Cross Site Attacks

Troy Hunt
troyhunt.com
@troyhunt

pluralsight
hardcore developer training

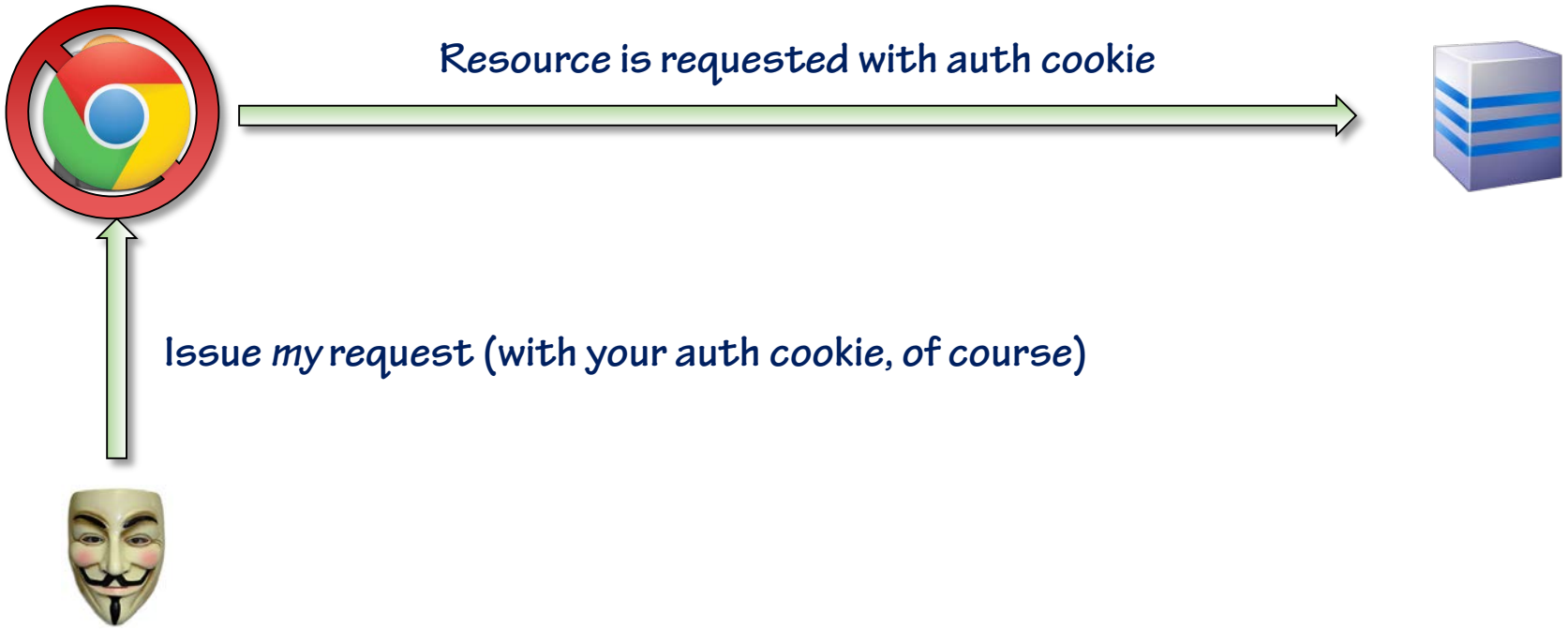
Outline

- Understanding cross site attacks
- Testing for a cross site request forgery risk
- The role of anti-forgery tokens
- Testing cross site request forgery against APIs
- Mounting a clickjacking attack

Understanding an authenticated request

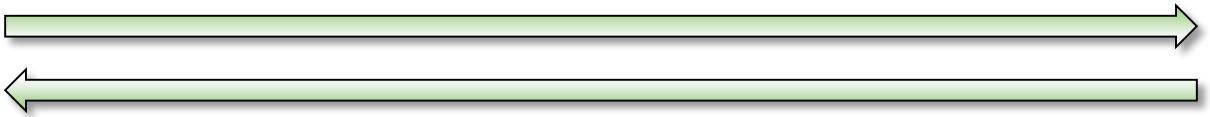
- Authentication state is usually persisted by an auth cookie
- The cookie is automatically sent to the website with each request
- The website can then identify and *authorise* the user based on the cookie
- But what if you could get the user to make a request they didn't intend to?

Anatomy of a cross site attack



Anti-forgery tokens in action

Page with a form is requested



Resultant page contains a token in a hidden field and one in a cookie
- The tokens are "paired" when issued
- They're also keyed to the current user

Browser sends back both the hidden form token and the one in the cookie

Website ensures tokens are a valid pair for the user and rejects the request if not

How anti-forgery tokens combat CSRF

- Anti-forgery tokens add randomness to the request pattern
- An attacker doesn't have the token in the hidden form field so they can't reproduce it in the forged request
- The cookie provides verification of the token from the hidden field

The X-Frame-Options (XFO) header

- **The XFO header is returned by the website and specifies how the page may be "framed"**
- **The browser then adheres to one of the following polices:**
  - Deny: no framing allowed
  - SameOrigin: only allow the page to appear in a frame on another page in the same site
  - Allow-From [URI]: only allow the page to appear in a frame on the specified site

Summary

- **Cross site attacks depend on the user's browser being tricked into issuing a request**
  - Usually while the use is authenticated
- **A CSRF attack simply creates a request of the correct structure and sends it direct from the attacker's website**
- **Anti-forgery tokens add randomness to the request**
  - They may be passed in both cookie and form or header data
  - They're keyed to the user so they can't be reused
- **APIs are at the same risk of CSRF and require the same defence**
- **A clickjack can easily circumvent CSRF protections**
  - It's also easily mitigated with an XFO header