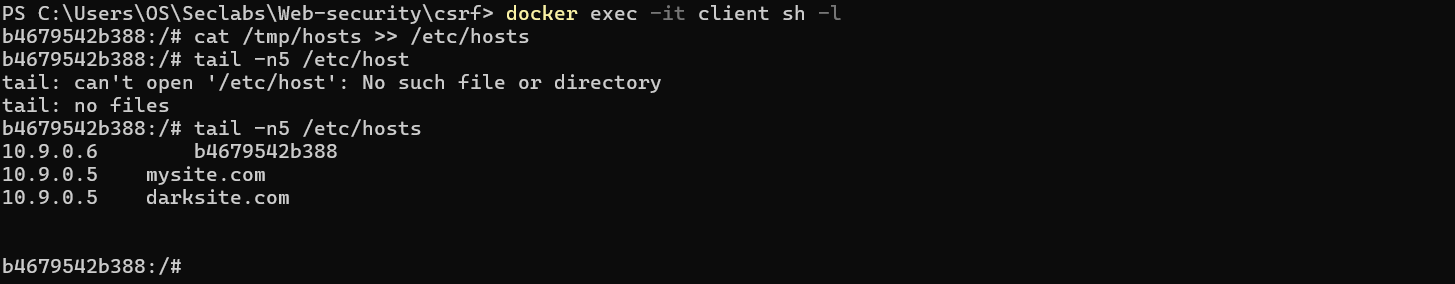
**c) Fire up VNC viewer on the host computer to connect to the web client on docker network by entering localhost:5900 on the address bar of VNC viewer. After the first connection is**

**successfully established, a thumbnail of web-client computer in VNC will show up ready for**

**connection.**

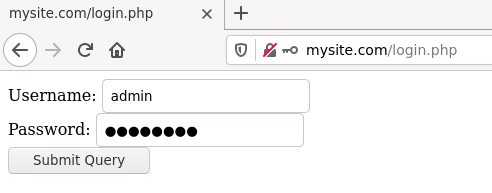
****

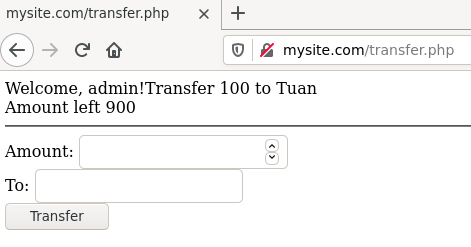
**d) Attach to the console of web-client docker container**

****

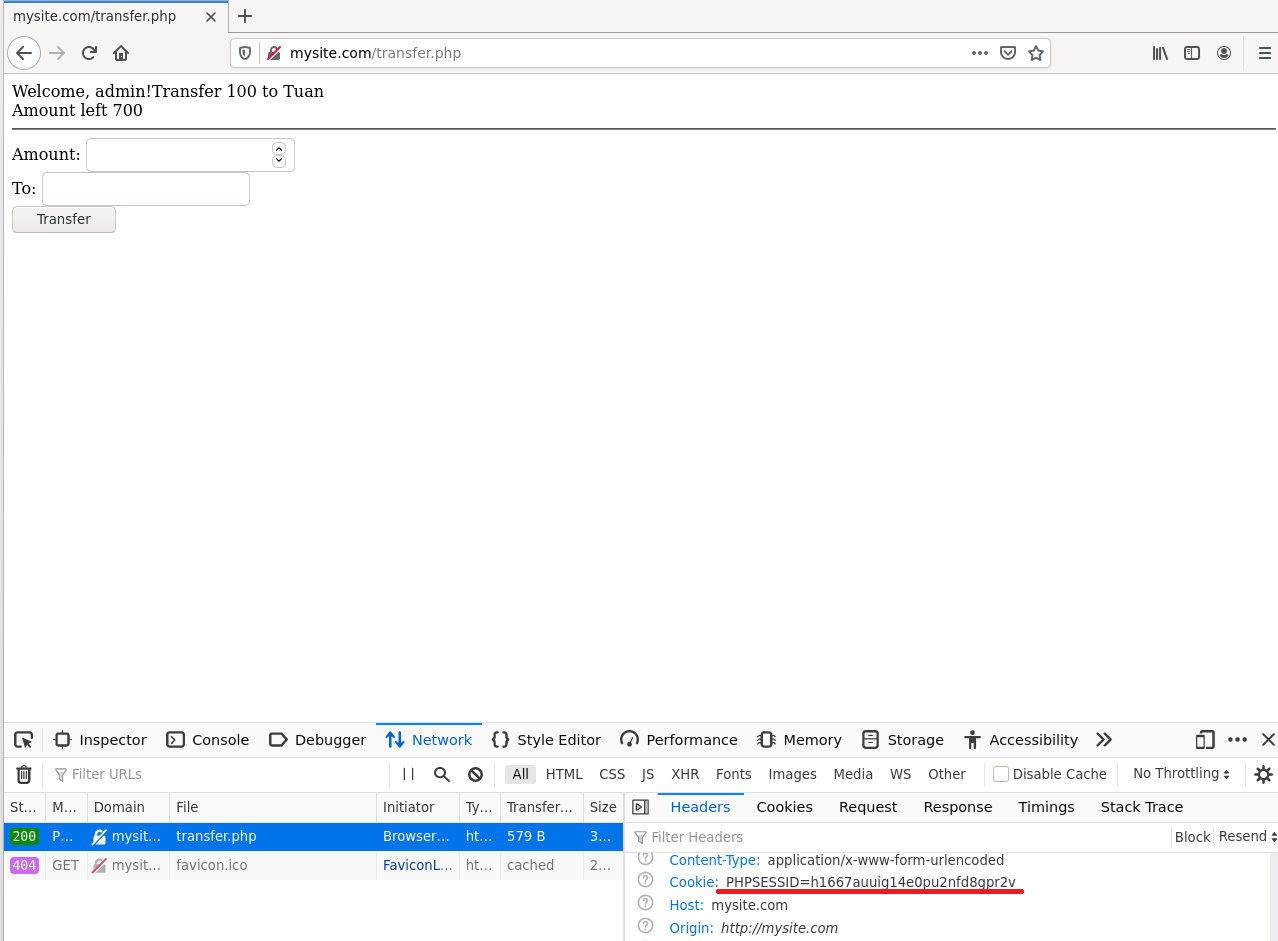
**2. Conducting the Transaction**

**a) Fire up Firefox inside the web-client container, enter http://mysite.com/login.php to login with credentials (admin, password) After successfully login, enter amount and the receiver’s name, click Transfer to proceed doing the transaction.**

****

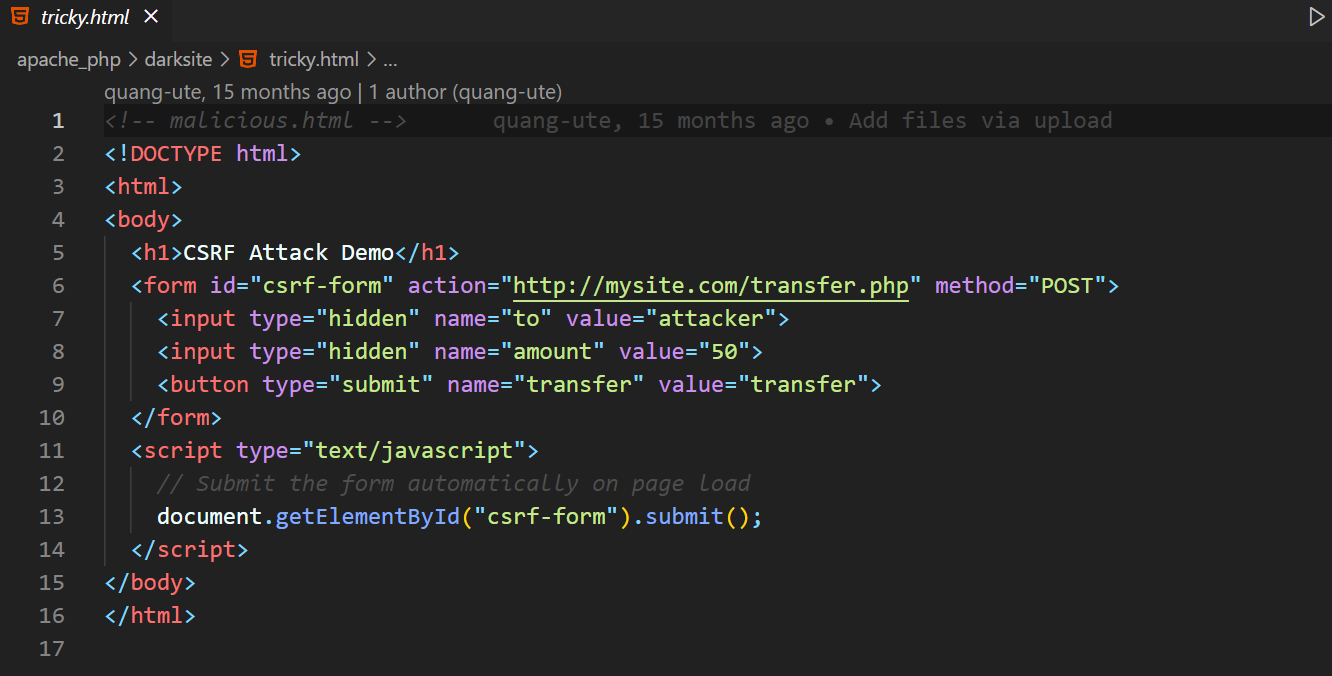
****

**b) Open Developer Tool (Ctrl + Shift + K), click Network tab then click reload page to view request /response headers, take note the cookie value.**



**3. Cross-Site-Request-Forgery**

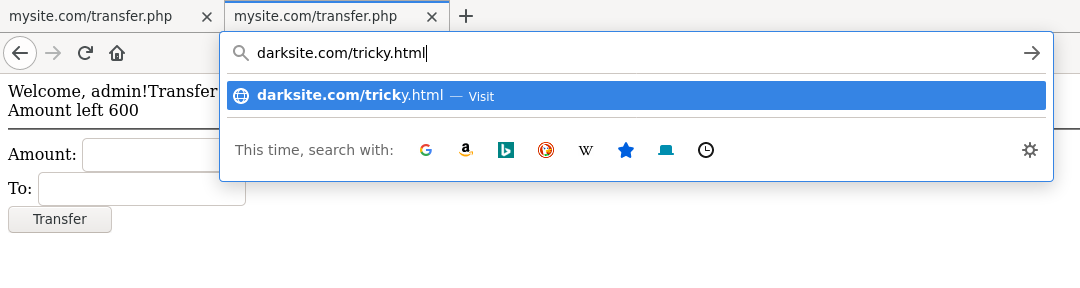
By observing page with Inspector in Developer Tool, the attacker forge the request by creating a hidden form that will be submitted automatically when loaded. This form is stored as tricky.html hosted on http://darksite.com that setup by the attacker.



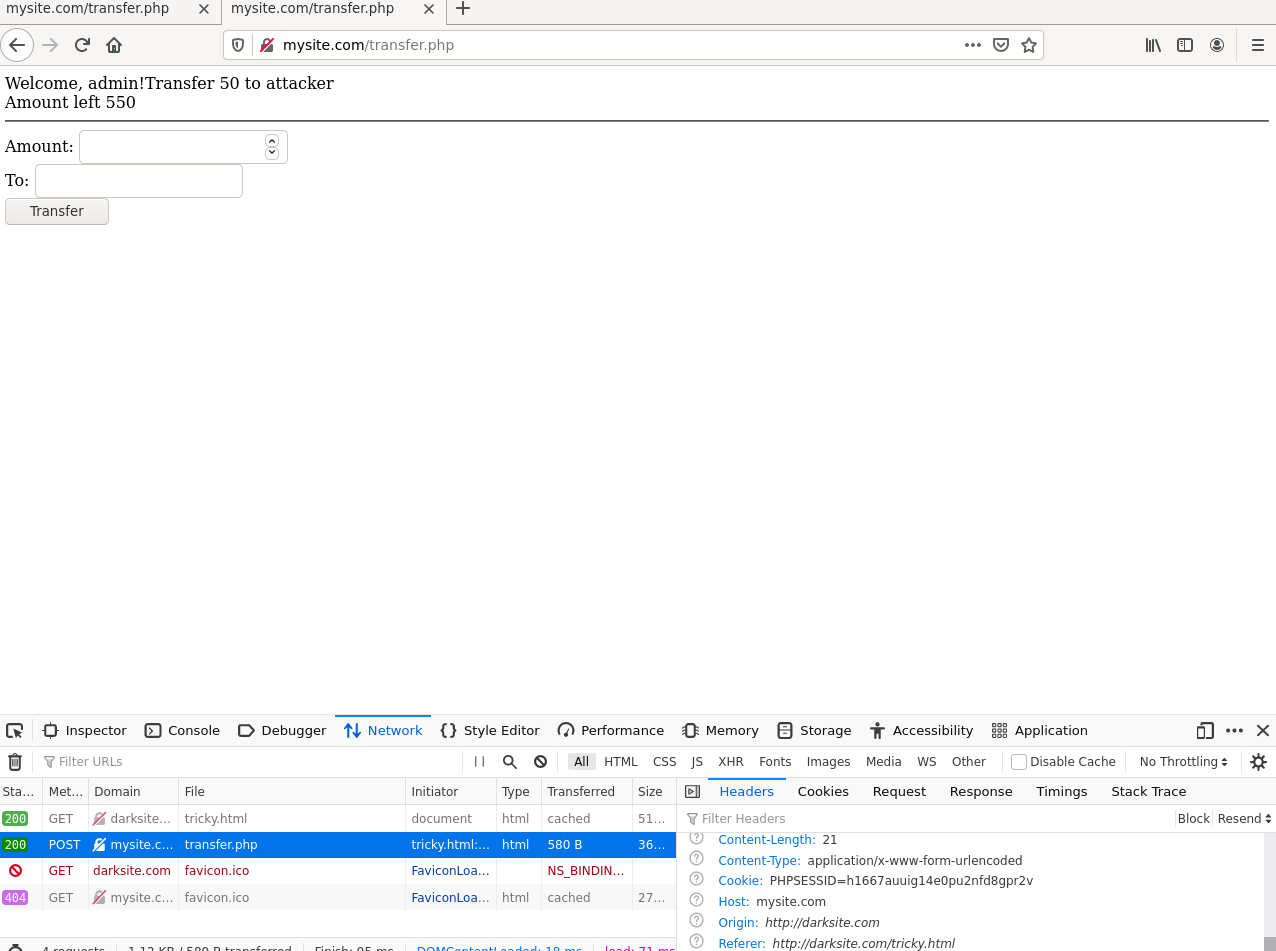
Attacker can send an email with a link that whenever user clicks the link, an amount of money from admin’s account will be transferred to the attacker.

Enter http://darksite.com/tricky.html to simulate the mentioned scenario.

**Before:**

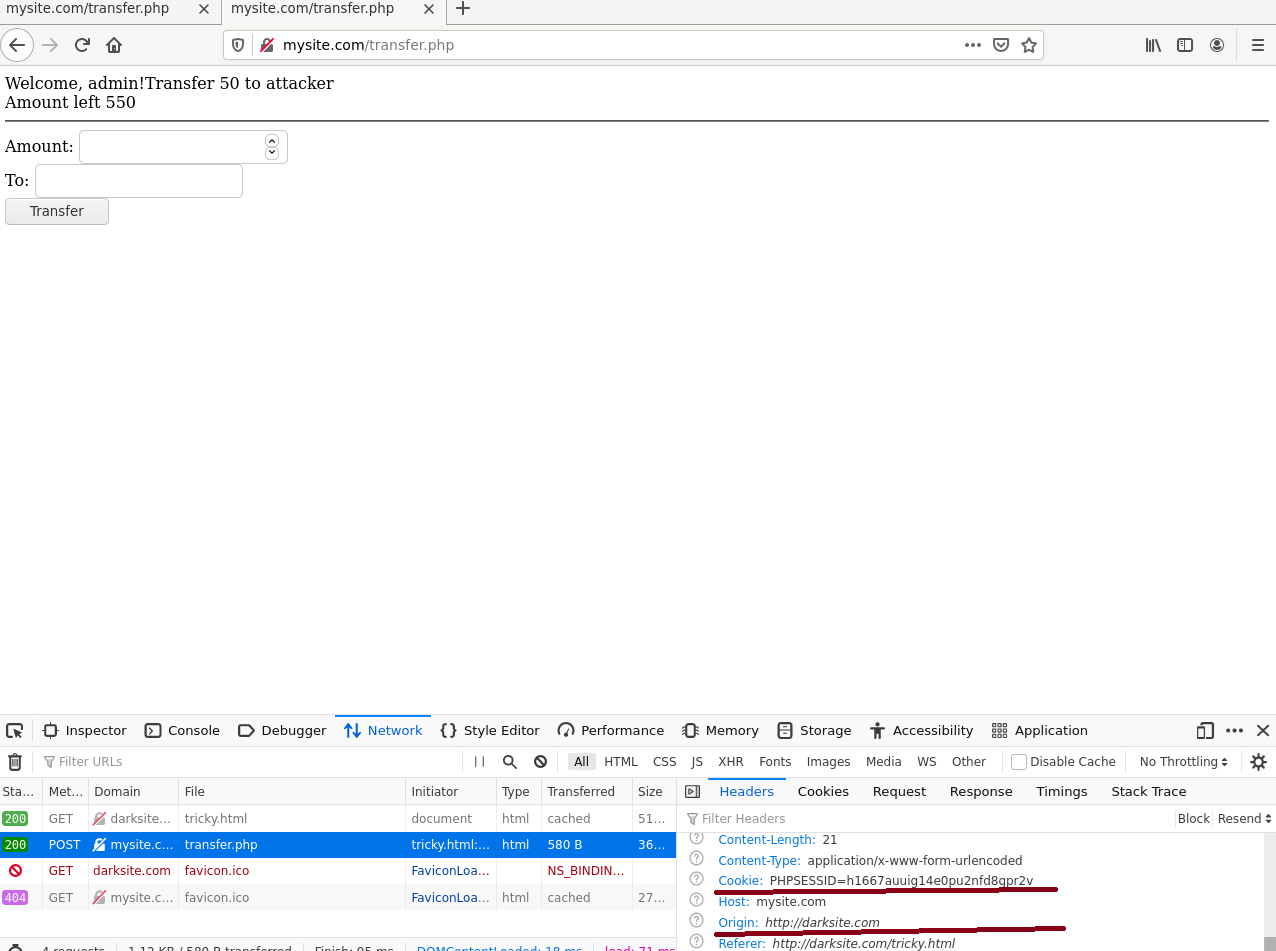


**After:**

****

Explain the reason why attacker can succeed with this forgery request:

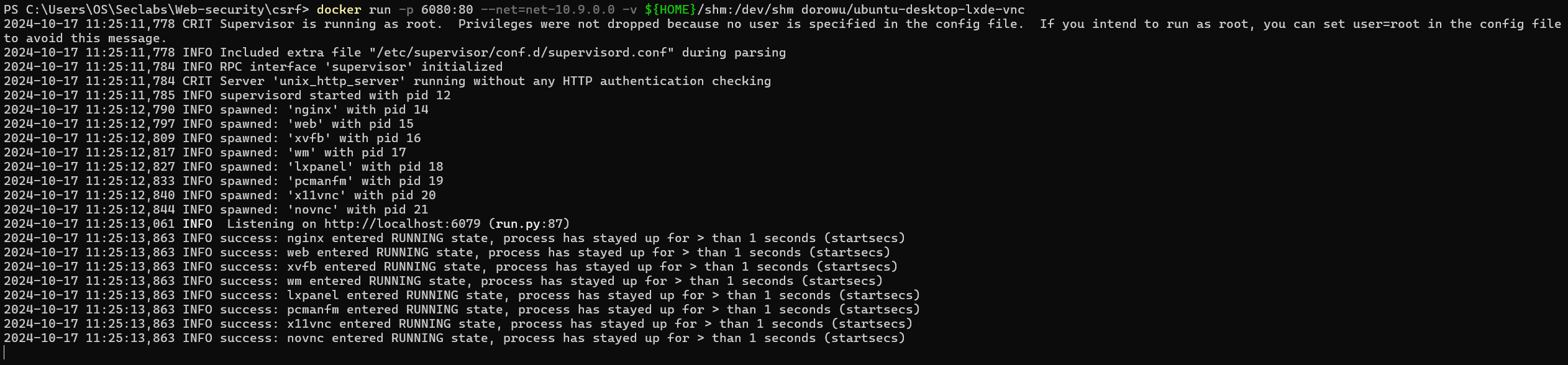
-The reason why attacker can success with this forgery request is that the browser automatically attach the cookie to the attacker request. As you can see in the picture below. The web server side does not have any method to differ between a cross site request vs same site request. Because of that, the web server will see the cross site request with the cookie is same site reqest => perform the action on behalf of the victim



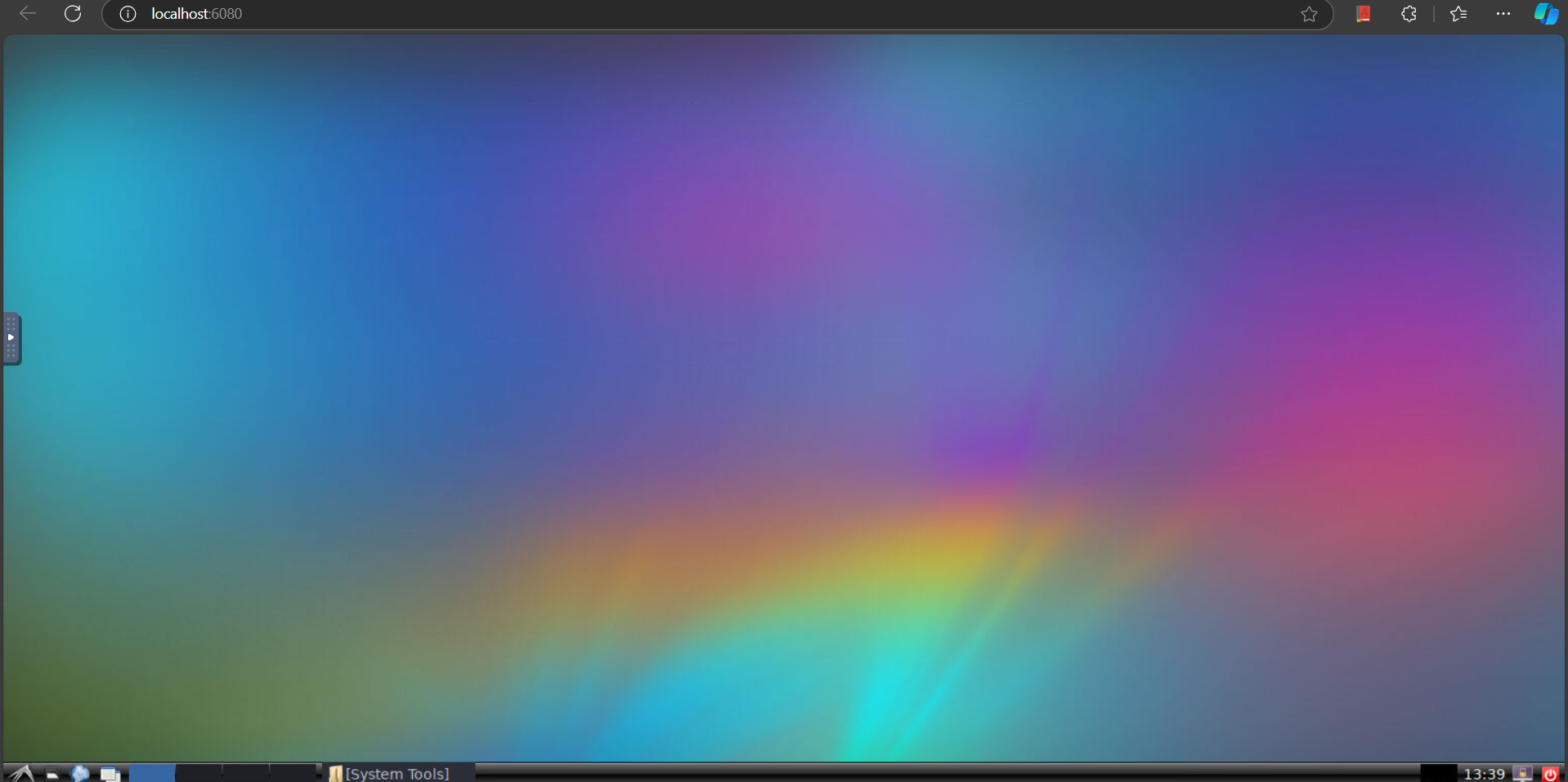
**4. Trying to attack directly without forgery request**

**a) Start an independent docker container by executing:**

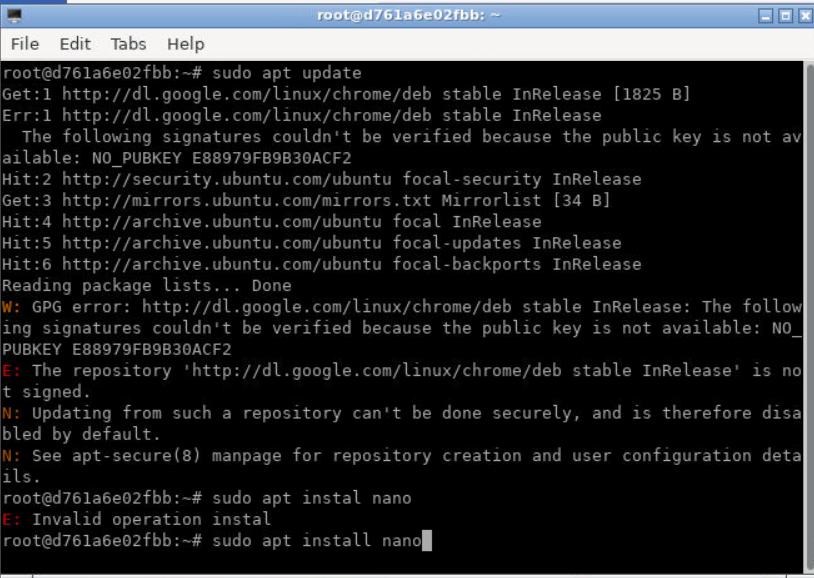
**docker run -p 6080:80 --net=net-10.9.0.0 -v ${HOME}/shm:/dev/shm dorowu/ubuntu-desktop-lxde-vnc**



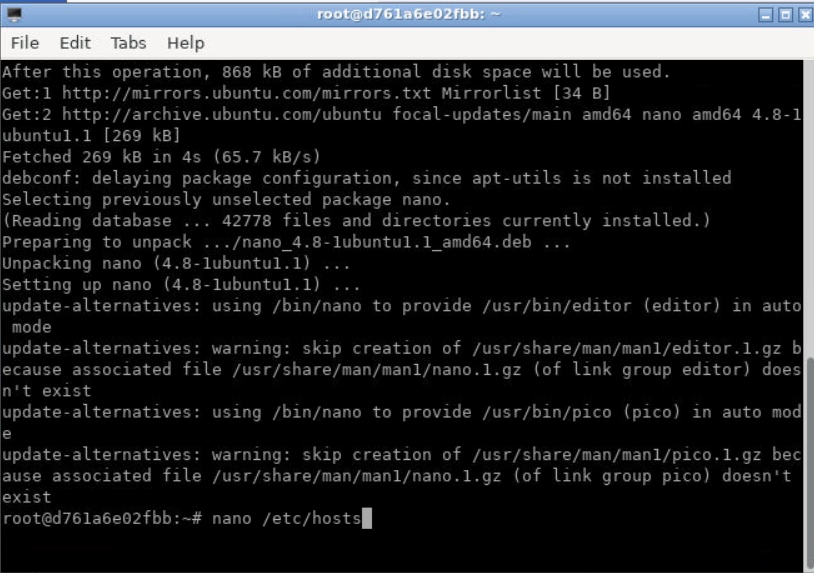
**b) Open browser then enter http://localhost:6080 in the address bar to access GUI desktop of the newly created docker container.**

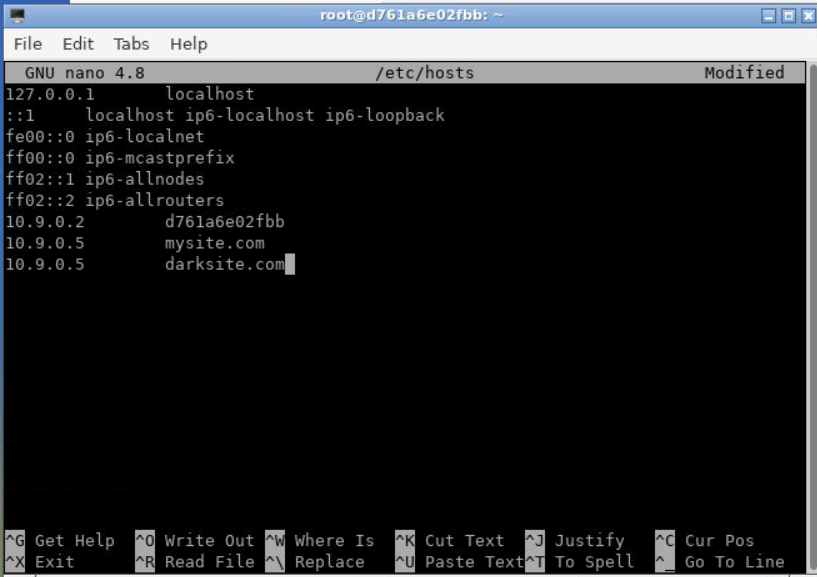


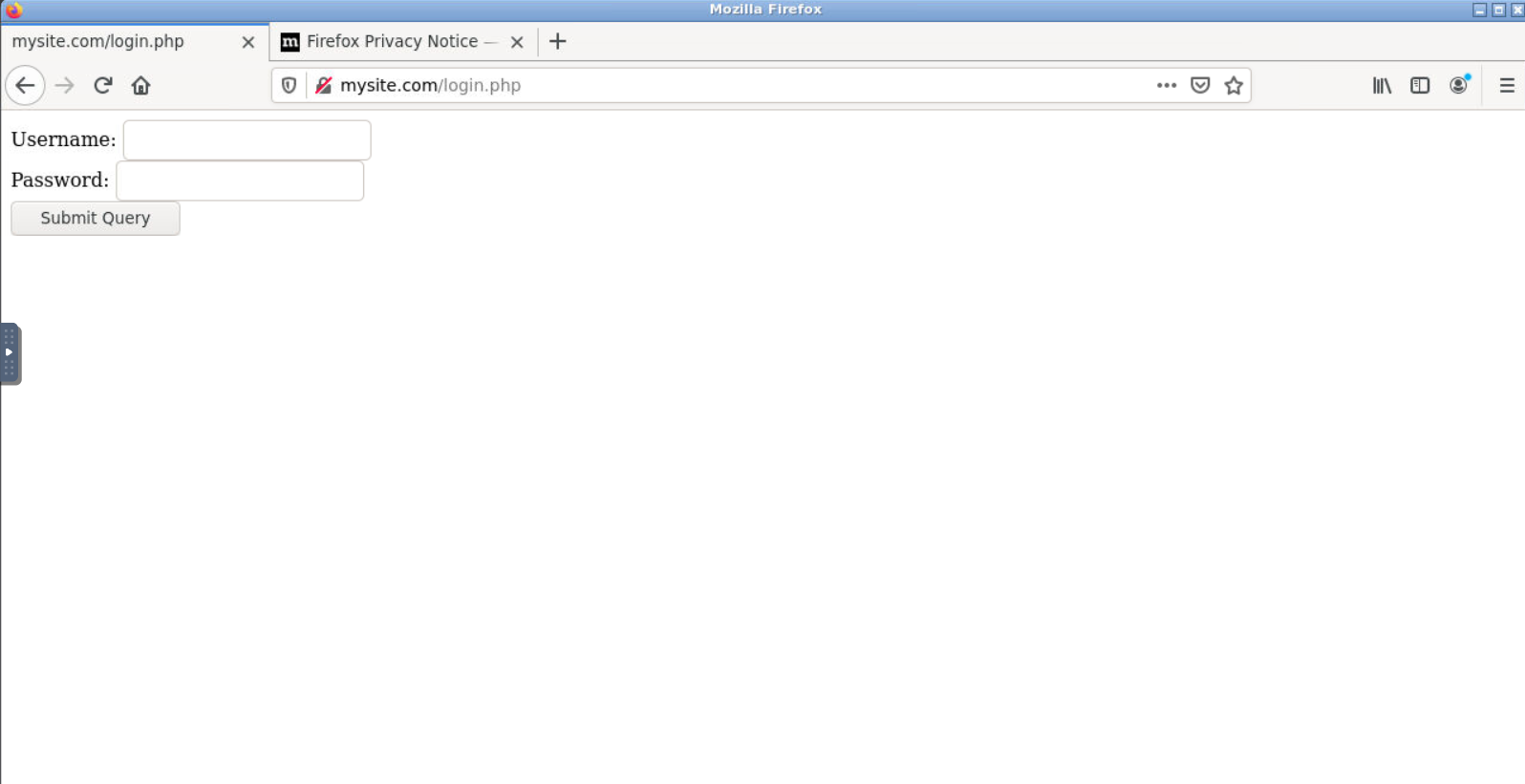
I will install nano for editing file more easier



I will edit the hosts file

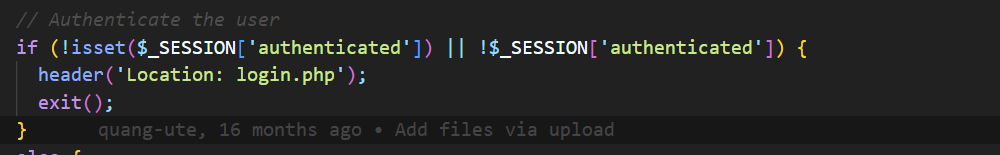




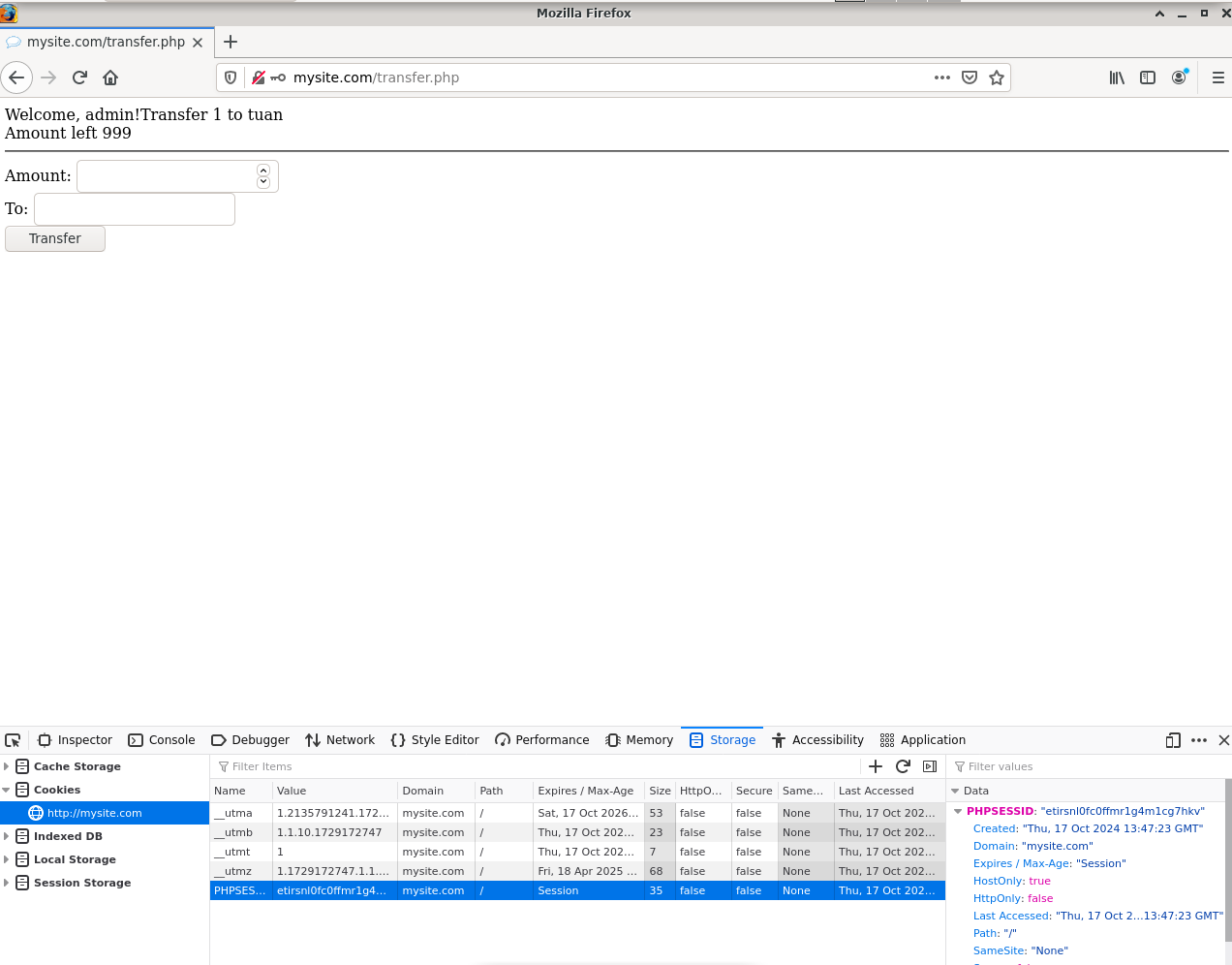


**c) Open firefox inside the docker container GUI, enter http://darksite.com/tricky.html. How does the transaction finish? Explain your observation.**

The transaction finish ending up with mysite.com asks me for login. Because we have not logined yet, when we enter darksite.com/tricky.html => the session authenticate will prevent us from doing the CSRF



So we can modify the tricky.html to send the cookie along with the form



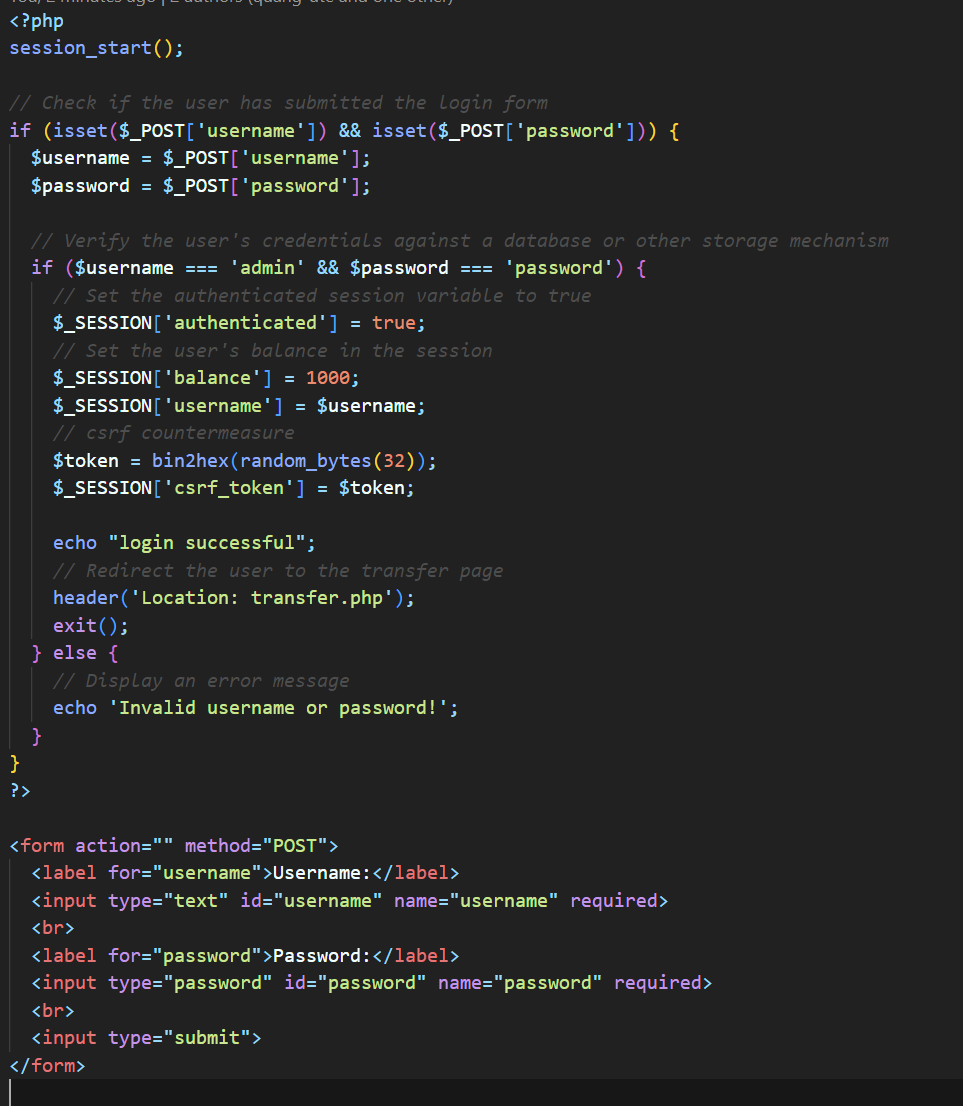
**d) Is there any way that make this attack succeed? Do it! (Modify cookie?)**

**4.5. Secure server code**

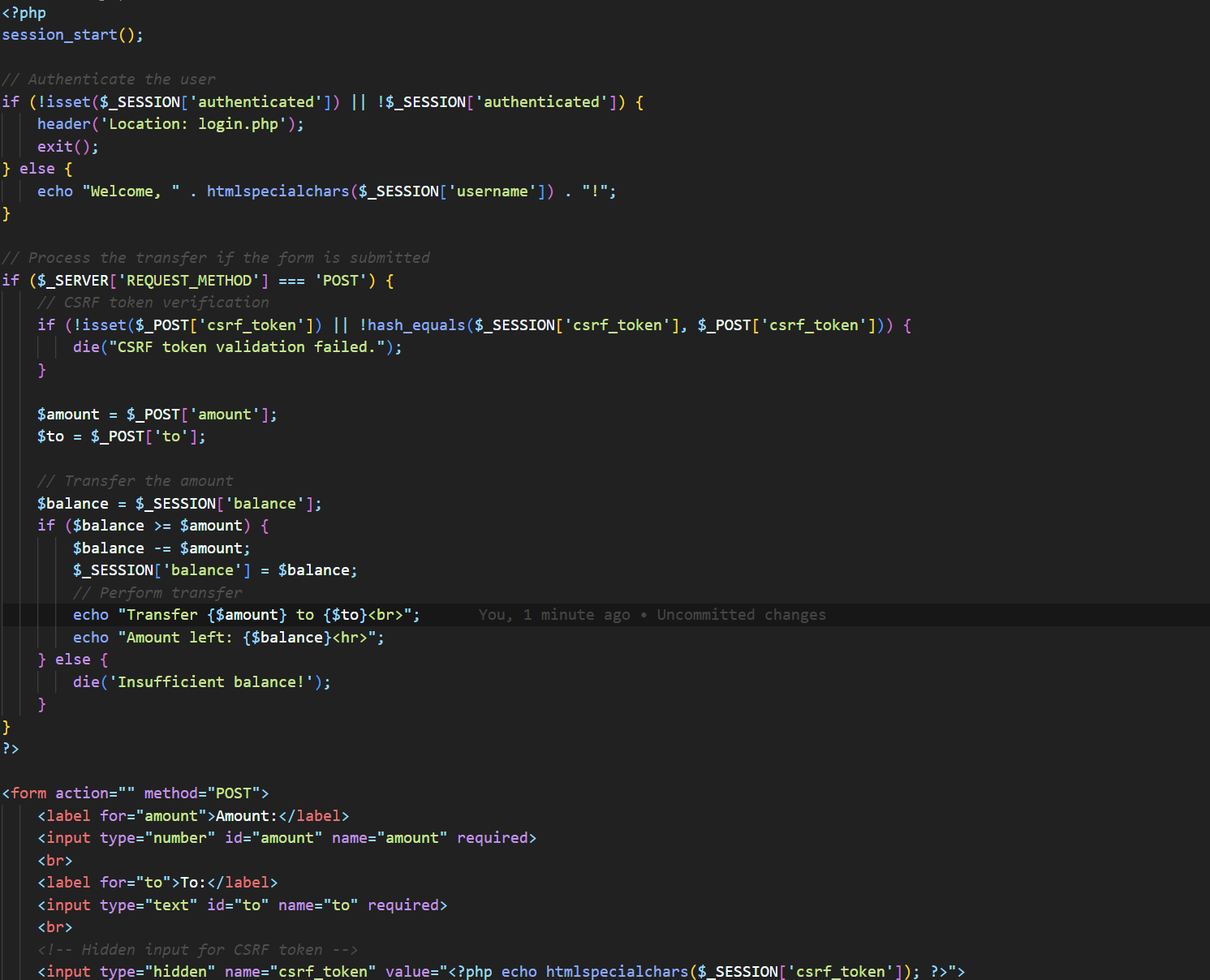
**Modify server code to protect the web server from CSRF attack.**

We can modify server code to protect the web server by creating a token, if the request token matches with the token we create in the side => then we allow

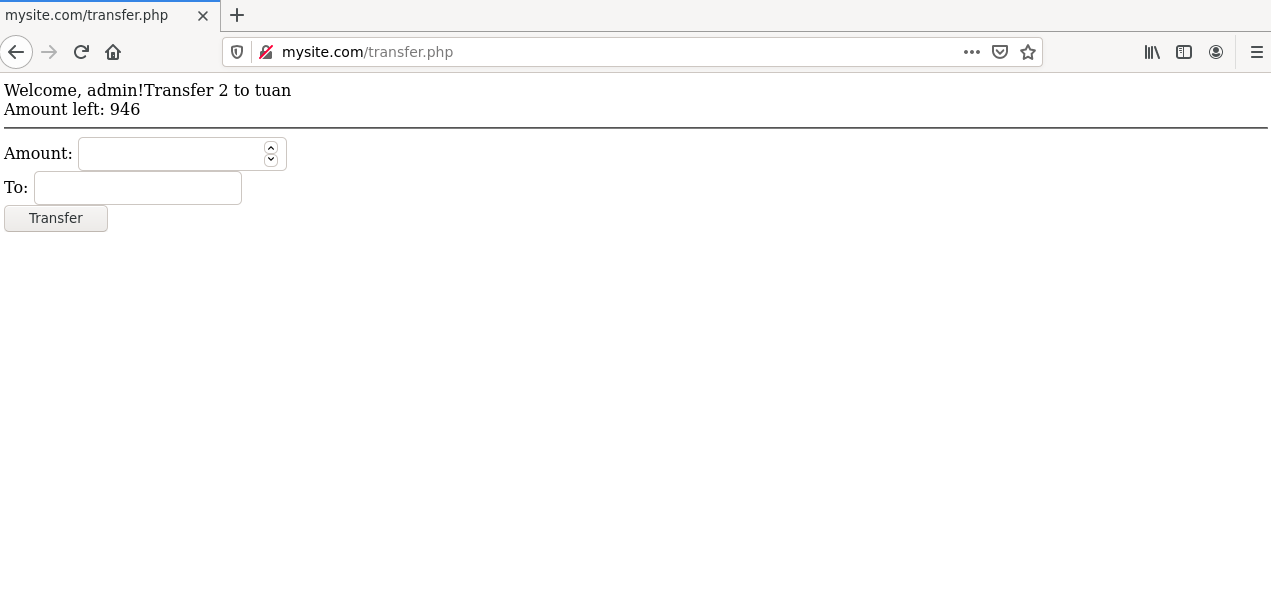
So we will give token to user when they login successfully:



And then in the transfer.php, we check if the post request csrf token is the same with the sesstion token we created => we allow them to transfer otherwise we don’t



Transfer on domain website



Transfer on darksite

