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

- With authentication, we want users to be able to:
  - Access their private data
  - Make authenticated posts to the server

- When you take these actions on a web app:
  - The app should verify that **you** made the request
  - Only serve your private data to you
  - Do not let anyone else make posts in your name

Authentication

- How does the server verify that you are the one who made a specific request?
- We only have authentication, so...
  - .. Users type their username and password with every single protected request?..
  - Authenticate using the stored [salted hash of the] password?..
  - **No**! Terrible user experience
  - You would not use this site

Authentication

- How does the server verify that you are the one who made a specific request?
- We only have authentication, so...
  - .. Store the username/password in cookies and send them on every request?..
  - Password is stored client-side in plain text
  - **No**! Never store passwords in plain text
    - Not even client-side! Don't do it

Authentication Tokens

- Instead of authenticating the username/password on every request:
  - **Issue an authentication token**
- When a user is authenticated, generate a random authentication token
- Store this token in your database and mark the username who authenticated
- Set the token as a cookie for that user
- Whenever a request come with that token, treat them as the user associated with that token

Authentication Tokens

- With authentication tokens:
  - We have a login system 🎉
- As long as the user has the authentication token as their cookie, they are logged in*
  - All their requests are authenticated by the server

*You can/should set these tokens to expire either client-side (cookie expiration), server-side (storing an expiration timestamp in your database and ignoring the token after that timestamp), or both

Authentication Tokens

- Authentication tokens need to be random
  - The token must have enough entropy that is cannot be guessed
  - Eg. An attacker should not be able to send requests with random tokens until one matches a logged in user
- Generally, there should be at least 2^80 unique tokens that could be generated (80 bits of entropy)
  - More is better!

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

Storing Authentication Tokens

- **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/paranoia (Not necessary since the entropy is so high)
  - Salting also makes DB lookups more difficult

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
  - Do not respect the request and return a 400-level response code:
    - 401 Unauthorized - User is not logged in
    - 403 Forbidden - User in logged in, but trying to something they are not allowed to do
- Ensure all sensitive pages/features are secured this way!
  - The front end cannot be trusted (NEVER trust your users)

Cookie Hijacking

- We're now using cookies for authentication
- The possession of the token verifies that this user did authenticate in the past

- What if someone steals your cookies?
  - They can authenticate as you without needing your password

Demos