Web Challenges

Log Me In

- can log in as admin with `username: admin; password: admin` although this leads to nothing useful
- "routes.py" shows that our password, regardless of login or register, is made in this order:
  - .strip() - all whitespace is removed
  - .encode() - UTF-8 encoded
  - sha256 - hashed for security
  - .hexdigest() - refer to hashlib library docu
- Accounts with `uid=0` can see the flag in the message portion of the "user panel"
  - "models.py" shows all admin are `uid=0` while users are `uid=1`
- "user.html" has templates but anything inserted is filtered
- So the problems are:
  - every field gets checked for alphanumericness
  - The template is seemingly inaccessible due to the above ^ check
  - any account created using "register.html" has its `uid=1` and this is hardcoded behavior
- **Solution:** We have the cookie and we know exactly what it's based off of
  - cookie = encoding(our username, displayname, & uid)
  - our "uid" as a user is naturally set to 1
  - So in this case you can XOR the token, "info", with the plaintext to find the key
  - Once you have the key, you can construct your own
  - One of those "1"s corresponds to the `uid` value so change that and we will have elevated access to see the flag
- **Solution #2(way easier one):** We know the original, "valid", cookie made for our account has `uid=1`
  - can just change the value of only the `uid` value's index to be `uid=0` to get elevated access
  - No extra work involved recreating the key or the token
  - reason being the uid is hardcoded to be "1" with no pseudorandomness
  - "keychars ⊕ 1 = whatever the 2nd to last hex-digits are"
  - just flip the least-sig-fig bit in the account at the `uid` index to change it from "1 --> 0"
  - example:
    - token = 48674c3731025651282f614a4d54053d0a1a5e4760050a04075b080d27700953454a021f6567585a040210787550**43**05
    - `0x43 = b'01000011'`, so flip the right-most bit to make `uid=0`
    - `0x42 = b'01000010'` , once decoded again makes `uid=0`
    - new token = 48674c3731025651282f614a4d54053d0a1a5e4760050a04075b080d27700953454a021f6567585a040210787550**42**05
  - All we did was take the token, find where the `uid` value is, flip the least-sigfig bit, so that when ⊕ed it becomes "1"