Modern Identity Management Patterns for Microservices and Mobile

Mark Perry
APAC CTO
Ping Identity
@markperryAU
IT’S A NEW WORLD

Everything’s Mobile
IT’S A NEW WORLD

Agile Backend Integration using Microservices

Image: http://ryanjbaxter.com
THEN WHY IS IT SO HARD?

**Answer:** Fragmented silos built over time

- **Separate Login**
  - Web E-Commerce (Added in 2013)
  - Marketing Programs (Added in 2008)
  - User Profiles
  - Marketing Automation
  - Partner Customer Loyalty Program (Added in 2010)
  - Separate Data Silos

- **Separate Login**
  - Website (Launched in 2005)

- **Separate Login**
  - Mobile App (Launched in 2012)
  - Separate Data Silos

- Incompatible standards
COMMON ARCHITECTURAL PROBLEMS

- Multiple Data Stores
  - Identity data is siloed — No Single View of Identity

- Applications built in isolation
  - Inconsistent authentication experiences
  - No SSO, SLO, or identity data sharing — Bad customer experience

- Home-grown SSO, or old, unsupported framework(s) used
  - Security issues, hard to maintain

- Scalability issues

- Architecture doesn’t work well in a cloud-hosted environment
RESULT: DISJOINTED EXPERIENCES

Why do I have to manually manage multiple passwords for one company?

You’re sending me email every other day! All I want is your monthly coupons.

You still have my old account info. I corrected that earlier, why didn’t you see that?

Your marketing partners keep contacting me. How can I make that stop?

Ping Identity
Modern IAM Patterns using REST/JSON
- Authentication
- SSO (web and native mobile)
- Multi-Factor Authentication
- User Authorisation
- Transaction/API authorisation

Security
- Prevent Man-In-The-Middle attacks
  - PKCE
  - Token Binding (coming soon!)
Patterns — Mobile
Patterns — Mobile

Authentication/SSO

- Common libraries to standardise the routine of acting as an OAuth client
- Developed by the industry
  - Open source SDK maintained by OpenID Foundation
- Enables SSO across multiple native apps
- Uses native controls to share authentication state
  - IOS: SFSafariViewController (IOS 9 and above)
  - Android: Custom Tabs (Jellybean and above)
  - Falls back to native WebView for older OS versions
AppAuth Flow Example

Native Mobile App

User Agent

OpenID Connect Provider

Data Store (i.e. AD)

1. Present SFSafariViewController with SSO URL
2. Redirect to CP for user authentication
3. Challenge user for authentication
4. Receive credentials from user
5. Validate credentials
6. Repeat for additional authentication mechanisms (i.e. 2FA)

Set authentication session cookie
Generate authorization code

POST https://example.com/auth/token换取cesss_token HTTP/1.1
client_id=0ef55ef5e5
client_secret=secret
redirect_uri=https://example.com/callback
grant_type=authorization_code
access_token=xyz1234567890

Return tokens to application (in POST response)

Validate id_token
Request UserInfo
Retrieve User Profile

Note: The end-user authentication method can be any supported method from the identity provider (such as a SAML flow or X509 certificate) - this example uses a HTML form.

Generate id_token, access_token & (optional) refresh_token

Ping Identity

RSA Conference 2017 Singapore
FIDO

- Industry specification for strong authentication
- Local device authentication
  - Based on public key cryptography
  - Protects against phishing, man-in-the-middle and replay attacks
  - Biometrics, if used, never leave device
  - No server-side shared secrets to steal
- FIDO 2 — Coming Soon!
  — W3C Web Authentication API
Patterns — Mobile

Multi-Factor Authentication (MFA)

- Define authentication flow in the Authorisation Server
  - Simplifies the app’s code
  - Potentially different authentication for each transaction
    - e.g. Login, change password, transfer money, change shipping address, etc.
  - Easily modify authentication flows without code changes
    - Add new MFA methods (e.g. facial recognition, voice biometrics)
    - Respond to security threats quickly
    - No need to support authentication methods encoded in previous app versions
- Make use of contextual data to trigger authentication methods
  - Device OS, type, geolocation, new device, IP address, jailbroken/rooted status, etc.
PKCE

- Proof Key for Code Exchange (PKCE), pronounced “pixy”
- RFC 7636
- Helps mitigate attacks on requests for OAuth tokens
- A dynamically created cryptographically random key called "code verifier" is created for every authorisation request
- The server compares it with the previously received request code so that it can perform the proof of possession of the “code verifier” by the client.
Authorisation Intercept Code Attack

Figure 1: Authorization Code Interception Attack
PKCE Mitigation

Image: https://pfelix.wordpress.com/2016/02/15/oauth-2-0-and-pkce/
What’s Coming — Token Binding

OAuth 2.0 Token Binding
draft-ietf-oauth-token-binding-03

Abstract

This specification enables OAuth 2.0 implementations to apply Token Binding to Access Tokens, Authorization Codes, and Refresh Tokens. This cryptographically binds these tokens to a client's Token Binding key pair, possession of which is proven on the TLS connections over which the tokens are intended to be used. This use of Token Binding protects these tokens from man-in-the-middle and token export and replay attacks.

- Currently (as of 3 July 2017) a draft RFC
- Prevent Man In the Middle Attacks
- Prevent token export/replay attacks
Token Binding Example (mobile app)

**Code Challenge**

```
GET /as/authorization.oauth2?response_type=code
&client_id=example-native-client-id&state=OUC2jyYtzRCrMyWrVnGj
&code_challenge=rBlgOyMY4teiuJMDgOwkrpsAjPyI07D2WsEM-dng6eE
&code_challenge_method=TB-S256 HTTP/1.1
Host: server.example.com
```

**Code Verifier**

```
POST /as/token.oauth2 HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

Sec-Token-Binding: AIkAAgBBQEOO9GRFP-LM0hoWw6-2i318BuuuUum5AL8bt1sz
lrlEFfp5DMXMNW3O8WjcIKr2DKJnI4xnuGsE6GyQd9Rbd0AQJD3xyo9PBxj8M6Y
jLt-60axqDkyoBoTkyrnNbLc8tJQ0JtXomKzBbj5qPtHDduXc6xz_1zvNpxSPxi42
8m7wkAAA

grant_type=authorization_code&code=mJAReTWKX7zI3oHUNd4o3PeNqNqxKGp6
&code_verifier=provided tabIndex=client_id=example-native-client-id
```

Figure 14: Authorization Request with PKCE Challenge

Figure 15: Token Request with PKCE Verifier
Patterns — Microservices
Patterns — Microservices

- Use OAuth2 with an API Gateway
- Avoid coding security into microservices
- Limit the permission of the microservice API call
- Ideally, API Gateway should use the same authorisation policies as web traffic
- Use OAuth Token Exchange to ensure security
  - Don’t replay user tokens down the stack
  - Request new scopes as required by the transaction
    — Downscoping
Microservices

- Token exchange is currently a manual process
  - Using OAuth Resource Server (API Gateway)
- Coming Soon... OAuth Token Exchange protocol
  - Draft as of 3 July 2017, version 9

OAuth 2.0 Token Exchange
draft-ietf-oauth-token-exchange-09

Abstract

This specification defines a protocol for an HTTP- and JSON- based Security Token Service (STS) by defining how to request and obtain security tokens from OAuth 2.0 authorization servers, including security tokens employing impersonation and delegation.
Outcomes

- Modern, flexible architecture
- Stateless identity backend
  - Removes the need for session databases/replication
  - Works well with PaaS/Cloud hosting environments
- Easily add or change authentication methods without code changes
  - Respond to business needs and competitive pressures more quickly
- Improved security
Next week you should:
- Educate your development and deployment teams on OAuth 2, AppAuth, MFA.
- Review the use of Identity in your mobile apps and microservices.

In the first three months following this presentation you should:
- Deploy a modern Identity Provider/OAuth 2 Authorisation Server in your dev environments and get up to speed!
- Review future plans for Identity.

Within six months you should:
- Be ready to update production apps and environments with these patterns.
- Start enabling better user experience and security!
Resources

- OAuth 2 / OpenID Connect / JWT

- AppAuth
  - https://github.com/openid?query=AppAuth

- FIDO
  - https://fidoalliance.org

- OAuth Token Binding

- OAuth Token Exchange
Thank You

@markperryAU
www.pingidentity.com