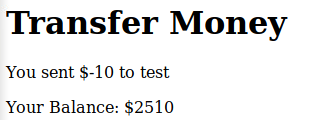
Discovery:

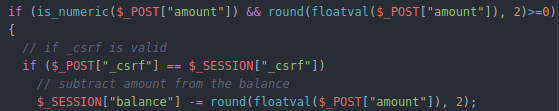
1. the non-lecture-relevant issues I found were that there is no check for negative values, and it doesn’t handle sub-dollar values.
2. This is the CSRF risk: The form contents are being sent via a get request, so an attacker could just send a victim a legitimate link with the appropriate variables and their account balance would be updated. Second issue with this is that there is no csrf token to certify the request came from our user, or our site. This means that even if we were to change the method to post, someone could just bait the user into clicking a button, and send a post to our server, from a different site accessed by the victim’s machine, Using their login session.

.gif of the vulnerability in use:

<https://gyazo.com/62f492713e3dab6ef1b806e4a84d111c>

Remediation:

First, add a check blocking anything <0, and adding sub-dollar tracking. I also found the functions floatval() which converts a form value to a floating point, if it is numeric data, and round() which truncates floating points to the specified decimal place (we can have .01 dollars, but not .001 dollars). I considered changing the form field type to number and using the ‘min’ attribute, however it occurred to me that those controls could be bypassed on the client side, by simply editing the html.



Second, I changed the method to post, this makes the form more difficult to exploit. Adding more steps to

and Third a randomly generated csrf token is (1) assigned at first visit, and (3) re-issued with each update to the page. It is (2) sent along with the form data to ensure that the request came from our user, and our site.

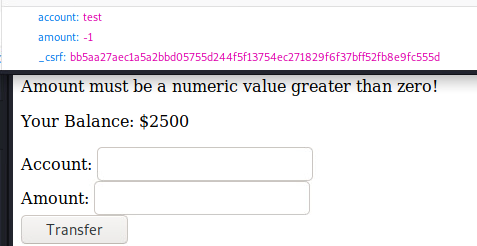
(1)

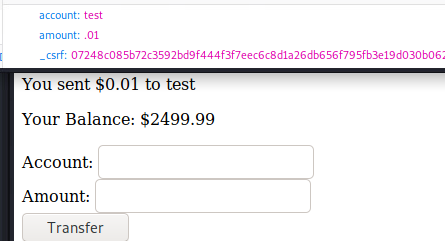
(2.1)

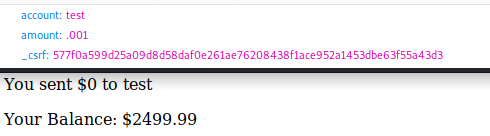
(3)

(2.2)

Testing / Demonstration of changes: since the request method was changed to post, the parameters are no longer listed in the url, I have popped out the dev tools and placed it over the firefox tray/searchbar.

**Negative check:**

**Handling of non-integers:**



**CSRF token:** as seen above, the csrf token regenerates anytime the page is refreshed or updated. Hopefully that is enough for a demonstration,since it slipped my mind to demonstrate the risks of using the post method without the CSRF token, demonstrating the resolution without the contrast of the issue seems ineffectual.

I understand that without the csrf token, someone can send a post to your webserver from another website on the client machine if they can get the user to submit a form / click a button. The csrf provides a method of self-authentication, that the post is coming from our page not a third party.