Week 8, Lecture 1

COMP6[48]43

Hamish Cox

Apr 8th, 2025

Week 7 Recap

- JS/HTML injection
- JS/HTML sanitisation
- Content Security Policy

Lecture 1 Overview

- Why/how could we use JS to go ‘cross-site’?
- SOP & CSRF
- CSRF Tokens
- JSONP

Wednesday: CORS

What can we do with untrusted JS?

Suppose we as an attacker have the ability to execute untrusted JS on users' browsers using some random page on the internet...

What can we do with it?

Sending Origin Requests

We have various options for sending requests to other websites:

- `fetch`
- `XMLHTTPRequest` (XHR, this was the precursor to fetch)
- `<form>` elements

What happens if our JS running on `attacker.com` sent a request to `bank.com` requesting the page containing a user's bank balance?

What if we sent a request transferring money?

Same Origin Policy

Explaining this sucks. Demo time!

Same Origin Policy

In summary:

- Cross-origin reads are blocked.
- Cross-origin writes are allowed.
- Cross-origin embeds (iframe, images, scripts) are allowed.

Cross-Site Request Forgery

This kind of cross-origin write attack using (typically) a form is known as Cross Site Request Forgery (CSRF).

What you thought it would be XSRF because Cross Site Scripting is XSS? Lmao nah.

How is the internet not broken?

Because we can just check the user opened the form before they sent the data!

Demo.

But wait...

There will always be devs wanting to circumvent security controls - they want cross origin reads for their fancy APIs!

What did that slide say earlier?

Cross-origin scripts are allowed.

Demo.

JSONP

JSONP is cursed and nobody likes it.

There is a better way to relax the SOP (CORS), but that is to talk about tomorrow.