# XSRF

# XSRF

- Cross Site Request Forgery

- We have a form that sends POST requests to our server
- AJAX calls exists that can send GET/POST requests with JavaScript

- XSRF: Add JavaScript on a site that will make an AJAX request to another site on the user's behalf
- If an attacker can get someone to visit their attack site, this attack can be attempted

# XSRF - Example

- You receive an email with a suspicious link
- You click on the link to visit the page
- Page contains JavaScript that makes an AJAX request to AutoLab
  - Requests your grades
  - Makes a submission on your behalf
  - etc.
- -or- Page makes an AJAX request to your bank and transfers your funds to the attacker's account

# How do we prevent XSRF attacks?

# CORS

- Cross-Origin Resource Sharing
- A policy enforced by browsers to control cross-origin requests
- Origin is the domain of the initial request
  - eg. The page that was visited in the first GET request
- A cross-origin request is a request from a different origin
  - eg. If you add an image to your page using a full url
  - eg. Adding a JS library to your page that you don't host

# CORS

- CORS determines which cross-origin requests are allowed and which are blocked
- By default, browsers will block many cross-origin requests
- Can explicitly allow cross-origin requests with the header:

```
Access-Control-Allow-Origin: *
```

# CORS

- Since CORS is enforced by the browser, we have limited control over its enforcement
  - What if a user has a very outdated browser that doesn't implement CORS?
  - What if the user installed a plug-in that disables CORS?
  - What if the user is using an obscure browser that does not implement CORS properly?
- CORS will protect most users, but not 100%

# XSRF - Tokens

- Let's add server-side protection from XSRF attacks

- Will work for all users regardless of their browser

# XSRF - Tokens

- On the Server:
  - Generate a long random token on page load (Attacker must not be able to guess the token)
  - Embed this token in the page
  - Store this token for later use
- In the browser:
  - Token can be a hidden input on the form
  - Send this token along with form submissions
- Back to the Server on HTTP requests:
  - Check that the token is valid or reject the request
  - Can link tokens to specific users and validate the user

# XSRF Token

- Add a new input to your form for the token
- Generate and inject the token as a value using HTML templates
- Add the hidden attribute so the token is not displayed to the user

```
<form action="/image-upload" id="image-form" method="post" enctype="multipart/form-data">
    <input value="AQAAAjppCA8mhugn2UvwOTaKnVY" name="xsrf_token" hidden>
    <label for="form-file">Image: </label>
    <input id="form-file" type="file" name="upload">
    <br/>
    <label for="image-form-name">Caption: </label>
    <input id="image-form-name" type="text" name="name">
    <input type="submit" value="Submit">
</form>
```

# XSRF Token

- Read the token from the request and verify

```
POST /image-upload HTTP/1.1
Host: localhost:8000
Content-Length: 1526
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary1PtXEW2MGb29dt1C

------WebKitFormBoundary1PtXEW2MGb29dt1C
Content-Disposition: form-data; name="xsrf_token"

AQAAAjppCA8mhugn2UvwOTaKnVY
------WebKitFormBoundary1PtXEW2MGb29dt1C
Content-Disposition: form-data; name="upload"; filename="image.jpg"
Content-Type: image/jpeg

<image_bytes>
------WebKitFormBoundary1PtXEW2MGb29dt1C
Content-Disposition: form-data; name="name"

hello class
------WebKitFormBoundary1PtXEW2MGb29dt1C--
```