Authentication Tokens

Authentication Overview

- Registration
  - User sends username and password
  - Validate password strength
  - Store salted hash of the password
- Authentication
  - User sends username/password
  - Retrieve the stored salted hash
  - Salt and hash the provided password
  - If both salted hashes are identical, the user is authenticated

Authentication Tokens

- Need to avoid asking for username/password on every request
- When a user is authenticated, generate a random token
  - The token should have enough entropy that is cannot be guessed
  - Generally, there should be at least 2^80 unique tokens that could be generated
- Associate this token with the user

Authentication

- Once a token is generated, set it as a cookie
- Now the token will be sent with all subsequent requests
- Use the token to lookup the user
- The possession of the token verifies that this user did authenticate in the past

Cookies

- **Caution**: These tokens need to be stored on the server
- These tokens are as sensitive as passwords!
  - Stealing a token and setting a cookie with that value grants access to an account without even needing a password
- **Solution**: Only store hashes of the tokens
  - Can salt for extra security (Not necessary since the entropy is so high)

Authentication w/ Tokens

- Check each request for a cookie with a token
  - Lookup the hash of the token in the database
  - If the token is found, read the associated username
  - Proceed as though this request was made by that user
- If the token is invalid or no cookie is set
  - Redirect to the login page
- Ensure all sensitive pages/features are secured this way!
  - Remember, the front end cannot be trusted
  - A user can manually make any request