Web Security II: Sessions and Requests, CSRF

CS1660 Introduction to Computer Security
What we know so far

• HTTP and Browsers
• Cookies (and what happens if you steal them)
• “Client-side controls”
Today

• More about requests: cross-origin/same-origin
• CSRF attacks
• Session token entropy
A generic web architecture

Client (Browser)

Browser

Renders HTML

Runs client-side code (mostly JavaScript)

HTTP GET/POST

Cookies

Web Server

APP

DB

Server-side code

- PHP, Go, Python, Node, ...

Backend
Review: Cookies

Key-value pairs (stored in browser) that keep track of certain information

- User preferences, session ID, session expiration, etc.
- Key attributes (so far):
  - **Domain**: eg. cs.brown.edu, brown.edu

A cookie’s ‘scope’
Review: Cookies

Key-value pairs (stored in browser) that keep track of certain information

- User preferences, session ID, tracking, ad networks, etc.
- Key attributes (so far):
  - **Domain**: eg. cs.brown.edu .brown.edu

When a request is made, all cookies with a matching domain are sent with it ...
...subject to certain other browser restrictions (today’s topic!)
Same origin policy (SOP): so far

- Limits how a site can set cookies*
- Limits which cookies are sent on each request

In general, “origin” must match:

https://site.example.com[:443]/some/path
Cookies: examples

- Session ID: cookie used for authentication
- App state: Shopping cart, page views
- Ad networks/tracking
- ...
Javascript

• Scripting language interpreted by browser
• Fetched as part of a page (just like HTML, images)

Capabilities
• Read/modify most page elements
  – DOM: Document Object Model
• Make requests (often asynchronously)
• Powers essentially all modern webapps
Same Origin Policy: JavaScript

• Scripts loaded from a website have restrictions on accessing content from another website (e.g., in another tab)
• All code within `<script> ... </script>` tags is restricted to the context of the embedding website
  – However, this includes embedded, external scripts
  – `<script src="http://mal.com/library.js"></script>`
  – The code from mal.com can access HTML elements and cookies on our website
  – **Notice**: Different from the SOP for third-party cookies
iframes

• Allows a website to “embed” another website’s content
• Examples:
  – YouTube video embeds
  – Embedded Panopto lectures on Canvas
• Same origin policy?

<iframe>
canvas.brown.edu
<iframe>
<iframe>
panopto.com
</iframe>
</iframe>
SOP: iframes

Only code from the same origin can access HTML elements on another site (or in an iframe).

- bank.com can access HTML elements in the iframe (and vice versa).
- evil.com cannot access HTML elements in the iframe (and vice versa).
SOP: Requests

Websites can submit requests to another site (e.g., sending a GET / POST request, image embedding, Javascript requests (XMLHttpRequest))

- Can generally embed (display in browser) cross-origin response
  - Embedding an image
  - Opening content / opening the response to a request in an iframe

- Usually can’t read (cross-origin response (i.e. via a script)
  - Sometimes websites always allow cross-origin reads
  - Why might this be bad?
Examples
What can we do with this?
Break!
Cross-Site Request Forgery (CSRF)

• Attacker’s site has script that issues a request on target site

• Example

  <form action="https://bank.com/wiretransfer" method="POST" id="rob">
  <input type="hidden" name="recipient" value="Attacker">
  <input type="hidden" name="account" value="2567">
  <input type="hidden" name="amount" value="$1000.00">
  
  document.getElementById("rob").submit();

• If user is already logged in on target site ...

• Request is executed by target site on behalf of user
  – E.g., funds are transferred from the user to the attacker
CSRF Trust Relationships

- Server trusts victim (login)
- Victim trusts attacker enough to click link/visit site
- Attacker could be a hacked legitimate site
CSRF Mitigation

• To protect against CSRF attacks, we can use a cookie in combination with a POST variable, called CSRF token
• POST variables are not available to attacker
• Server validates both cookie and CSRF token

More next class!
CSRF Demo
What We Have Learned

• Motivation and specifications for session management
• Session ID implementations
  – Cookie
  – GET variable
  – POST variable
• Cross-Site Request Forgery (CSRF) attack
• CSRF mitigation techniques