Secure web apps

• Security is a primary concern for anyone developing web applications.
• Data access must be controlled, passwords must be validated securely, and users just be able to trust the information presented to them.
• Complete security is very hard to achieve and beyond the scope of this unit, but basic authentication is relatively easy.
• An interesting case study of internet security is anonymous’ attack on HBGary:
Adding authentication to MEAN apps

- There are some useful packages to handle authentication. Passport is a popular option with many strategies, including authentication through Google and Facebook.
- We’ll look at using the passport-local which stores the authentication details in a local mongo database.
- This is taken from the web tutorial http://mherman.org/blog/2015/01/31/local-authentication-with-passport-and-express-4
Elements of Web-security

- Web security depends on trust. There are several elements to this:
  1. The web server needs to be confident that someone accessing data is authorised.
  2. The user needs to know that the site they are visiting is the one they intend to.
  3. Both the server and the client need to be confident that no one in the middle is accessing unauthorised data.
- 2 is typically handled by browsers, and 3 is achieved with https. We’ll focus on 1.
Authentication Strategy

- To track a user's identity we need to have them register so we can associate a user name with them.
- When someone uses an application a session is maintained via a variable held by the web-browser.
- When someone logs in they provide a password. This is salted and hashed to provide a digest which can be compared to a hash in a database (keeping the password secure).
- Once the user is authenticated, they will be served their requested pages, and their ID will be a parameter of the requests.
Steps to implement security:

- Install the npm packages: passport-local, express-session and passport-local-mongoose
- In app.js we need to include and initialise passport.
- The express-session stores data in the backend away from the user.
A model for user accounts:

- We can create an account model using mongoose. Including passport-local-mongoose will take care of hashing and salting:

```javascript
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var passportLocalMongoose = require('passport-local-mongoose');

var Account = new Schema({
    username: String,
    password: String
});

Account.plugin(passportLocalMongoose);

module.exports = mongoose.model('Account', Account);
```
Addings Routes

- The login, register and index pages all require get and post actions so these need to be added to the routes.
- A controller should really be used to group the call-backs in a single js file.
Jade views

- The Jade views can now be rendered differently depending on whether a user is logged in:
Login and register pages

- These can now be implemented as simple forms

```html
extends layout
block content .container h1 Login Page p Login to the simple app here. br form(action="/login", method="post") label(for='username') Enter username: input(id='username', type='text', name='username') label(for='password') Enter password: input(id='password', type='password', name='password') input(type='submit', value='Submit')

extends layout block content .container h1 Registration Page p Register here br form(action="/register", method="post") label(for='username') Enter Username: input(id='username', type='text', name='username') label(for='password') Enter Password input(id='password', type='password', name='password') input(type='submit', value='Register')
```
Registration Page

Register here

Enter Username: bill Enter Password: ***** Register

Simple

Login
Register

First Post!
5
6
7
8
9
better
hello!
d ok here
ladjb
Hello

Register

You are currently logged in as Fred

Logout
Name Age Email Delete?
Bob 39 bob@bob Delete
Max 16 max@max Delete
tim 37 tim@mail Delete
Bill 27 bill@bill Delete

Name Age Email
You can see how the account data is held via the Mongo Shell:

```bash
> db.accounts.find().limit(1).pretty()
{
   "_id": ObjectId("5743f2ce76bb25692994956f"),
   "salt": "bbfc4fa4df70b775932688828b825e57a153235952d19f04770e833e3e3f4c76e",
   "hash": "6c5a4309693301f2e532e2cc67f44617af15fab1fddf15a5dc99d90486c95fd730c6af1b5f4b2295ca1ff3a3f99530ca68a5cc6cf3ab36f36bda1871f16fd1bf602a4603c2191251d114dbad6513545f2f5b627a48af7ef38ddf2f038a690da6b72603a0a4dd114fa8b35ff19de663100c6ebeceb976eb47c8926b2d282d65bc433f33232a40b6819eae2d67b0cd9d630f5723b2bacc37677b673636ecc05a1347b00817d6f0d94bc00ddab3668570b89db2ec00a9f642adc8a7054a18e66981a9997358133ad4b7238030492dca793016d98ffbd0ef78fb60b3a58627eca59c976dc85360849acbd226deeb270ce1ae9e815f5f7a5716e982a14240965ceae6bb52ba4098b2b65803a260f26c9e5d35eb931ff7bf7e13659c980297287f3c1a67120830f6c11b49169854286c95c4371e0b9d47b273232120af489ee83eac56834db62bc6262253ce495c3d77eb7757267c4b271f6c6f2515264d7dc614eb158f166dccc002e8ec27cf5910996010b66cf7bdc6d28818c11f0f2e32e9061cb7198a7e4d242d0607bc041b2ac8e1a244bd1da6018864e7340f7915a8f4568142a3b59b2c2d6c3c41288a99028f5c671575ff7911306da71d75c4048c631f1c2ae5e43cf24f6ab35895101e56d55ba3b56b15e90dd6ef6ef4ddad4b3936075bd0c482605e80ac1217895438918c3140a0d102f4f5314d9593d9a93",
   "username": "Tim",
   "__v": 0
}
```
Next lecture....

- Testing and Agile development....