openid connect all the things

@pquerna
CTO, ScaleFT

CoreOS Fest 2017 - 2017-07-01


ScaleFT


THE
APACHE®
SOFTWARE FOUNDATION


node
JS®


Bloglines


CYAN

Problem

- More Client Devices per-Human
- Many Cloud Accounts
- More Apps: yay k8s
- More Distributed Teams
- VPNs aren’t fun
- Legacy Solutions aren’t keeping up

Traditional Authentication and Authorization assumed a perimeter architecture which no longer exists

	“Web Native”	UX	Implementation Complexity?	Security Properties
LDAP	No	Password Prompt	2/10	D
Kerberos	Maybe	MIT Students are ready for it	7/10	A
SAML	Yes	Browser Redirects	9/10	B
x.509 Certificates	Maybe	Estonians are ready for it	10/10	A
OpenID Connect	Born from “Web 2.0”	Browser Redirects	6/10	B

blogs

blogs with comments

blogs with comments with spam

SPAM

Abridged History of OpenID, OAuth, OAuth2, OIDC

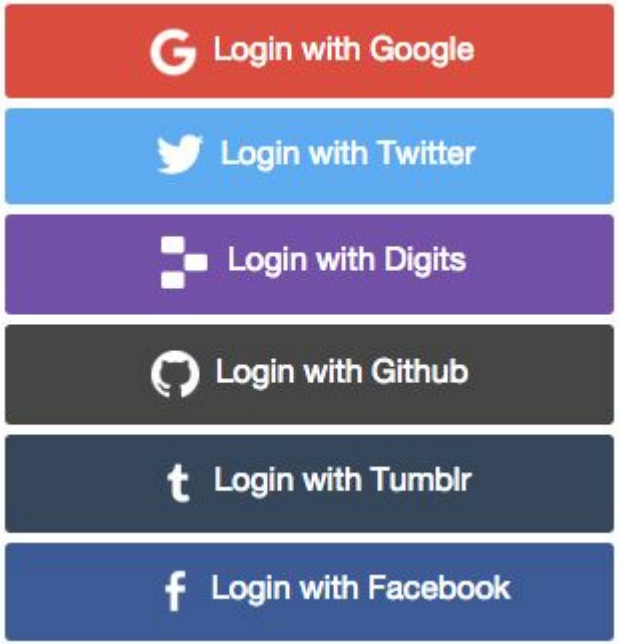
- 2005: @bradfitz (LiveJournal): Yadis: Yet another distributed identity system
  - Login across sites easily
- 2006: @blaine (Twitter): OAuth:
  - Get Access Token & Authorization
- 2007: OpenID Foundation and OpenID 2.0
- 2012: OAuth 2.0
  - Make it simpler
- 2014: OpenID Connect
  - Make it simpler
- Today:
  - Dozens of RFCs and Drafts

OAuth 2.0 and **OpenID Connect (OIDC)**
are the ones you want

Terminology decoded

- Identity Provider (IdP): Someone in IT owns this, it connects to Active Directory eventually
- Relying Party (RP): Your Application, the thing you want to protect
- Client, End-User: Your User

The Real World



- Identity Providers (IdPs):
  - Consumers: Few “mega” IdPs*
  - Corporate:
    - ADFS
    - Google-Apps
    - Okta
    - Ping
- Use cases:
  - Webapps!
  - CLI Tools
    - `gcloud auth login`
    - `kubectl**`
  - ScaleFT for SSH/RDP

* Facebook isn’t technically OIDC
** kubectl just passes it around

Picking your OIDC Flow

- You want: **“Authorization Code Flow”** or **“Code Flow”**, almost never “Implicit” or “Hybrid”:
  - Implicit: Use case Single-Page Javascript Applications
  - Hybrid: Ignore unless you are an IdP

Authorization Code Flow

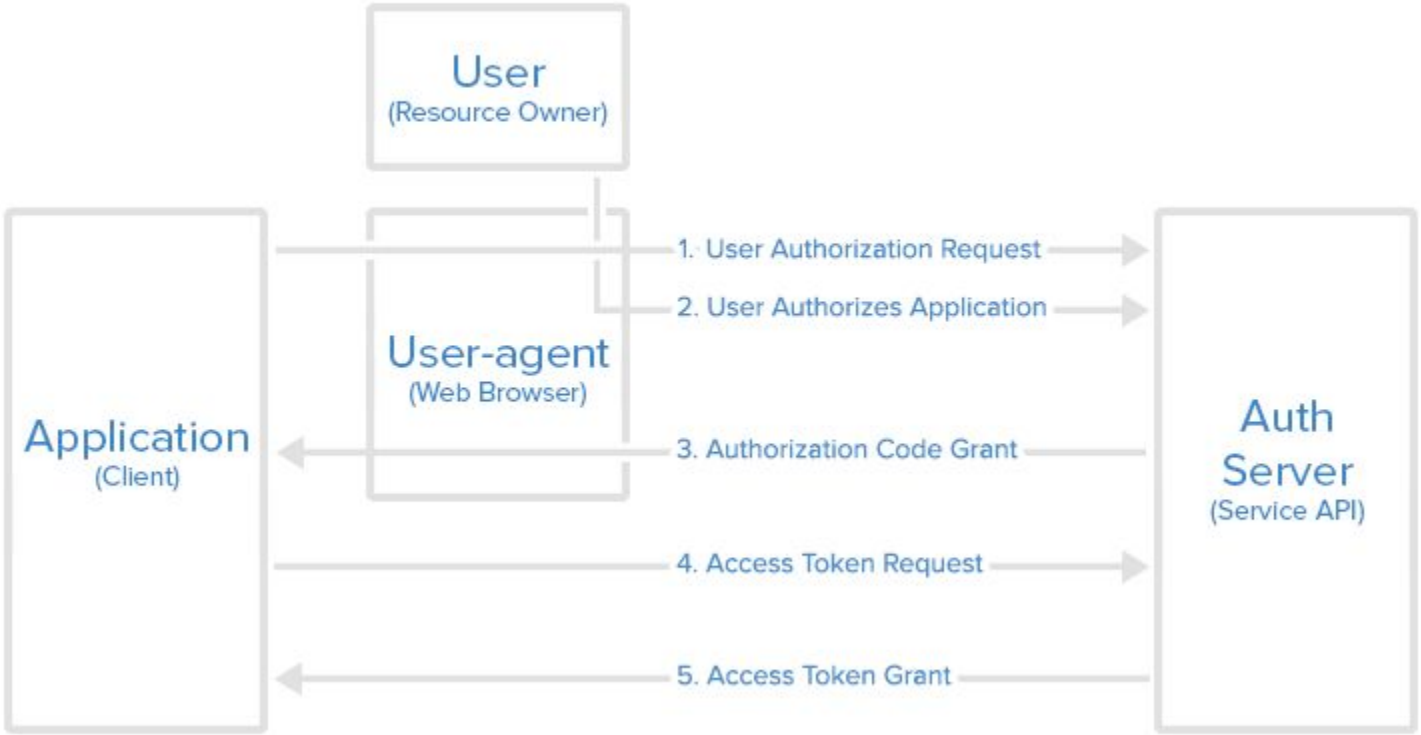


Diagram Source: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

What?

I just want this to work

1. Register your App Callback URL with IdP
    a. Get “OIDC Client ID” and “OIDC Client Secret”
2. Fetch and cache Discovery Document
3. Session Management
    a. Redirect Browser to IdP
    b. Receive Callback from Browser
        i. Exchange Code for Token
        ii. Validate, If everything is cool, create browser session (cookie?)

Discovery Document


```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/
 "token_endpoint": "https://www.googleapis.com/oauth2/v4
 "userinfo_endpoint": "https://www.googleapis.com/oauth2
 "revocation_endpoint": "https://accounts.google.com/o/o
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
```


- `/.well-known/openid-configuration`
- JSON Contains
  - Endpoints
  - Supported Scopes
  - Supported Claims
- Not “static”

Construct a Redirect URL

- authorization_endpoint: HTTPS URL from Discovery Document

- State
  - Opaque value passed back to you
  - Bind state to Browser Cookie.
  - Use a NaCL Secret Box or similar.
- Redirect Client
  - HTTP 307 (Temporary Redirect)

${authorization_endpoint}?

&client_id=${OIDC_CLIENT_ID}

&redirect_uri=${REGISTERED_URL}

&response_type=code

&scope=openid profile email

&state=${SIGNED_STATE}

Receive Callback

1. Validate State
2. Handle Errors
3. Exchange Code
   - POST to token_endpoint
4. Validate ID Token
5. Establish Session

Access Tokens

- Opaque
- Used as API bearer tokens
- IdP / Application use case specific

ID Tokens

- JSON Web Token (JWT)
- Must be **verified**
- Contains Claims
  - These are what you care about

Javascript Object Signing and Encryption (JOSE)

- JSON all the things
- Formats and standards for
  - Keys
  - Encryption
  - Signing

JWTs

eyJhbGciOiJSUzI1NiIsImtpZCI6IjFiNDgyZWZhLWE
wNDYtNDk2ZC1iNTNhLTBlMTkwNjU2MzBkNyJ9.eyJhd
WQiOlsiNTE5NmYwNTItYTJiMC00MTMyLWI0NzAtNDQx
MjkyMWRkY2Q4Il0sImF1dGh6X3N2YyI6Imh0dHBzOi8
vYXV0aG9yaXplLnNjYWxlZnQuY29tIiwiZW1haWwiOi
JwYXVsLnF1ZXJuYUBzY2FsZWZ0LmNvbSIsImV4cCI6M
TQ5NjI5NDgwMiwiaWF0IjoxNDk2MjU4ODAyLCJpc3Mi
OiJodHRwczovL2FwcC5zY2FsZWZ0LmNvbSIsImp0aSI
6ImNmMWMzNDUxLTMyZDctNGMzOS04N2IyLTJjY2Y3Zj
Y5NDVhZSIsIm5iZiI6MTQ5NjI1ODY4Miwic3ViIjoiO
TE0MGQ2NWQtMjRhOS03OTY2LWZmZDMtYzI4NzM2MTE1
OTA2In0.quNzteqys1jREeNfA-G_oOD20_5jgmJ80Vy
vGliBvzxckiJN-ALcTzeNUiZKAGmdCaRXIuhbnZj2OD
pscSENKIp-FhQ4L7C8bPo7s279E0E-RmuCFTdgKvb4y
b_EleCP9jkbkeHa7sT9-pIvcne6_Czr9OyvaFp9I9rD
-o28wOlvEL1K-EzLvJEEvUehld6MJ5SNrDeIEWtHM37
-u-D0UtBzd6E_Qj45uWJaP0HnXVTT8ovfAWZkedyApK
xOGqR6xdRhHFzCNb3RilM_KfnbrvQ5BT4p52v8l2pNg
PDnkQv1I8HZVEOQvspKSSPiBEQBaTJJQcN3rpIVsVov
Zla0Xw

JWTs

- Period Separated
- Base64 (w/ url encoding)
- Header [red]
- Payload [blue]
- Signature [orange]

eyJhbGciOiJSUzI1NiIsImtpZCI6IjFiNDgyZWZhLWE
wNDYtNDk2ZC1iNTNhLTBlMTkwNjU2MzBkNyJ9.eyJhd
WQiOlsiNTE5NmYwNTItYTJiMC00MTMyLWI0NzAtNDQx
MjkyMWRkY2Q4Il0sImF1dGh6X3N2YyI6Imh0dHBzOi8
vYXV0aG9yaXplLnNjYWxlZnQuY29tIiwiZW1haWwiOi
JwYXVsLnF1ZXJuYUBzY2FsZWZ0LmNvbSIsImV4cCI6M
TQ5NjI5NDgwMiwiaWF0IjoxNDk2MjU4ODAyLCJpc3Mi
OiJodHRwczovL2FwcC5zY2FsZWZ0LmNvbSIsImp0aSI
6ImNmMWMzNDUxLTMyZDctNGMzOS04N2IyLTJjY2Y3Zj
Y5NDVhZSIsIm5iZiI6MTQ5NjI1ODY4Miwic3ViIjoiO
TE0MGQ2NWQtMjRhOS03OTY2LWZmZDMtYzI4NzM2MTE1
OTA2In0.quNzteqys1jREeNfA-G_oOD20_5jgmJ80Vy
vGliBvzxckiJN-ALcTzeNUiZKAGmdCaRXIuhbnZj2OD
pscSENKIp-FhQ4L7C8bPo7s279E0E-RmuCFTdgKvb4y
b_EleCP9jkbkeHa7sT9-pIvcne6_Czr9OyvaFp9I9rD
-o28wOlvEL1K-EzLvJEEvUehld6MJ5SNrDeIEWtHM37
-u-D0UtBzd6E_Qj45uWJaP0HnXVTT8ovfAWZkedyApK
xOGqR6xdRhHFzCNb3RilM_KfnbrvQ5BT4p52v8l2pNg
PDnkQv1I8HZVEOQvspKSSPiBEQBaTJJQcN3rpIVsVov
Zla0Xw

It’s just JSON


```
eyJhdWQiOlsiNTE5NmYwNTItYTJiMC00MTMyLWI0NzAtN
DQxMjkyMWRkY2Q4Il0sImF1dGh6X3N2YyI6Imh0dHBzOi
8vYXV0aG9yaXplLnNjYWxlZnQuY29tIiwiZW1haWwiOiJ
wYXVsLnF1ZXJuYUBzY2FsZWZ0LmNvbSIsImV4cCI6MTQ5
NjI5NDgwMiwiaWF0IjoxNDk2MjU4ODAyLCJpc3MiOiJod
HRwczovL2FwcC5zY2FsZWZ0LmNvbSIsImp0aSI6ImNmMW
MzNDUxLTMyZDctNGMzOS04N2IyLTJjY2Y3ZjY5NDVhZSI
sIm5iZiI6MTQ5NjI1ODY4Miwic3ViIjoiOTE0MGQ2NWQt
MjRhOS03OTY2LWZmZDMtYzI4NzM2MTE1OTA2In0
```



```
{
  "aud": [“5196f052-a2b0-4132-b470"],
  "email": "paul.querna@scaleft.com",
  "exp": 1496294802,
  "iat": 1496258802,
  "iss": "https://app.scaleft.com",
  "jti": "cf1c3451-32d7-4c39-87b2",
  "nbf": 1496258682,
  "sub": "9140d65d-24a9-7966-ffd3"
}
```

Verifying a JWT

1) Fetch and cache jwks_uri (from discovery document)
2) Verify Signature from trusted JWKs
3) Validate Claims fit expectations
    a) iss check
    b) sub check
    c) aud check
    d) exp check
    e) nbf check
    f) iat check
    g) jti check

Scopes, Claims & Profiles

- Must send `openid` scope
- Unique Key: `iss + sub`
- Very IdP specific
- Google Apps: hd claim (“hosted domain”)

IdP Claim Portability: Can I actually use X?

sub

email

picture
name

groups
phone
address
etc*

Deployment Architecture

- Implementation is too damn hard **and** risky!
- Validating Authn & Authz in separate component that just forwards traffic
  - ~~microservice~~, errrrr a Reverse Proxy
- Google's BeyondCorp
  - Google Cloud IAP
- Implementations:
  - ScaleFT Access Fabric
  - Istio (roadmap)
  - Apache: mod_auth_openidc
  - Nginx: lua-resty-openidc

Painful Things

- Incomplete & Non-Portable Claims
  - Google Group Membership: Separately use Groups API
  - Github Org Membership: Use Access Token to read Org API
  - Require MFA
- Relying Party Provisioning & Revocation Model
  - System for Cross-domain Identity Management (SCIM)
- Phishing & UX (Google Docs May 2017)
  - Teaches users to go to very big URLs, click yes every time
- Bad JWT Implementations (alg=none)
- Signing Secret vs Bearer Token (Cloudbleed)

OIDC in Go

https://github.com/golang-samples/gopher-vector

- DIY:
  - golang.org/x/oauth2
  - CoreOS"s go-oidc: github.com/coreos/go-oidc
  - Square's go-jose (v2 branch): github.com/square/go-jose/tree/v2
- Simple:
  - @dghubble gologin: github.com/dghubble/gologin

thank you!
questions?

paul@scaleft.com
@pquerna on twitter, github, etc

Slides: paul.querna.org/slides/OIDC-CoreOS-Fest-2017.pdf

https://goo.gl/g8Q9Vc