Class 16 Basic Authorization

Objective

- Students will learn about cryptographic hash and cypher algorithms
- Students will be able to model a User and safely store their sensitive data
- Students will be able to implement a Basic Authorization parser

Hash Algorithms

- One-way street
- Takes data as input and returns a long number
- Impossible to reverse the output number into the original data
- The same data will always produce the same long number
- Used to verify the integrity of data

Cypher Algorithms

- Two-way street
- Encrypts data, but provides a reversible way to decrypt it

Never seriously try to make you're own cypher algorithm. People have spent years and years making and breaking cypher algorithms.

- Read: 5 Common Encryption Algorithms

ROT-13 Algorithm

A simple cypher algorithm. Rotate each letter 13 spaces in the alphabet. The encode method is the same as the decode method.

- A becomes N
- B becomes O
- ...
- N becomes A
- O becomes B
- Wikipedia: ROT13
- ROT13 Tool

The Authorization Header

There's a specification that dictactes how users can supply usernames and passwords to log in to system via HTTP. Users can use a special HTTP header `Authorization`.

It looks like this:

```
Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l
```

The string `QWxhZGRpbjpPcGVuU2VzYW1l` represents an encoded form of a username and password.

The specification says to take the username `Aladdin` and the password `OpenSesame` and put them together with a colon, like `Aladdin:OpenSesame` then it runs that string through Base64 encoding.

Base-64 Encoding

Base64 encoding is a common encoding. It is easy to encode and decode information. Encoding usernames and passwords with Base64 does not protect any secrets. It only makes for a good standard for the specification.

- Wikipedia Base64 Encoding

JavaScript has two Base64 encode and decode methods built into the language:

```
let username = 'Aladdin';
let password = 'OpenSesame';
let userInfo = username + ':' + password;

let encoded = btoa(userInfo);
let decoded = atob(encoded);
```

Accessing the Authorization Header

- Use `req.get('Authorization')` to read the value of the `Authorization` header off the HTTP request.
- Use `Buffer.from(someString, 'base64').toString()` to convert a Base64-encoded string into it's original value.

```
router.get('/signin', (req, res) => {
  let authHeader = req.get('Authorization')
  let payload = authHeader.split('Basic ')[1]
  let decoded = Buffer.from(payload, 'base64').toString()
  let [username, password] = decoded.split(':')
}
```

bcrypt

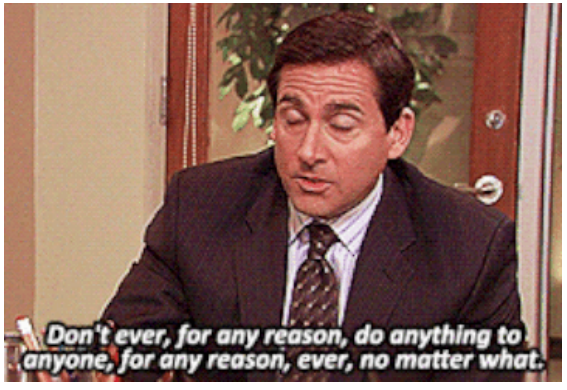
- `bcrypt` is a useful NPM package.
- Use `.hash(password, numRounds)` to encrypt a password.
- Use `.compare(password, hash)` to see if a password matches a hash

```
const bcrypt = require('bcrypt');

const password = 'elephant'
const rounds = 10;
bcrypt.hash(password, rounds, (err, hash) => {
  bcrypt.compare(password, hash, (err, isValid) => {
    console.log('Password:', password);
    console.log('    Hash:', hash);
    console.log('  Match?:', isValid);
  });
});
```

Saving Hashed User Passwords

- Never store plain-text passwords!
- Only store the hash of the password.



Saving Hashed User Passwords

```
// modify the 'save' pre hook on a mongoose User schema
User.pre('save', function(next) {
  // hash the password the first time it's saved
  if (this.isNew) {
    bcrypt.hash(this.password, 10, (err, hash) => {
      this.password = hash
      this.passwordHash = hash
      next();
    });
  }
});
```

Notice: we must use an old-school ES5 function here to get the proper context of this with `this.isNew` attached Trust me, a `() => {}` arrow function won't work here.

Checking Passwords

Attach a new method to the User Schema so we can easily check passwords later.

```
User.checkPassword = function(attempt) {
  return new Promise((resolve, reject) => {
    bcrypt.compare(attempt, this.password, (err, valid) => {
      if (err) {
        reject(err)
      }
      resolve(valid)
    })
  })
}
```

Protecting a Route

- Protect any route by tying all the pieces together.
- Access the `Authorization` header
- Decode the Base64-encoded string
- Access the username and passwaord
- Query your database to find a user with that username
- Use bcrypt to compare the password they just send with the stored hashed password
- Return and send `200` or `400` status codes as appropriate.

Check for Understanding

What's the difference between a hash and a cipher?

Check for Understanding

What's the difference between a hash and a cipher?

- A hash is irreversible.
- A hash is used to verify data-consistency.

- A cipher is reversible.
- A cipher can be encoded and decoded.
- It should be hard to reverse data encoded with a cipher.

Check for Understanding

What's the best cipher algorithm in the world?

Check for Understanding

What's the best cipher algorithm in the world?

- Whichever one you didn't write.
- There are multiple good implementations.
- Some have been proven non-useful over time
  - cyptographic researchers discovered fundamental flaws
  - governments put in backdoors

Bruce Schneier is a popular, credible security researcher. Read his blog if you want to inform yourself more about what happens in the security world.

- The SHAppening
- Bruce Schneir: Can the NSA break AES?
- Bruce Schneir: Refuse to be Terrorized
- Bruce Schneir: In Praise of Security Theater