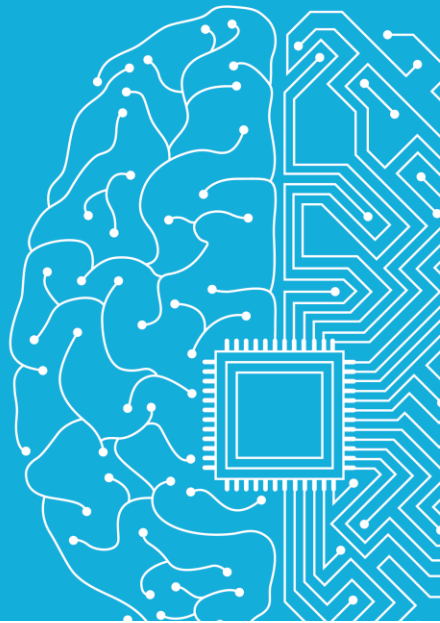
Securing your Web APIs with Azure AD



architech

Agenda

1. What is Azure Active Directory?
2. Identity Management & Authentication Scenarios
3. Oauth 2.0 & OpenID Connect
4. Integrating with Azure AD
5. Demo

What is Azure Active Directory?

- Cloud based IdMaaS. Provides authentication & authorization services to applications and services
- Supports multiple standard protocols:
  - Oauth 2.0
  - OpenID Connect (authentication layer above Oauth 2.0 authorization)
  - SAML
  - WS-Federation
- With Azure AD, you can easily add SSO to any application that is registered in a given Azure AD tenant
- Support for single tenant, multi-tenant and B2C scenarios (e.g. the user identity is in another OAuth provider such as Google, Facebook, etc)

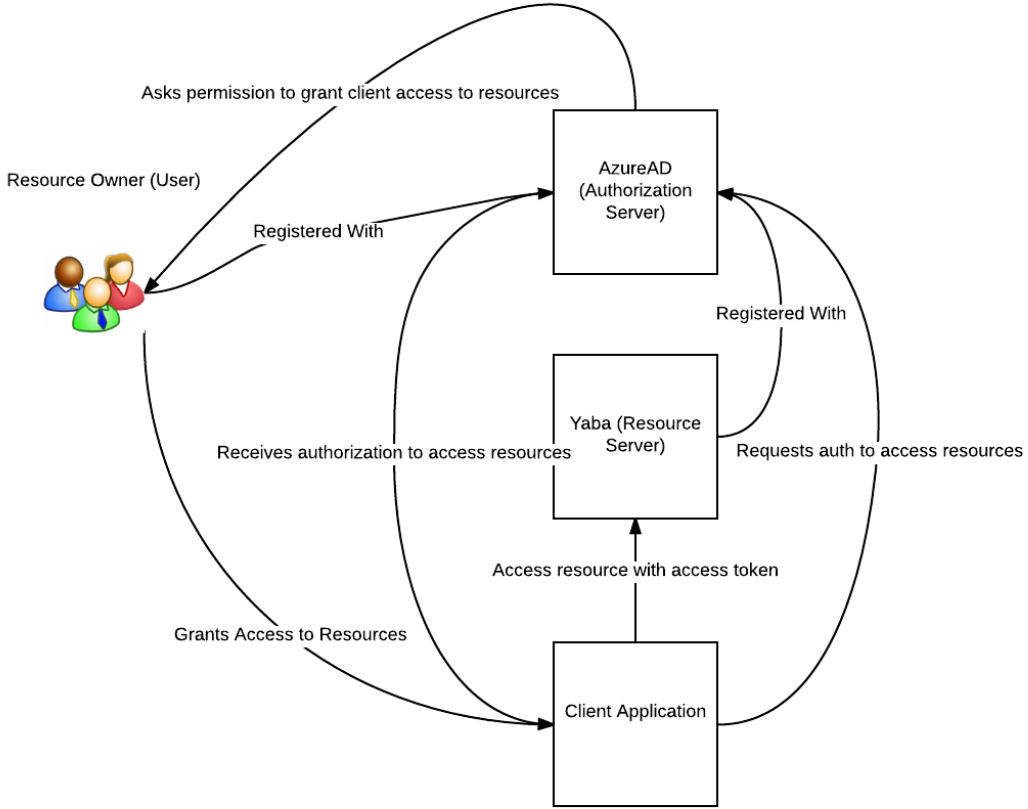
Benefits

- Better UX as the user doesn’t have to remember yet another set of credentials
- More control to users as they can revoke access without changing credentials
- Easier user administration as you can add/remove users in one place
- Support of multiple application architecture scenarios:
  - Web application with a server-side component
  - Native mobile applications that directly access protected APIs
  - SPA web applications that directly access protected APIs
  - Application or service to API

OAuth 2.0 & OpenID Connect

- OAuth 2.0 is an authorization protocol. Highly extensible through ‘grant types’
- OAuth 2.0 answers the question, “what can the user do”?
- Does not know who you are.
- OpenID Connect is the authentication/identity protocol that leverages OAuth 2.0 authorization grants to handle authentication and identity.
- OAuth 2.0 defines an ‘access token’
- OpenID Connect defines an ‘id token’
- These tokens distributed as JSON and are signed by the issuer (e.g. Azure AD)

High-Level Roles and Interactions

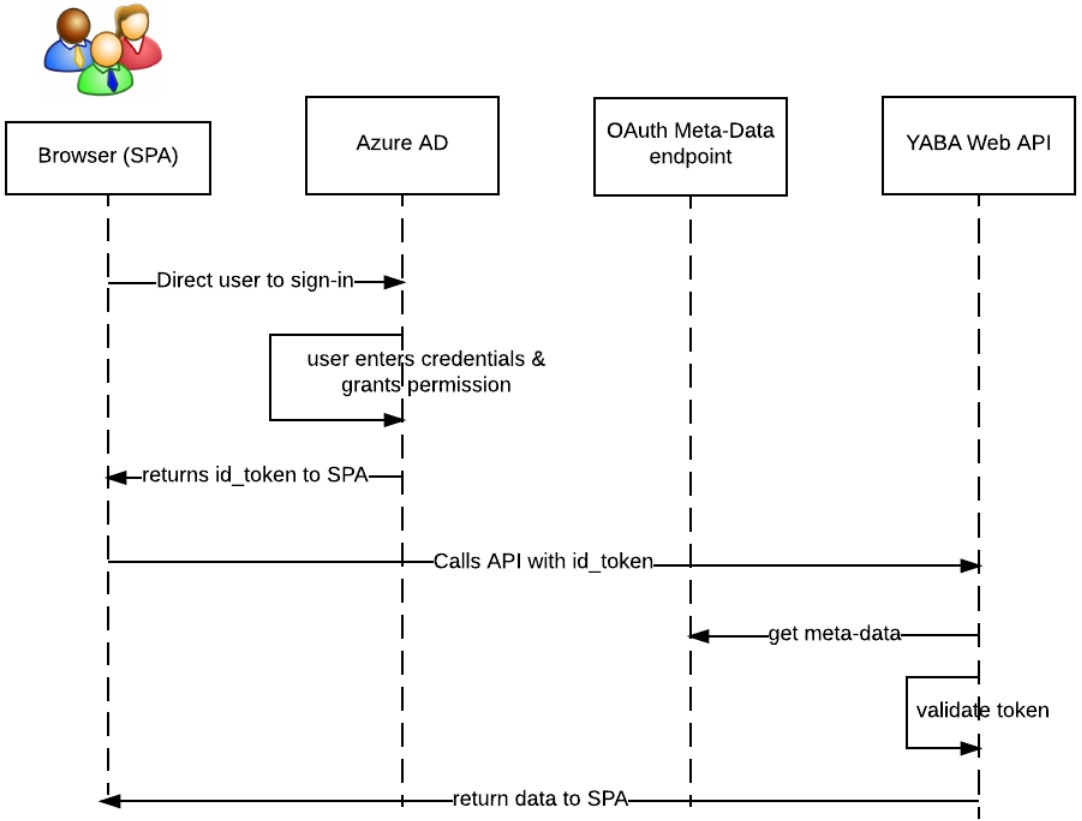


OAuth 2.0 Grant Types

- The OAuth 2.0 specification defines multiple ‘grant types’
- Grant types are essentially authorization flows
- Which grant type you use depends on the architecture of the client that needs authorized access to the API, specifically the ‘trust level’ of the client
- Authorization Code Grant – used by clients with high-level of trust e.g. can securely store client secrets. E.g. Web applications with a server side component that calls the APIs
- Implicit Grant – used by untrusted clients e.g. SPA apps that calls the API directly from the browser and does not have a means to securely store client secrets
- Resource Owner Password Credentials Grant – used with trusted applications e.g. you built both the API and the client app
- Client Credentials Grant – applications without an user that needs access to APIs e.g. a batch job, etc

OpenID Connect Implicit Grant Flow

- In our case, we have a SPA app, so we are going to leverage the ‘implicit grant’ flow



References

- Azure AD OpenID Connect Flow - https://docs.microsoft.com/en-us/azure/activedirectory/develop/active-directory-protocols-openid-connect-code
- OAuth2 Flow on Azure AD - https://docs.microsoft.com/en-us/azure/activedirectory/develop/active-directory-protocols-oauth-code
- SPAs using the Implicit Flow - https://docs.microsoft.com/en-us/azure/activedirectory/develop/active-directory-v2-protocols-implicit
- Azure AD Developer Guide - https://docs.microsoft.com/en-ca/azure/activedirectory/develop/active-directory-developers-guide
- OAuth2 Grant Types - http://oauthlib.readthedocs.io/en/latest/oauth2/grants/grants.html
- Azure AD Tokens - https://docs.microsoft.com/en-us/azure/activedirectory/develop/active-directory-token-and-claims
- Azure AD B2C - https://docs.microsoft.com/en-us/azure/active-directory-b2c/activedirectory-b2c-overview
- OAuth 2.0 Specification - https://tools.ietf.org/html/rfc6749