Security is important (SSL and CSRF attacks)

Now that you're becoming a master in the way forms work, let me talk about an important topic: **security**.

The goal of this article is to teach you a little about how to secure web authentication works. And because authentication starts when a user enters their details into a browser, **form security is seriously important**.
With that said, let's take a look at 2 important things to remember when you're using forms.

SSL

When you're building any website of substance, using SSL (Secure Sockets Layer) is a MUST.

Websites that aren't secure (that don't use SSL) pass everything between the browser and the web server in **plain text**.
And this means that any user attempting to log into your website will be sending their email address and password over the public internet (through their ISP, and along the way many other ISPs) to your server.

This is bad news because a hacker could easily access the user's login request along its way to your server.

This is why SSL is important. Using SSL allows your users to *encrypt* all communication between their browser and your website in such a way that nobody else can see what's going on.

Without SSL, any type of authentication you build (and all the data that is sent to or from your website) will be insecure.

Note: in case you are wondering, setting up your site to have SSL is simple, *and once done all you have to do is route people to use HTTPS instead of HTTP.*

CSRF

This course is focused on FORMS.

And when you have forms on your site, another important security concern is **CSRF: Cross-Site Request Forgery** .

If you Google CSRF, you'll see lots of complicated definitions (sigh!).

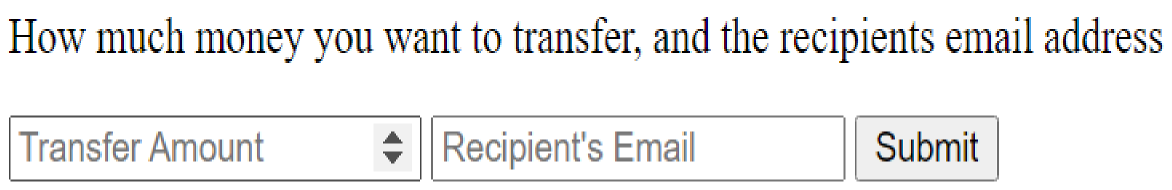
But the gist of the idea is pretty simple: it can be really easy to trick people into doing things they don't want to do. Preventing CSRF attacks means you're stopping hackers from tricking your users into doing things unintentionally.

I'll give you an example.

Let's say you're logged into your banking website, and you want to transfer money from your account to another person's account. Let's also say that the <form> to transfer your money looks like this:

```
<html>
    <head>
        <title>Transfer Money</title>
    </head>
    <body>
        <p>How much money you want to transfer, and the recipients email address</p>
        <form>
            <input type="number" name="transferAmount" placeholder="Transfer Amount">
            <input type="text" name="email" placeholder="Recipient's Email">
            <button type="submit">Submit</button>
        </form>
    </body>
</html>
```

It will then look this like:


How much money you want to transfer, and the recipients email address

If someone knows what this form looks like, they might be able to trick you into sending them money!

How can this happen?

Well, a bad guy (or girl) could send you an email with some HTML inside of it that looks like this:

```
<a
href=”https://yourbank.com/transferPage?amount=999999&email=badguyorgirl@gmail.com
”>Click here!</a>
```

If you were to click that link (and are already logged into your banking website), then you'd essentially be submitting a transfer request to transfer $999,999 from your account into badguyorgirl@gmail.com's account.

Not cool.

So how do you prevent this?

You can use a CSRF token and cookie.
The idea is pretty simple.

When a user first views the transfer page of their banking website, 2 things will happen:

1. You'll generate a long random number and insert it into the HTML form as a hidden input field (remember, we learned about `<input type="hidden'>`).
2. You'll then store that same long random number in a cookie called csrf inside of the user's browser.

When the user logs onto the transfer page, nothing will have changed, but in the source code you'll have this CSRF token in your form:

```html
<html>
    <head>
        <title>Transfer Money</title>
    </head>
    <body>
        <p>How much money you want to transfer, and the recipients email address</p>
        <form>
            <input type="number" name="transferAmount" placeholder="Transfer Amount">
            <input type="text" name="email" placeholder="Recipient's Email">
            <input type="hidden" name="csrf" value="e7s7a461k-2sde3-1234-baa2-0dcf53lauw83kd">
            <button type="submit">Submit</button>
        </form>
    </body>
</html>
```

Can you see how this is helpful?

When the user submits the form to your server, the server will check to ensure that the csrf data submitted in the form (i.e., in the hidden input field) is identical to the csrf cookie value (which is sent along to the server by the user's browser). If those two codes are different, then it means the user didn’t view the transfer form before submitting it and must have been tricked!

Pretty cool, right?

Most web frameworks have built-in or readily available tools for automatically handling things like CSRF prevention.

Being aware of how it works, however, is always good.

Conclusion

Understanding how forms work, and how they transmit sensitive information to websites is the first step in understanding web authentication.

Hope this was a fun article to read.

See you in the next lecture.