Securing your APIs on the Cloud

Toronto

Hugo Guerrero
APIs & Messaging Developer Advocate
hguerrero@redhat.com
@hguerreroo

Why?

Businesses use APIs to connect services and to transfer data

Broken, exposed, or hacked APIs are behind major data breaches. They expose sensitive medical, financial, and personal data for public consumption.

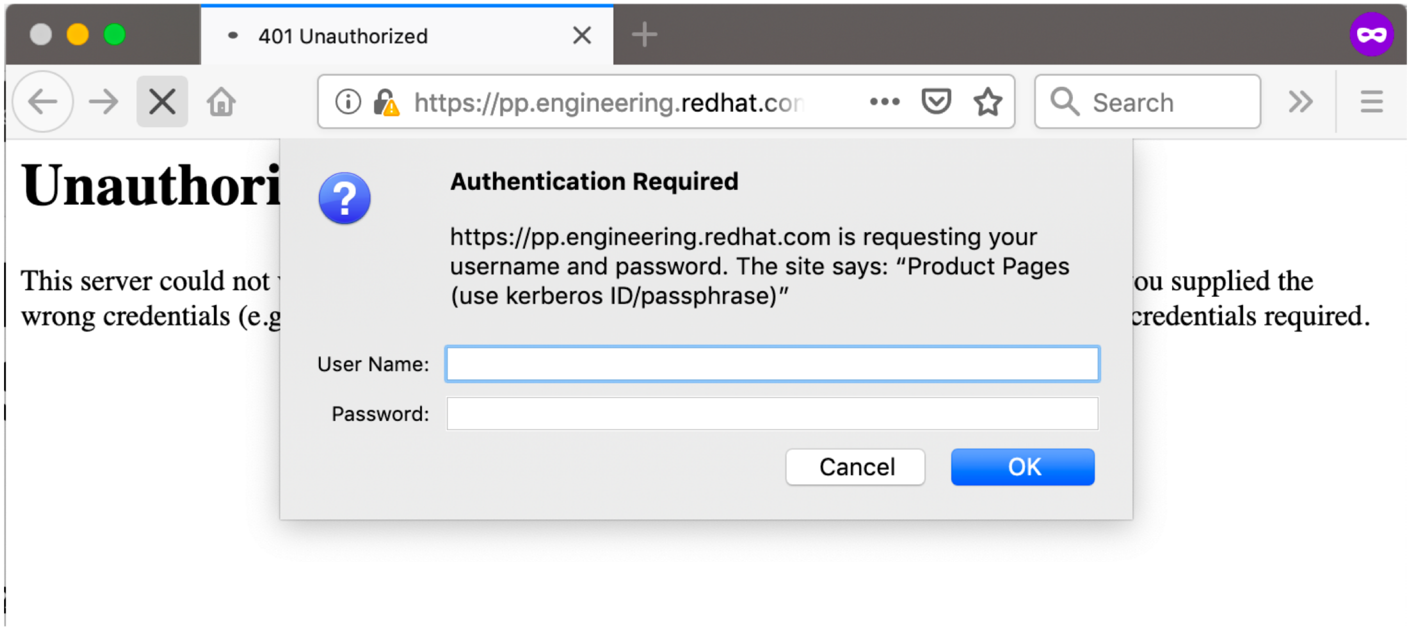
Evolution of API security

Naked
API

Simple
API Keys

Federated Access
Control

Authentication Grandfather



Unauthori

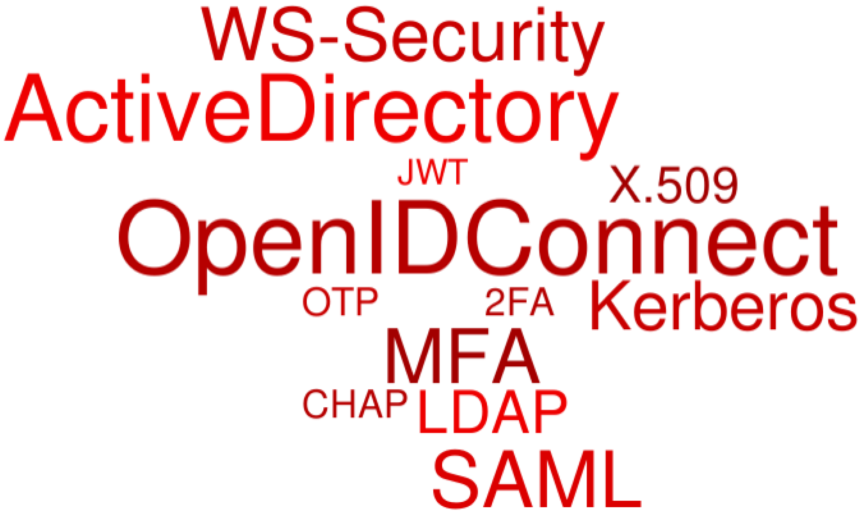
This server could not
wrong credentials (e.g

ou supplied the
credentials required.

Authentication Required

https://pp.engineering.redhat.com is requesting your username and password. The site says: “Product Pages (use kerberos ID/passphrase)”

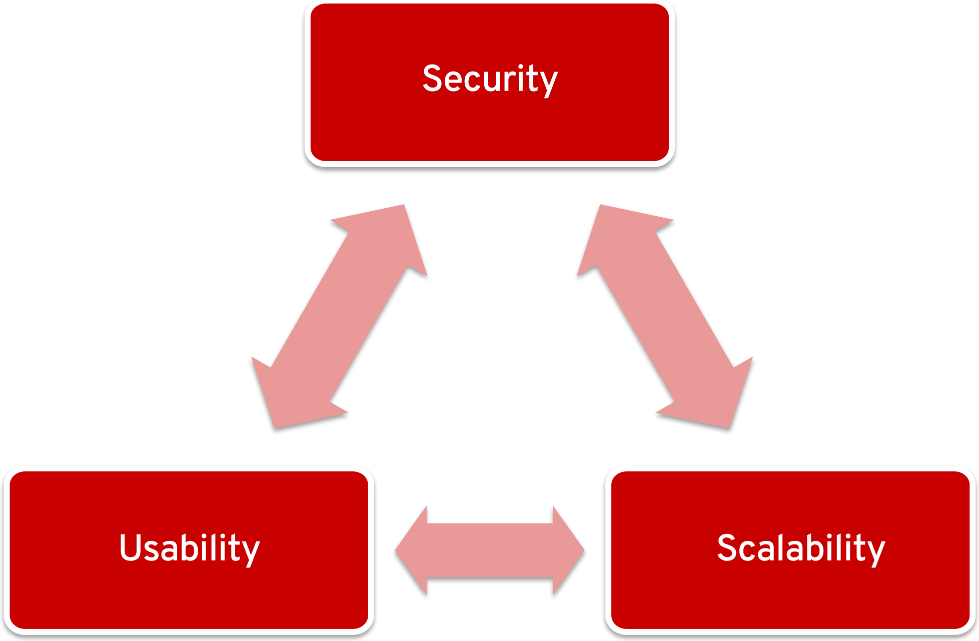
Identity authentication schemes



What is the CAP of the identity world?

CAP Theorem is a concept that a distributed database system can only have 2 of the 3: Consistency, Availability and Partition Tolerance.

SUS makes sense of tradeoffs in authentication schemes


Security

Usability

Scalability

Converged Access Management

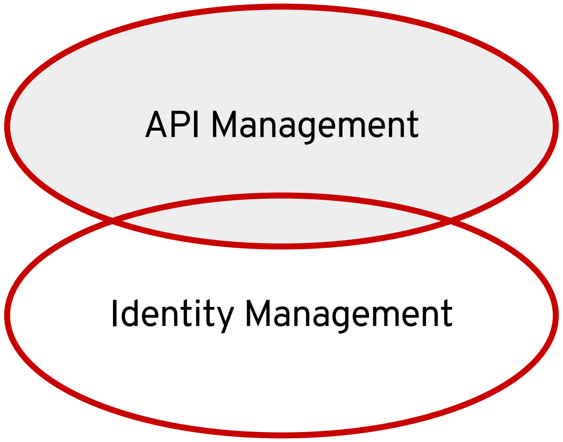
Proliferation of:

Consumer types
- End users
- Apps

Devices
- Web browser
- Mobile app

Security protocols
- SAML
- 2FA

Convergence and Integration:



Let’s focus on security for APIs

Web API security is concerned with the transfer of data through APIs that are connected to the internet.

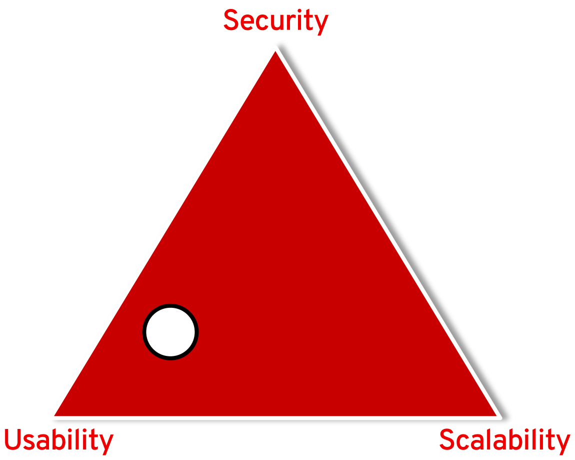
Sample of “big” API providers auth modes

Recurly	Basic Auth curl -u [API Key]: (nothing after the colon)
Twilio	Basic Auth curl -u '{AccountSID}:{AuthToken}'
Wufoo	Basic Auth curl -u api_key:garbage_filler
Stripe	Basic Auth curl -u api-key: (nothing after colon)
FreshDesk	Basic Auth curl -u apikey:X
Stormpath	Basic Auth curl -u $API_KEY_ID:$API_KEY_SECRET
Atlassian	Basic Auth curl -u fred:fred
Sendgrid	Basic Auth curl -u sendgrid_username -X (went back to uname/pwd
Zendesk	Basic Auth curl -u joe_enduser@zendesk.com/token:{YOUR_API_TOKEN}
Github Oauth	Basic Auth, Oauth personal tokens (curl -u <token>:x-oauth-basic), 2FA
AWS	Oauth1 API key with HMAC signature
Yelp	Oauth1 with HMAC signature
Fitbit	Oauth1 with HMAC signature
Rememberthemilk	OAuth1 with MD5 hash signature
Flickr	Oauth1 with MD5 hash signature
Dropbox	OAuth1, OAuth2 (preferred)
Disqus	OAuth2
Stack Exchange	Oauth2
Vimeo	Oauth2
Instagram	Oauth2
LinkedIn	Oauth2
Soundcloud	Oauth2
StatusPage.io	Oauth2
Twitter	Oauth2 in headers

Legacy and ubiquitous at the same time

Basic Auth

- Lots of simple tooling make it very usable
  - HTTP "Authentication Basic:" header
  - curl - u
  - URL access:
  "username:password@mydomain.com/resource"
- Easiest for API providers and consumers because of ubiquity
- New APIs avoid using Basic Auth

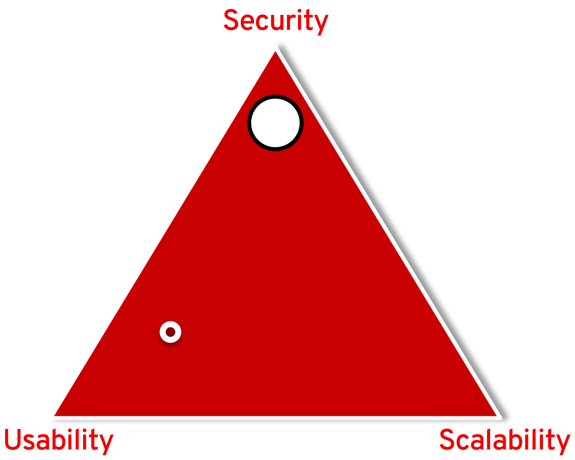


X.509

Mutual SSL Auth

- High security but complex to coordinate
- Good for environments where there is a very low number of consumer apps and the provider has control of both

  e.g. between an API Gateway and Backend service
- Otherwise avoid

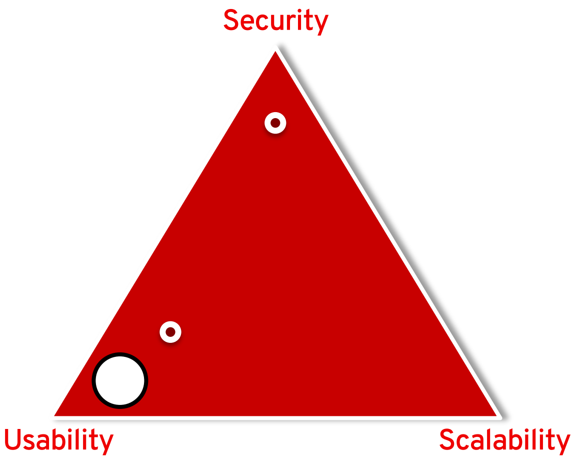


Extreme simplicity

API Key

- Single-string shared secret
- Lots of flexibility:

  - HTTP header
  - URL query parameter
- But key rotation is complicated
- Best option for fast adoption of a low security API



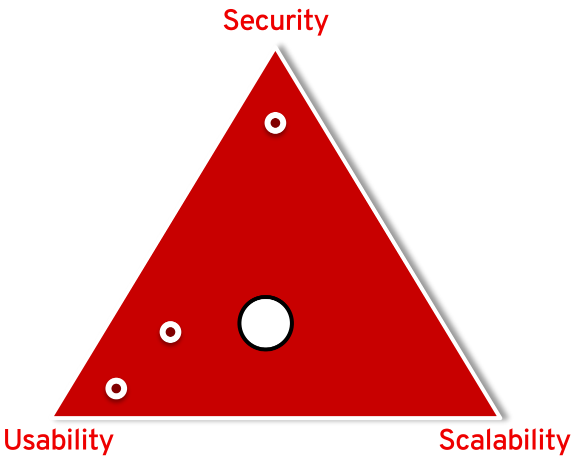
Communication between apps

App ID + Secret

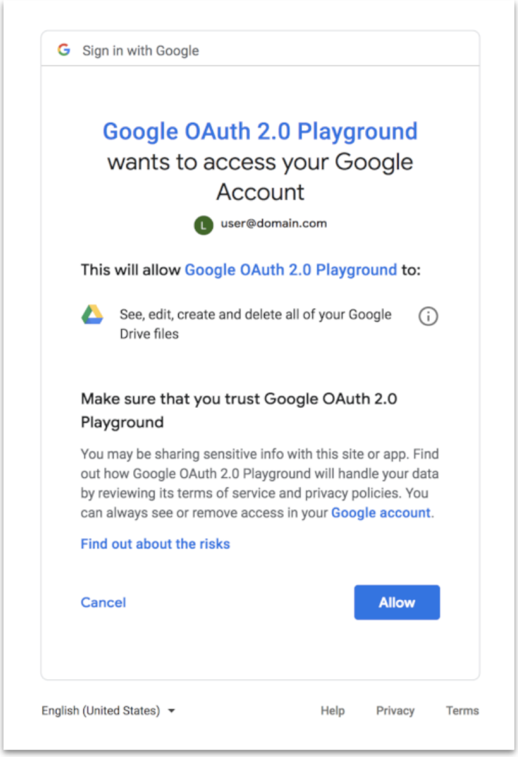
- Same flexibility as API key:

  - HTTP header
  - URL query parameter

- The secret key can be rotated easily:
- Avoid downtime with multiple secrets active at the same time
- Great compromise for app to app communication without complexities of OAuth



More advanced but exploding in popularity is to federate access enabled by OAuth



Sign in with Google

Google OAuth 2.0 Playground wants to access your Google Account

user@domain.com

This will allow Google OAuth 2.0 Playground to:

See, edit, create and delete all of your Google Drive files

Make sure that you trust Google OAuth 2.0 Playground

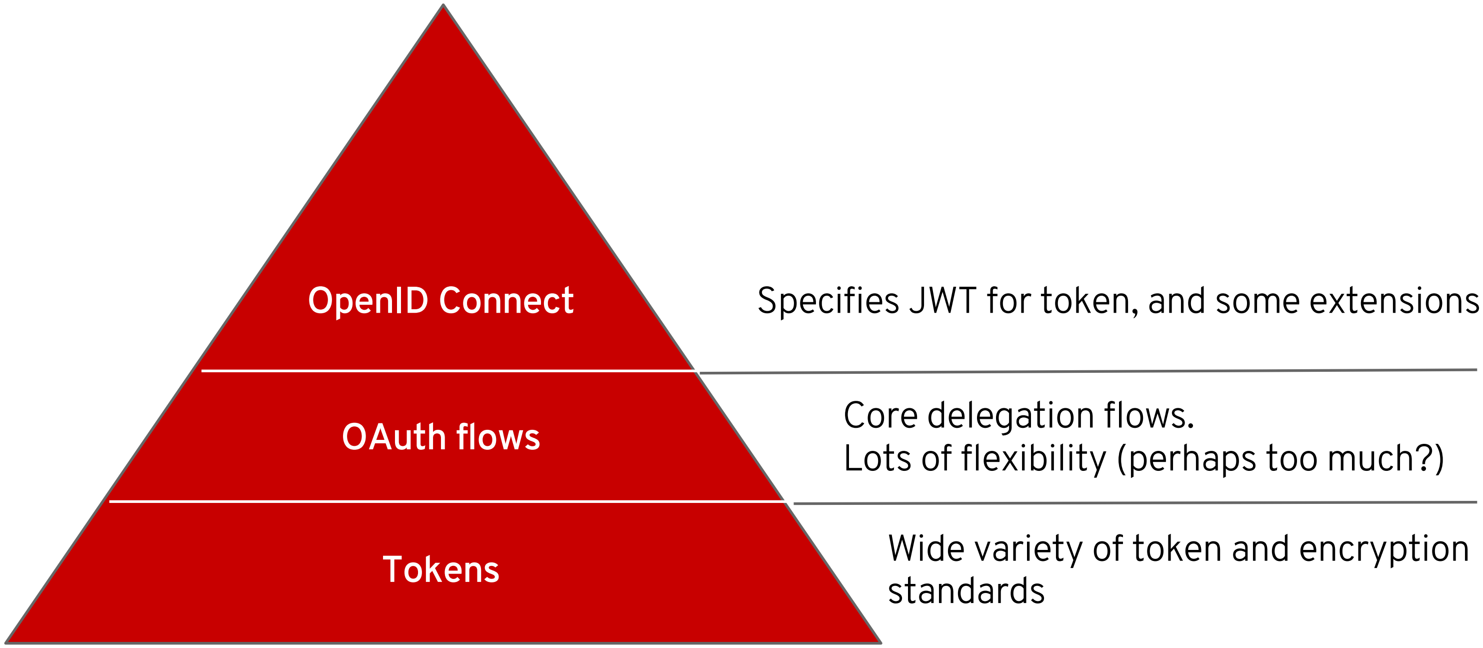
You may be sharing sensitive info with this site or app. Find out how Google OAuth 2.0 Playground will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google account.

Find out about the risks

OAuth enables people to delegate access for apps to act on our behalf

Google Documentation: https://developers.google.com/identity/protocols/OpenIDConnect

Layered Security Standards



Specifies JWT for token, and some extensions

Core delegation flows.
Lots of flexibility (perhaps too much?)

Wide variety of token and encryption standards

Open Authentication (OAuth) Terminology

Authorization Server
Entity in charge of generating and managing both the bearer and refresh tokens. In our case, the Identity Provider (IdP), RH SSO.

Access Token
Data object a client can use to authorize user access to a resource. Has different attributes like longevity and scope.

Refresh Token
Another type of token used in authorization server to get a new access token with the same author as an existing one.

Resource server
Hosts the resources, protects and makes the resources available to properly authenticated and authorized clients.

Resource owner (sometimes referred to as the API provider)
Resource owner manages the resources served by the resource server, typically the user of the application.

OAuth 2.0 Flows

Relevance Order

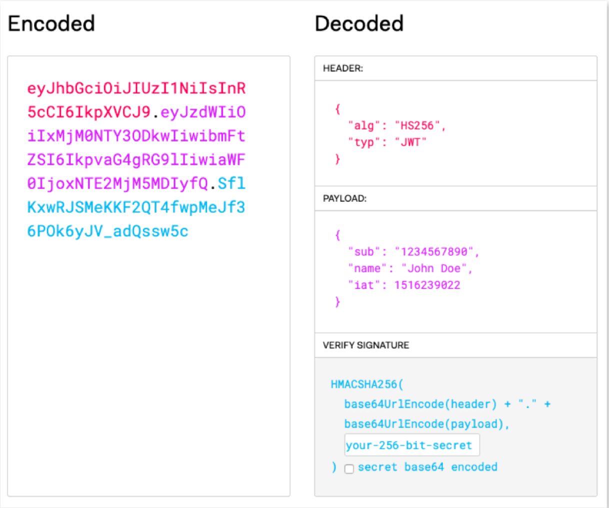
- ✓ **Authorization Code Flow**
  The most secure and used where a user logs into Identity server and grants access to Application to retrieve their data
- ✓ **Client Credentials Flow**
  Only Application data is passed in a single request for an Access Token. Usually used in Machine to machine communication.
- ✓ **Implicit Flow**
  User logs in but secret is not passed - less secure than authorization code flow
- ✓ **Resource Owner Password Flow**
  Application, username and password data is passed in a single request for an Access Token

Access policies (authorization) becomes more complex with OAuth and are harder to federate

JWT (“jot”) to the rescue with OpenID Connect

JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

- Huge deal because policies can be encapsulated in a Bearer Token
- Anyone who possesses the bearer token can certify that they are authorized to access the resource in the JWT
- Eliminates the need to look up against a central access control list
- Massive benefit of distributing responsibility



Encoded

```
eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCJ9.eyJzdWIiO
iIxMjM0NTY3ODkwIiwibmFt
ZSI6IkpvaG4gRG9lIiwiaWF
0IjoxNTE2MjM5MDIyfQ.Sfl
KxwRJSMeKKF2QT4fwpMeJf3
6POk6yJV_adQssw5c
```

Decoded

HEADER:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD:

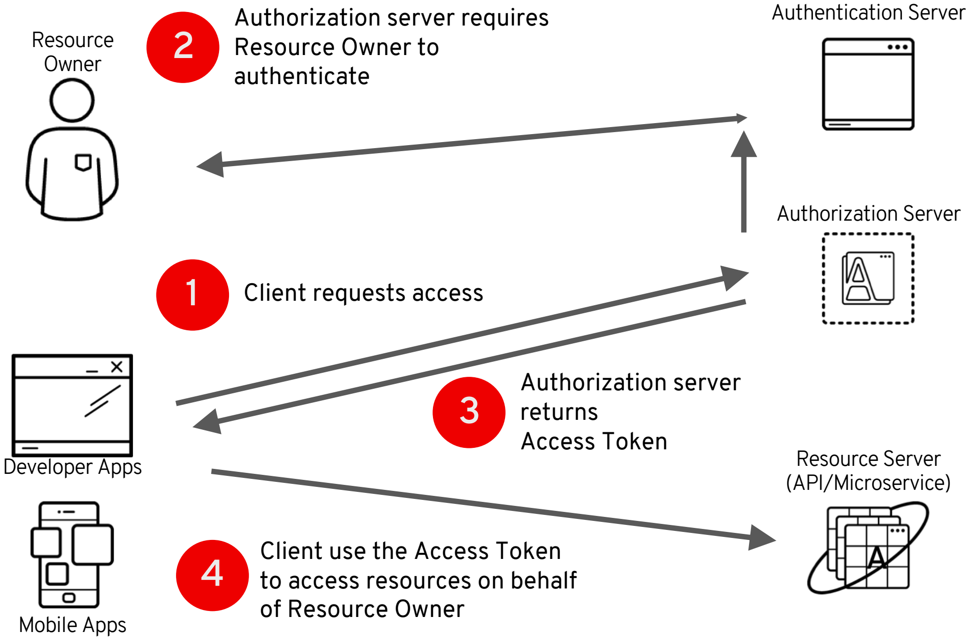
```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

wt.io https://jwt.io/introduction/

OpenID Connect Workflow



Resource Owner
2
Authorization server requires Resource Owner to authenticate
Authentication Server
Authorization Server
1
Client requests access
Developer Apps
3
Authorization server returns Access Token
Resource Server (API/Microservice)
Mobile Apps
4
Client use the Access Token to access resources on behalf of Resource Owner

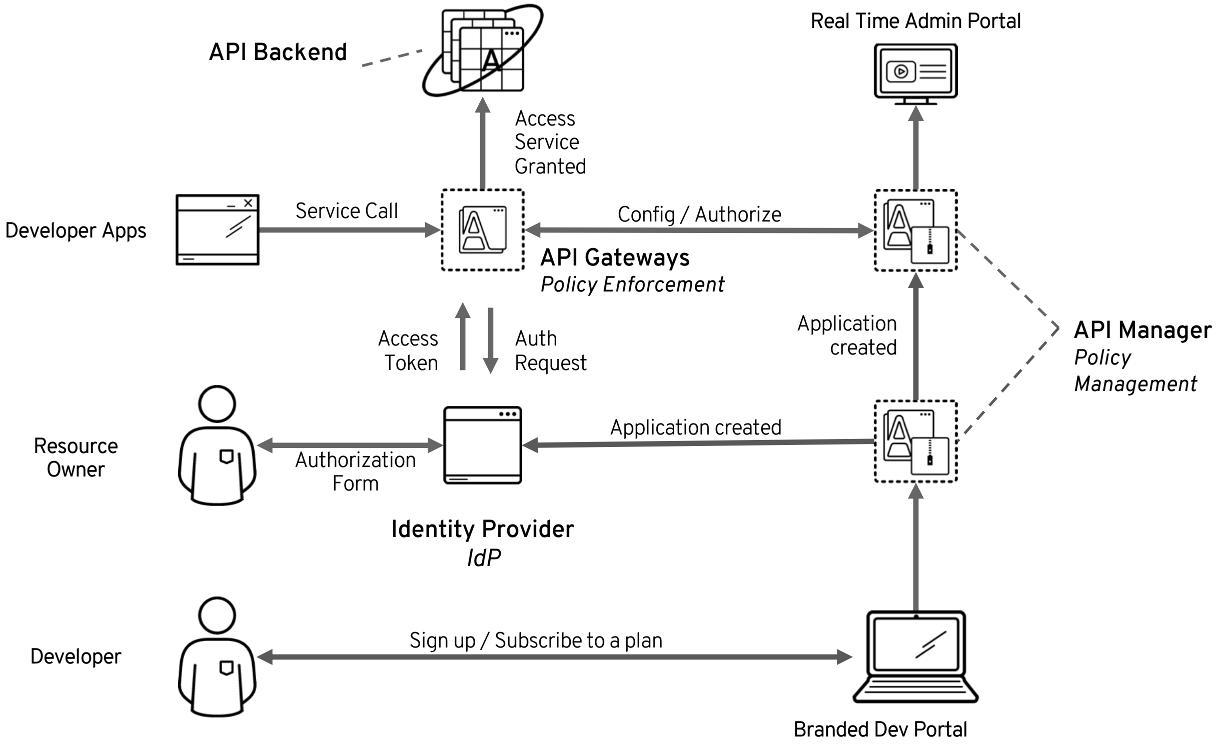
j

- Devil is in the details with OAuth
- Get the requirements explicit and very detailed
- Identify all the actors (End user, application, IdP, Gateway, Resource Server)
- Use sequence diagrams to validate if Red Hat implementation meets requirements
- If the requirements are unique, Red Hat gives the greatest amount of customization flexibility

Photo Source: M.L. Quezon National Highway, Lapu-Lapu City, Philippines

OpenID Connect

Red Hat 3scale and SSO Implementation



Future proof

OAuth

OAuth 1.0

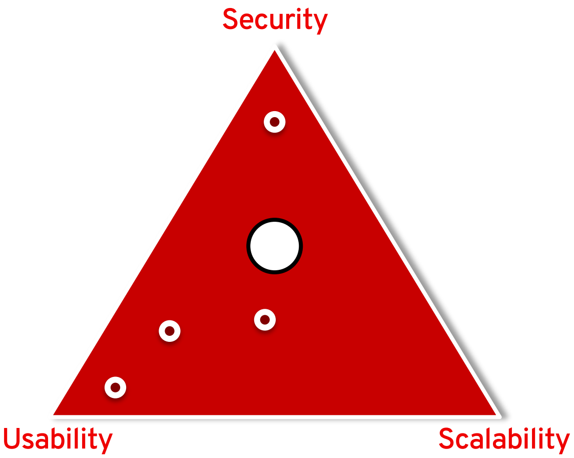
- best on security (due to client signing)
- complex for clients to implement
- generally avoided for new APIs

OAuth 2.0

- best future-proof security model for APIs
- complex for providers to implement
- default choice for new APIs

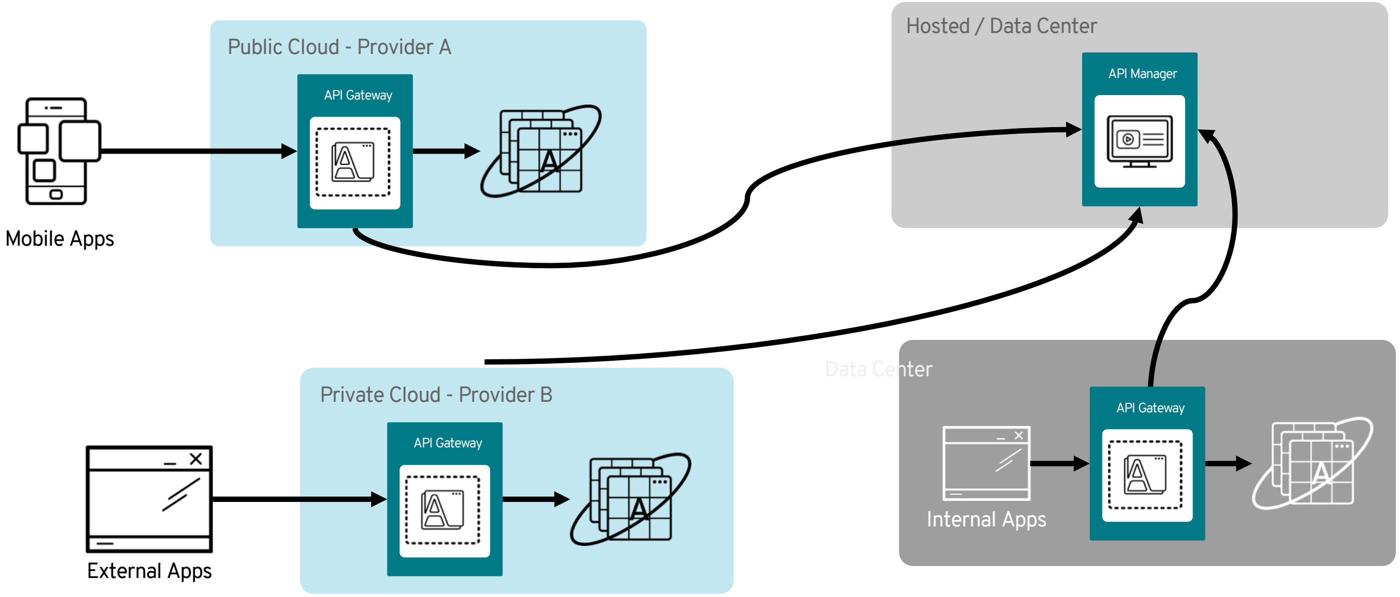
OpenID Connect

- best implementation of OAuth for user delegation

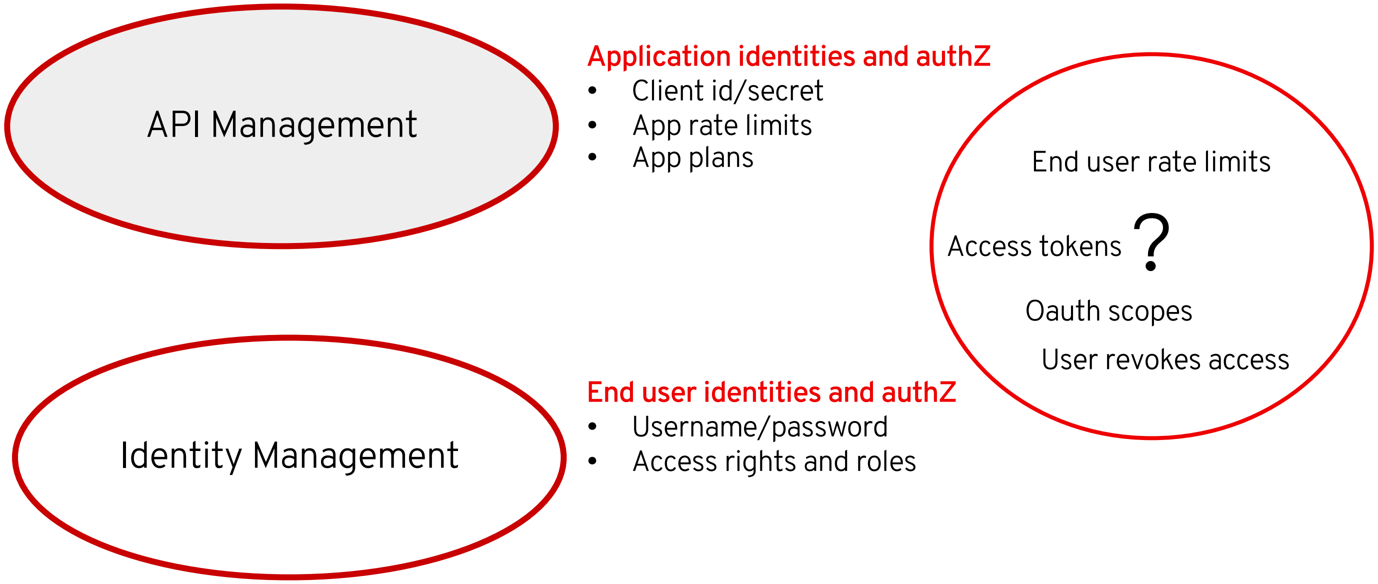


Distributed Policy Enforcement

Multiple Deployment Options



Define clear roles and responsibilities



API Management

Application identities and authZ

- Client id/secret
- App rate limits
- App plans

End user rate limits

Access tokens ?

Oauth scopes

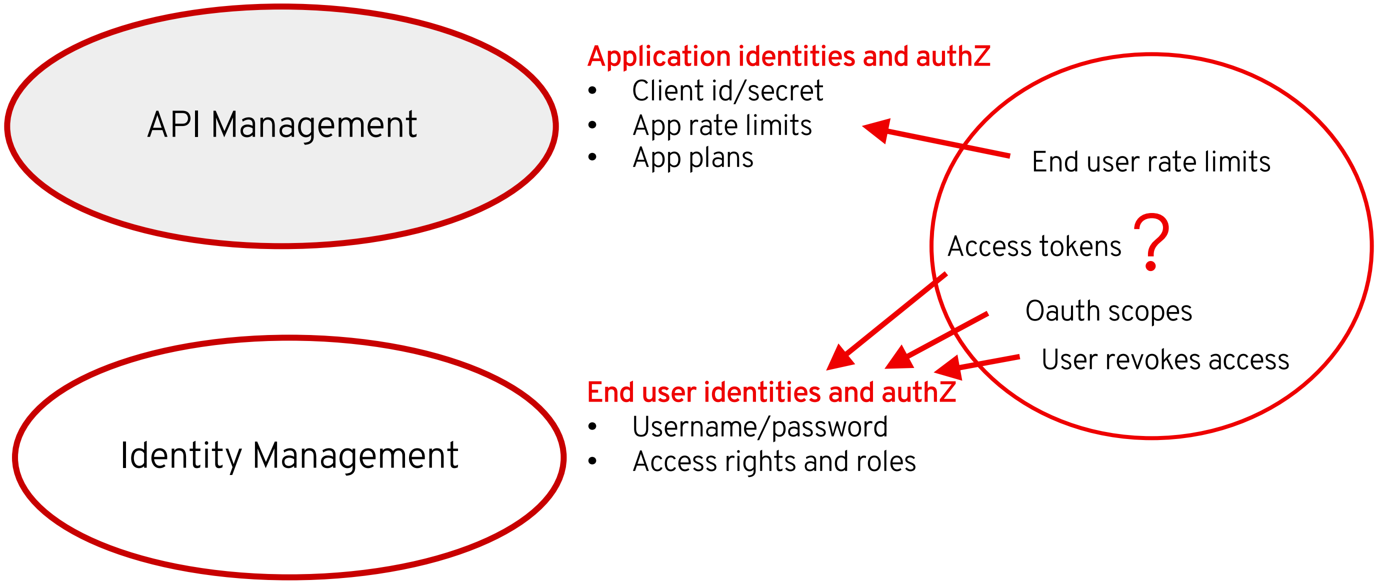
User revokes access

Identity Management

End user identities and authZ

- Username/password
- Access rights and roles

Define clear roles and responsibilities



API Management

Application identities and authZ

- Client id/secret
- App rate limits
- App plans

End user rate limits

Access tokens ?

Oauth scopes

User revokes access

Identity Management

End user identities and authZ

- Username/password
- Access rights and roles

Service Mesh

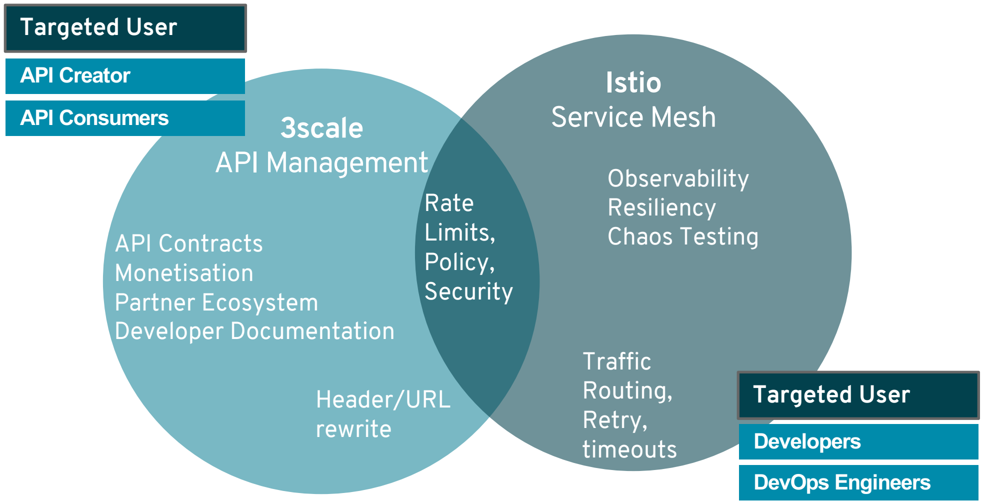
Do I need API and Service Mesh management?

As the number of services increases this becomes more a MicroServices Architecture (MSA) discussion.

A service mesh is decentralized application-networking infrastructure between your services that provides resiliency, **security**, observability, and routing control.

A service mesh is about **connecting** your **applications**.

Difference between API Management and Advanced Traffic Management, is not so much where the traffic is flowing, but what concepts are used to view & control the traffic, and all things concerning the business side of APIs.



Service Mesh and API Management use cases

Advanced Traffic Management

Application performance, debugging, analytics data, incident management

Security (mTLS, RBAC)

Resiliency

Traffic routing

Infrastructure rate limiting based on multiple sources

Managing Relationships

Manage who can access APIs

Manage how they can access APIs, configuring contracts & limits

Developers can find services & sign up

Ability to package multiple services into one API product

Get insights on usage of APIs

External APIs

Send invoices and charge developers for API usage

Service mesh will be able to do some rate limiting, but it won’t be able to handle subscription based security.

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

in linkedin.com/company/red-hat

▶ youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat