A8: Cross-site Request Forgery (CSRF)
A8: Cross-site Request Forgery (CSRF)

- XSS
  - Trick browser to execute code without user knowledge
- CSRF
  - Trick browser to access sensitive pages without user knowledge
CSRF Vulnerability Pattern

- **Problem**
  - Web browsers automatically include most credentials with each request
  - Even for requests caused by a form, script, or image from another site
- **Sites relying solely on automatic credentials are vulnerable!**
- **Automatically Provided Credentials**
  - Session cookie
  - Basic authentication header
Attacker finds function on vulnerable site he wants victim to hit while authenticated

Sets a trap via a website or e-mail

While logged into site with CSRF vulnerability
Victim views attacker site

Hidden <img> tag points to function on vulnerable site

Site with CSRF vulnerability

Vulnerable site sees legitimate request from victim and performs the action requested

<img> tag loaded by browser
Sends GET request with user credentials to site
Example

- Trick user with account at bank.cxx to visit your rogue page
  
  ```html
  <html>
  <body>
  <img src=https://www.bank.cxx/transfer_funds?amount=1000&to_account=12345678 />
  </body>
  </html>
  ```

- If user previously logged into www.bank.cxx, transfer occurs unbeknownst to user
Common CSRF activities

- Initiate transactions (transfer funds, logout user, close account)
- Access sensitive data
- Change account details
A8 - Prevention

http://www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet
Secret tokens

- Add a secret token to ALL sensitive requests
  - This makes it impossible for the attacker to spoof the request unless there’s an XSS hole in your application that leaks secret
- Tokens should be cryptographically secure random number

Examples

- Store a single token in the session and add it to all forms and links
  - Hidden Field
    ```html
    <input name="token" value="687965fdfaew87agrde" type="hidden"/>
    ```
  - Beware exposing the token in a referer header or in the clear
    - In URL: /accounts/687965fdfaew87agrde
    - In Form Token: /accounts?auth=687965fdfaew87agrde ...
- Can have a unique token for each function
  - Use a hash of function name, session id, and a secret
Server methods

- Only use HTTP GET for “safe methods”
  - Methods that have no persistent side effects on server
  - Rely upon well-formed HTTP POST requests for actions with persistent side-effects
- Require secondary authentication for sensitive functions (e.g., eTrade)
- Expire authorization cookie quickly if session is idle
Homework

- See handout