ITIS 5166 Network-Based Application Development

Password Security

Password Security

- Store users' passwords as plaintext is not secure
- If an attacker compromises the database, then users' passwords are compromised
- The best practice is to encrypt and/or hash a password before storing it in the database
- In this class, we will use Bcrypt, which is designed for securing password

Bcrypt

- Bcrypt is a slow algorithm, thus, it reduces the number of passwords by second an attacker could hash when crafting a dictionary attack
- For each plaintext, it appends a random string called salt at the end
- Then, it hashes the password and the salt to create a hashed password
- The salt is included in the output string

$2b$10$ 7oN29jOP3MVI0zUv1MXJ6O krepXA8v2XRS4mFvFdq06zvWNZSuysq

Version, cost salt hash

Bcrypt

- In NodeJS ecosystem, there is a 3rd party module that implements the Bcrypt algorithm
- To hash a password, call
  ```
  bcrypt.hash(myPlaintextPassword, saltRounds);
  ```
- To check a password, call
  ```
   bcrypt.compare(myPlaintextPassword, hash)
  ```
- Both are asynchronous functions calls

Bcrypt module documentation

Signup and Login with Bcrypt

- When a user creates an account, hash the password and store the hashed password in the database
- When the user attempts to login, hash the password they entered and compare it to the hashed password in the database
- If the hashes match, the user is authenticated, otherwise, the user is not authenticated

Demo: Bcrypt