CS 410: Web Security
A2: Labs, Homework, and Program

WFP2: Authentication

- **Example #1**
  - **Default usernames and passwords are often left unchanged for many network devices and services.**
  - **This admin username and password is trivially guessed.**
- **Example #3**
  - **Cookies are often used as an authentication token that validates a client has authenticated in the past**
  - **Use your browser to reverse-engineer the cookie being used and write a Python script to obtain admin access to the site.**
- **Example #4**
  - **To hide the format of the cookie, cryptographic hash functions are sometimes employed. Weak hash functions such as md5, however, are easily brute-forced and several sites currently provide hash lookups that produce plaintext**
  - **Reverse-engineer the cookie format and write a Python program that sends an admin cookie to obtain admin access to the site.**
- **Example #5**
  - **Mismatches between the web application and backend databases can cause security errors**
  - **Case-sensitivity is one such conflict**
  - **The page is case-sensitive to usernames, but the database is not**
  - **Use this to register an admin user**
- **Example #6**
  - **Another mismatch is the treatment of whitespace between the web application and backend database**
  - **Use this to register an admin user**

Homework

- **Lessons: Session Management**
- **Challenges: Session Management Challenges #1-6**

Program #2 (WFP2: Authentication #2)

- **The authentication routine leaks timing information that allows adversary to guess characters of both the username and password**
- **Assuming the username is 'hacker', write a Python program that uses the vulnerability to automatically determine the password**
  - **You may either use Python's timing facility or the timing information in the requests library**
  - **To shorten the run-time of your program, on your WFP2 instance, edit the credentials in `/var/www/authentication/example2.rb`**
  - **Then do**
    ```
    sudo service apache2 restart
    ```
  - **Note that the username and passwords I will be testing your program on will be alpha-numeric**
- **Rubric**
  - **Your program must take a single argument from the command line (`sys.argv[1]`) that represents the IP address or name of `<wfp2_site>`**
    - **(e.g. python3 program2.py wfp.oregonctf.org)**
  - **Your program should be robust against spurious delay spikes. For example, taking the best candidates of a round and rechecking them to find the correct character is an excellent strategy.**
  - **Your program should be concise and modular**
  - **Your program should check for errors such as missing arguments or HTTP errors**
  - **Your program should include some code documentation via Python docstrings**