School of Computer Science

COMP SCI 2207/7207 Web and Database Computing
Lecture 24 – Session Login, and 3rd Party Auth with OpenID

adelaide.edu.au

seek LIGHT

Some discussion on your Data Plan

Login Basics

AJAX Site Authentication Workflow

1) Client sends AJAX request for resource.
2) Authenticate & Authorise Request.
   - Does the user have a valid session?
   - Is the user allowed to access the resource?
3) If okay, send 200 with requested resource (DONE)
4) If NOT okay, send a 401/403
   - Client displays login dialog or redirects to login page
5) Client sends credentials
6) If okay, create session and link to user
   - Client may re-send original AJAX request

AJAX Site Authentication Tips

- Initialise sessions by default
- Use Middleware
  - Separate your routes into public vs user data
  - Add middleware to authenticate and authorise all requests to the user routes
- Perform all redirects on client (because AJAX)

Website login & Security Challenges

- Can be difficult to implement well.
  - Very easy to make security mistakes.
- Risk of compromise
  - If compromised users may abandon your service or worse…
- Users commonly have poor security practices
  - Users often use the same password multiple places.
- Users don't like having to create new accounts for every site they visit

The 1st Rule of Security Programming

Don't implement your own security

Get someone who knows what they're doing to do it

Introducing OpenID Connect

From openid.net:

- “OpenID Connect is an interoperable authentication protocol based on the OAuth 2.0 family of specifications.”
- “Lets app and site developers authenticate users without taking on the responsibility of storing and managing passwords in the face of an Internet that is well-populated with people trying to compromise your users’ accounts for their own gain.”

In a nutshell:

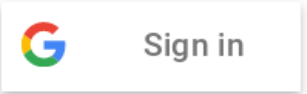
Use a trusted 3rd Party (Identity Provider)

- User clicks OpenID button for their Identity Provider.
- User logs in to Identity Provider.
- Identity provider verifies user information for our web application.
- Web application matches identity info to user's account

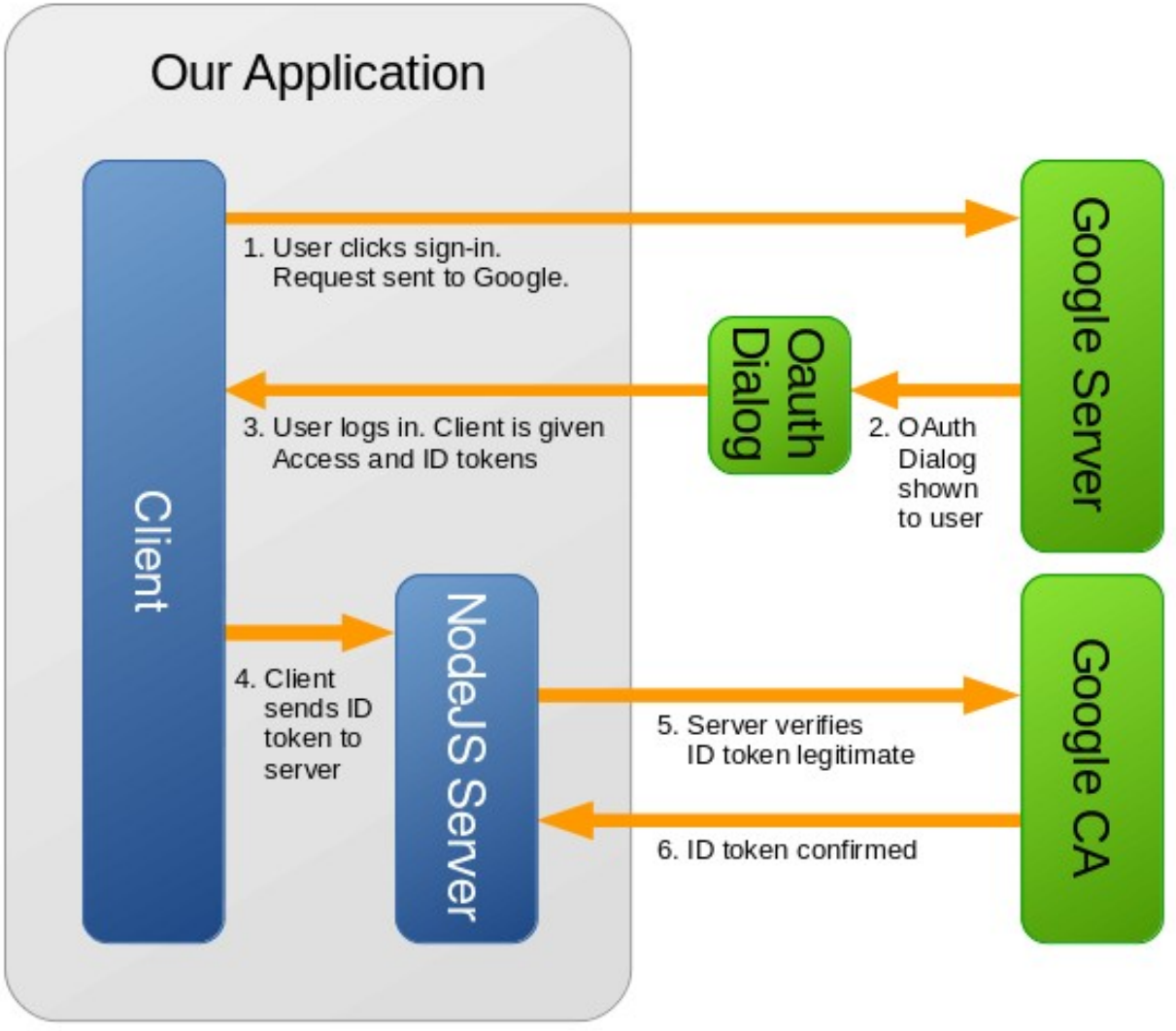
You can use any Identity Provider(s) or any OpenID Libraries

for your project

But today we're using Google's implementation



More detail:


Our Application
Client
NodeJS Server
Google Server
Oauth Dialog
Google CA
1. User clicks sign-in.
Request sent to Google.
2. OAuth
Dialog
shown
to user
3. User logs in. Client is given
Access and ID tokens
4. Client
sends ID
token to
server
5. Server verifies
ID token legitimate
6. ID token confirmed

Understanding OpenID

OpenID relies on the client sending our server a token from the Identity Provider. How do we verify the legitimacy of the Token?

- The token is encrypted using Public Key Cryptography.
  - Encryption is done using the provider's private key.
  - We can decrypt the token with the provider's public key.
  - Decryption will only work if the provider performed the encryption i.e. token unmodified
- If the token is verified, then the user data inside is legitimate and we can authenticate the user.

What next?

- Use the identity information provided to match user's info to their user account on the web app.
- If they don't have an account, generate one.
- Perform standard login actions such as grant session token access.

Demo


THE UNIVERSITY
of ADELAIDE

What's happening?

Due

- Prac Exercise 7 – due Wednesday
- Group Project Milestone 1 – due Friday

This week

- Workshops
- Livestream Wednesday night
- Server review
- Integrating Databases