Hacking OAuth: Pitfalls and Remedies
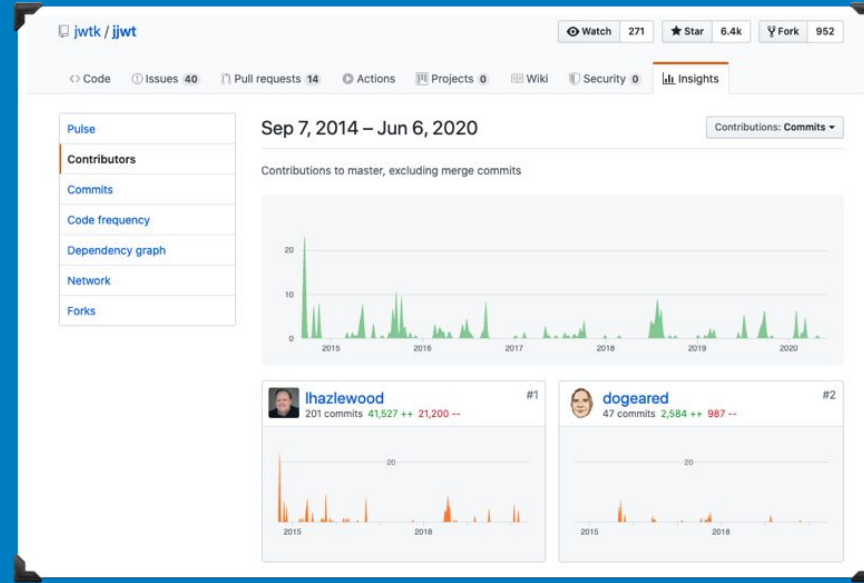

https://github.com/jwtk/jjwt

https://bit.ly/OIDCPrimer
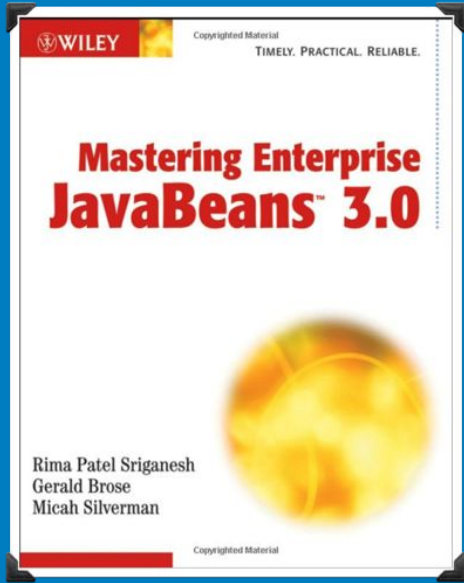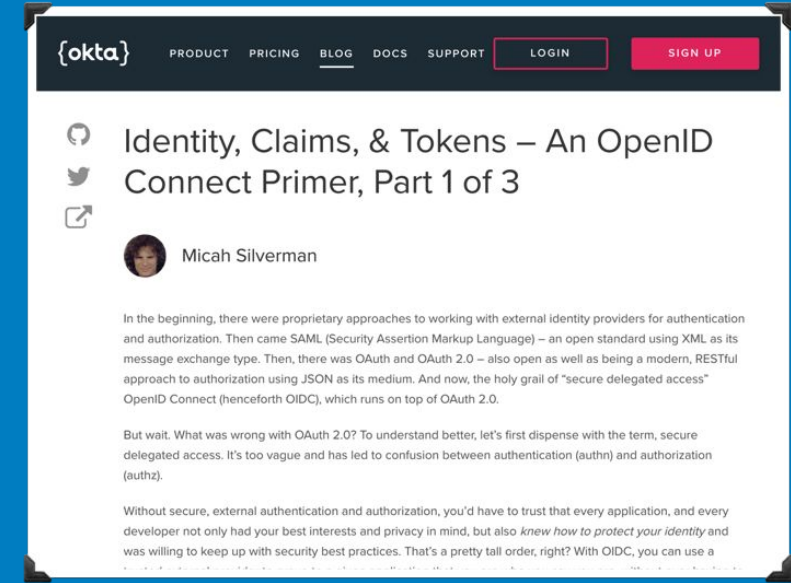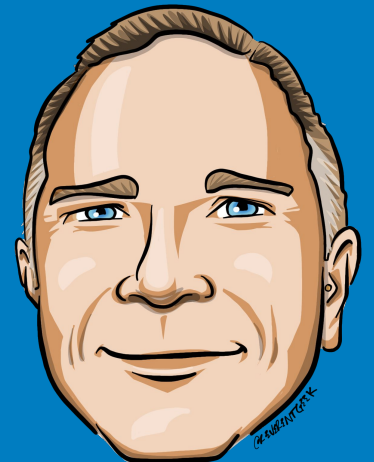
MICAH SILVERMAN
Senior Security H@X0R
@AFITNERD
@OKTADEV

Identity use cases (circa 2007)

★ Simple login – forms and cookies

★ Single sign-on across sites – SAML

★ Mobile app login – ???

★ Delegated authorization – ???

Don't do it this way!

Are your friends already on Yelp?

Many of your friends may already be here, now you can find out. Just log in and we'll display all your contacts, and you can select which ones to invite! And don't worry, we don't keep your email password or your friends' addresses. We loathe spam, too.

Your Email Service — ○ msn Hotmail   ○ YAHOO! MAIL   ○ AOL Mail   ⦿ Gmail

Your Email Address — `ima.testguy@gmail.com`   (e.g. bob@gmail.com)

Your Gmail Password — `••••••••••••`   (The password you use to log into your Gmail email)

Skip this step   **Check Contacts**

Don't do it this way!

The delegated authorization problem

HOW CAN I LET A WEBSITE ACCESS MY DATA, WITHOUT GIVING IT MY PASSWORD?

A lot of confusion around OAuth:

- ✕ Terminology and jargon
- ✕ Incorrect advice
- ✕ Hard to find a life-like example

Hotel Analogy
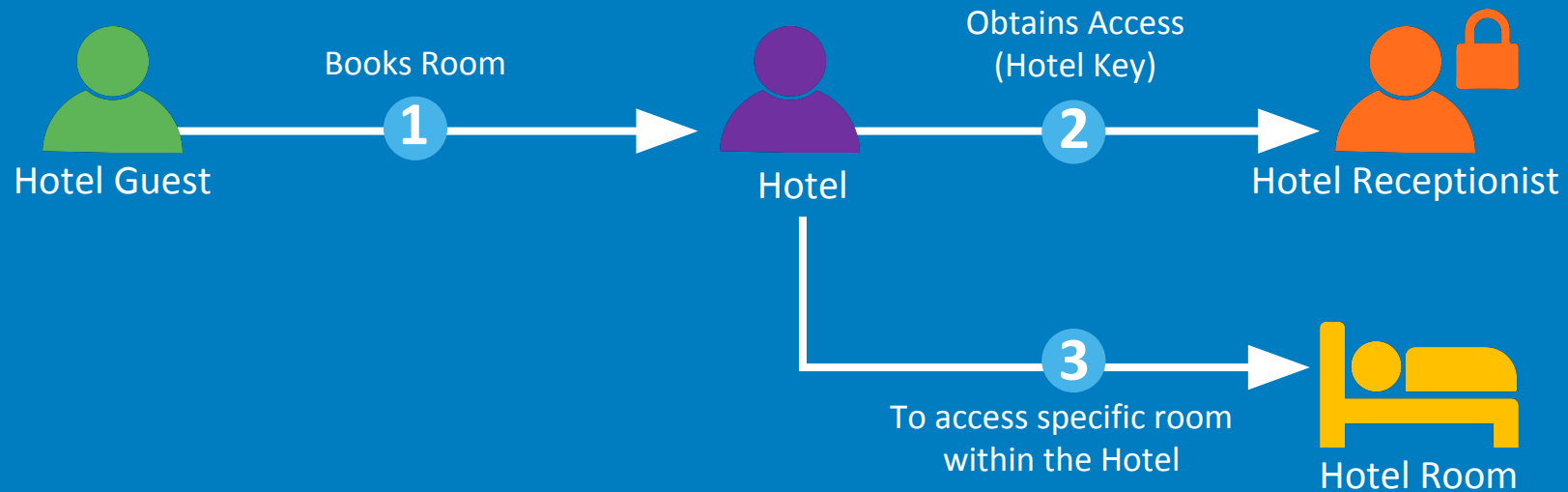
OAuth: App Scenario
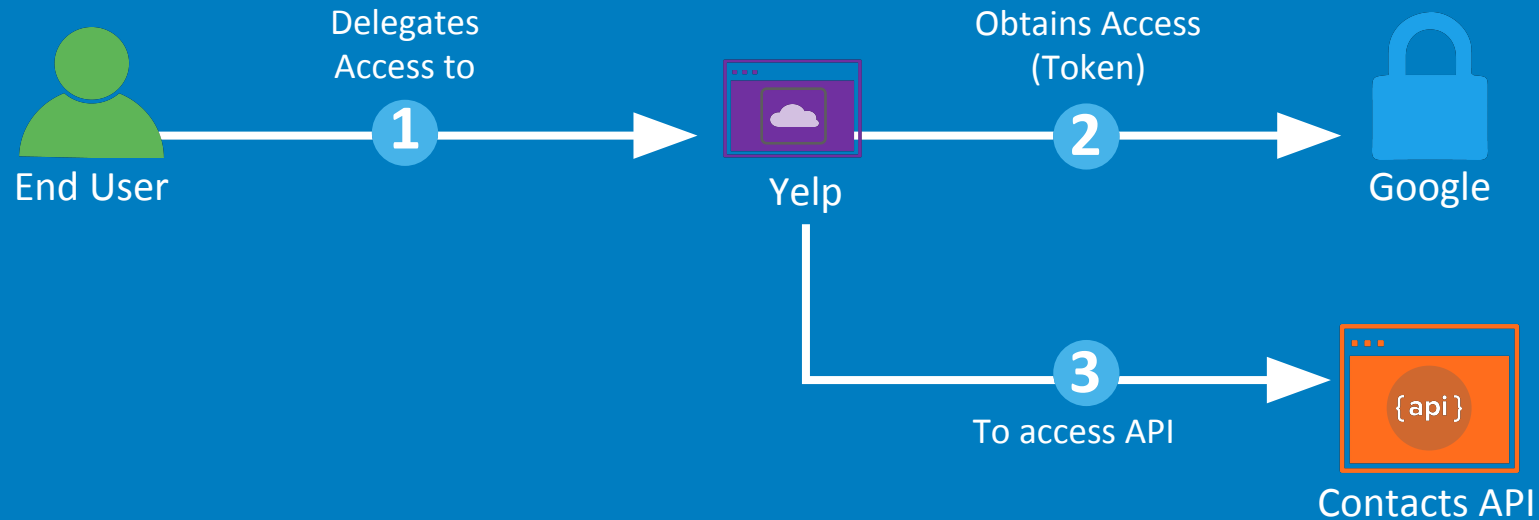
OAuth 2.0 Terminology

Hotel	App	OAuth	Description
Hotel Guest	End User	Resource Owner	Wants a Client App to do something on their behalf
Hotel	Yelp	Client Application	Needs authorization to interact with an API on behalf of a user
Hotel Receptionist	Google	Authorization Server	Grants access (in the form of tokens) to an app
Hotel Room	Contacts API	Resource Server	Has an API that an app can use if presented with a token

Delegated authorization with OAuth 2.0

I trust Gmail and I kind of trust Yelp. I want Yelp to have access to my contacts only.

yelp.com

Connect with Google

OAuth 2.0 implicit flow

Client

Yelp Angular app

Hello!

Resource owner

Authorization server

accounts.google.com

Email
Password

Go to authorization server

Redirect URI: `yelp.com/callback`
Response type: `token`
Scope: `profile contacts`

Back to redirect URI with token

Request consent from resource owner

Talk to resource server with access token

contacts.google.com

accounts.google.com

Allow Yelp to access your public profile and contacts?

No   Yes

More OAuth 2.0 terminology

★ Back channel (highly secure channel)
★ Front channel (less secure channel)

OAuth 2.0 implicit flow

Client

Yelp Angular app

Hello!

Resource owner

Go to authorization server

Redirect URI: `yelp.com/callback`
Response type: `token`
Scope: `profile contacts`

Authorization server

accounts.google.com

| Email |
| Password |

Request consent
from resource owner

Back to redirect URI
with token
(front channel)

Talk to resource server
with access token
(back channel)

contacts.google.com

accounts.google.com

Allow Yelp to access your public
profile and contacts?

No   Yes

DEMO
Implicit Flow Detector

Confidential Clients

Public Clients

Application running on a server

Has the ability to keep strings secret
since code is running in a trusted environment

The application can't keep strings secret

Javascript/Single-Page apps: "view source"
Native apps: decompile and extract strings

OAuth 2.0 authorization code flow

Client App

yelp.com

[Connect with Google]

Resource owner

contacts.google.com

Talk to resource server
with access token
(back channel)

yelp.com/callback

Loading…

Go to authorization server
(front channel)

Redirect URI: `yelp.com/callback`
Response type: `code`
Scope: `profile contacts`

Exchange authorization code
for access token (back channel)

Back to redirect URI
with authorization code
(front channel)

Authorization server

accounts.google.com

| Email |
| Password |

Request consent
from resource owner

accounts.google.com

Allow Yelp to access your public
profile and contacts?

[No] [Yes]

Authorization Code Injection Attack

Target User's Computer (Resource Owner)

malicious browser extension

2. Intercept and publish auth code

Centralized Attack Server

1. Obtain auth code per regular flow

6. Client exchanges target's auth code as normal

accounts.google.com (Authorization Server)

4. Retrieve target's auth code and replace in redirect uri

papercall.io (Confidential Client)

3. Attacker kicks off flow as normal. redirect uri is intercepted

5. Send browser redirect uri with attacker's state and target's auth code

7. Attacker is now logged in as Target

attacker's browser extension

Attacker's Computer
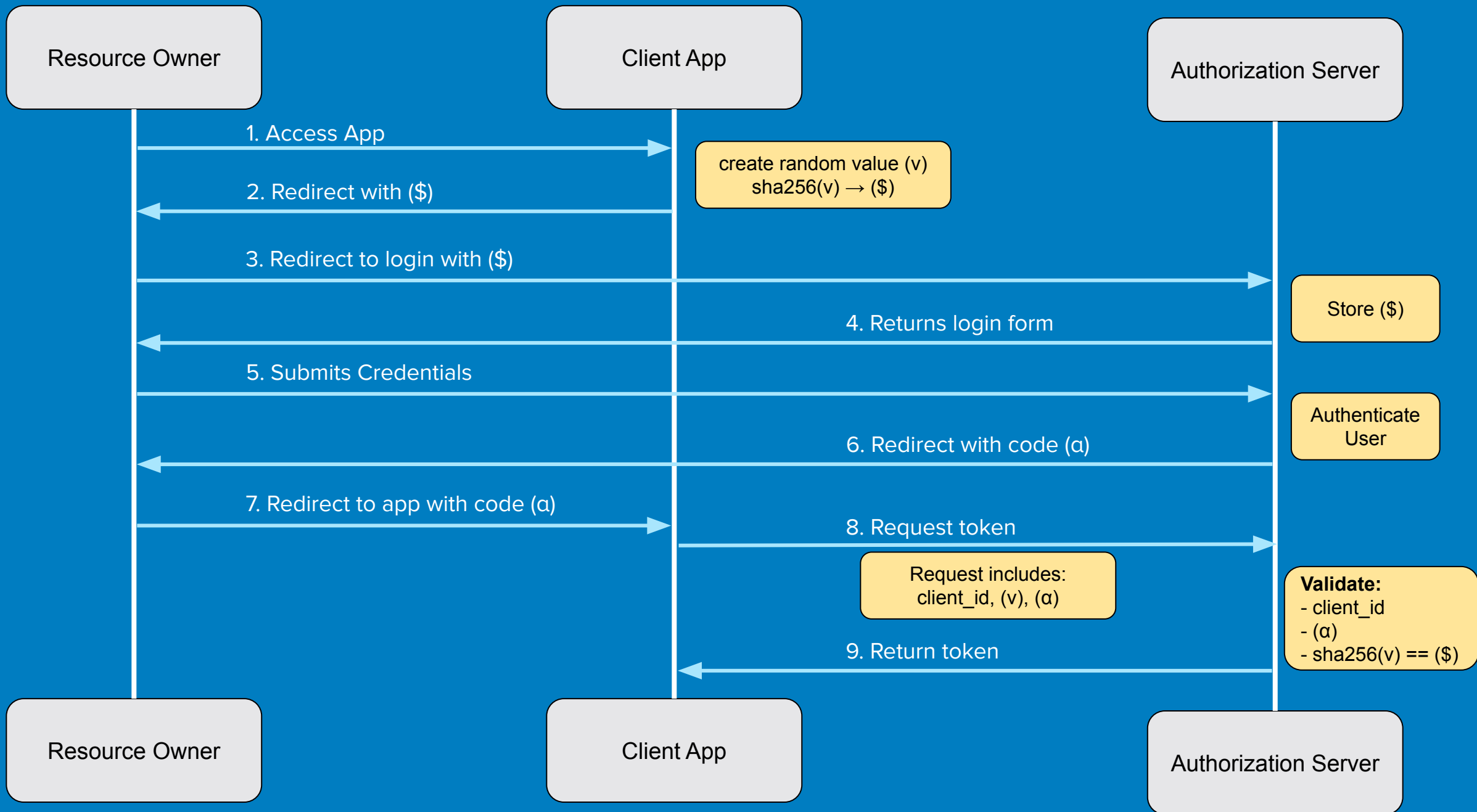
DEMO
Auth Code Injection

DEMO
PKCE
Auth Code Injection: Foiled

Micah Silverman
@afitnerd

oauth.com
@oktadev

Free hosted authorization server:
developer.okta.com