

Introduction to OpenID Connect

Dominick Baier

<http://leastprivilege.com>

@leastprivilege



Outline

- **OAuth2 and authentication**
- **OpenID Connect**
- **Flows**

OAuth2 & Authentication

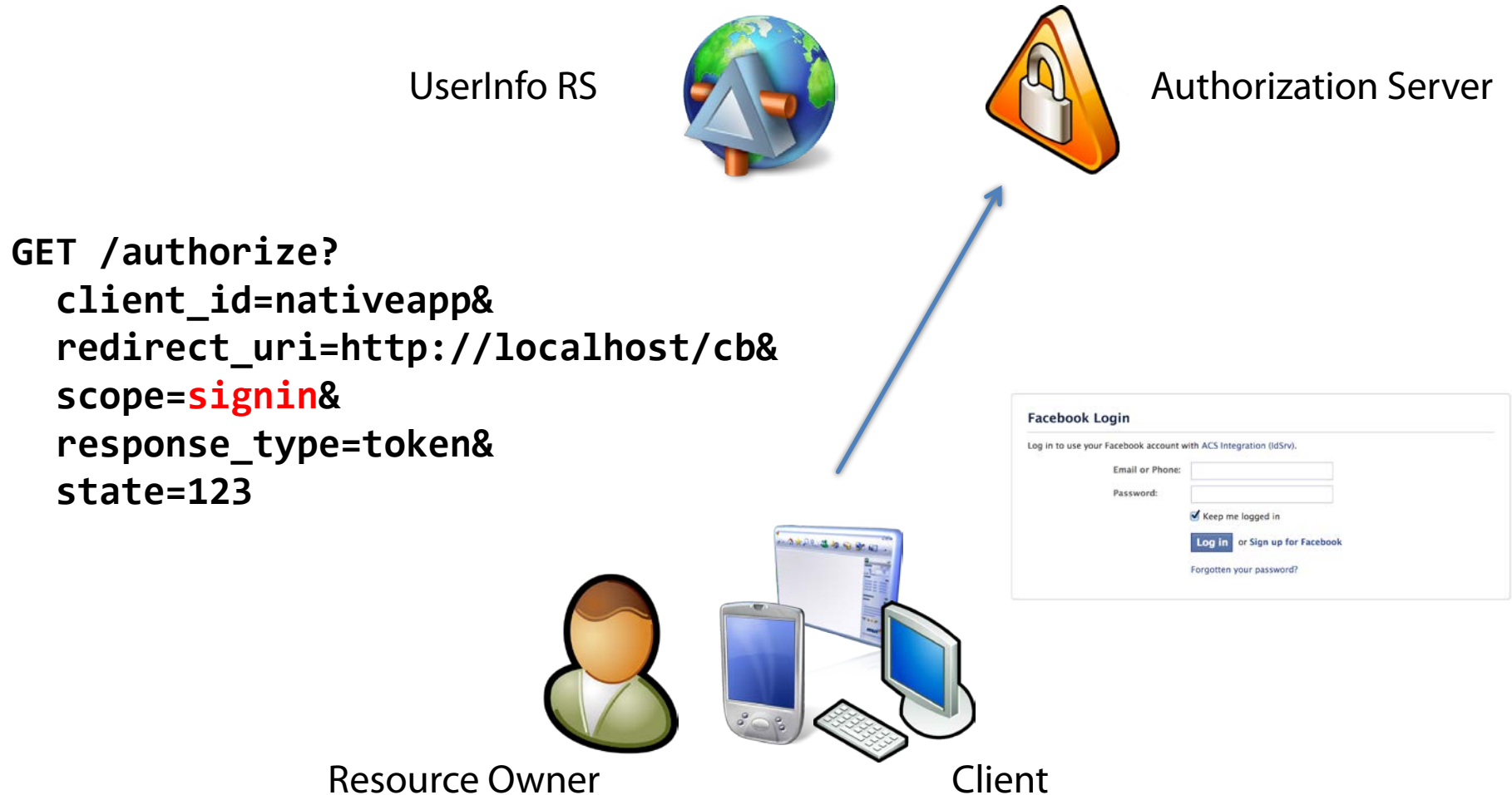
- **OAuth2 is for (delegated) authorization**
 - authentication is a pre-requisite for that
 - access token is for some back-end service
- **Sometimes you "just" need authentication**
 - (at least to begin with)
 - identify user in an application
 - control access to application features
- **OAuth2 is regularly "abused" for that**



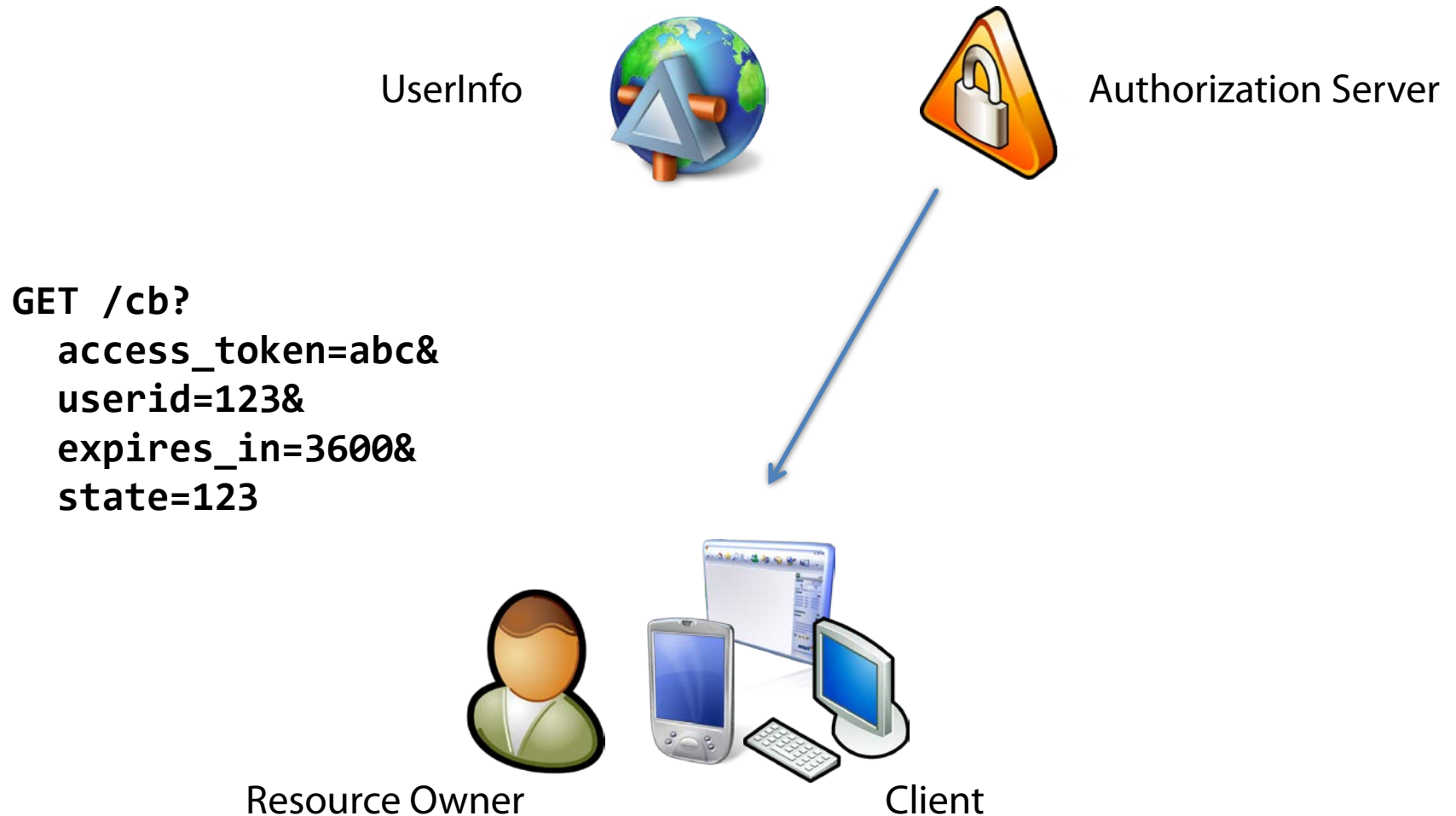
<http://www.thread-safe.com/2012/01/problem-with-oauth-for-authentication.html>

<http://www.cloudidentity.com/blog/2013/01/02/oauth-2-0-and-sign-in-4/>

OAuth2 for Authentication: Request



OAuth2 for Authentication: Response



OAuth2 for Authentication: Accessing User Data



The Problem



**userid,
access token**



**1. User logs into malicious app
(app steals token)**



**2. Malicious developer uses stolen
access token in legitimate app**

The Solution?



OAuth.io
OAuth that just works.

- 1 Setup your **Facebook** API Keys in OAuth.io

We give you a sandbox and a production **Public key**.

- 2 Setup OAuth.js in your HTML

```
<script src="path/to/OAuth.js"></script>
```

- 3 Request user authorization for **facebook**

```
OAuth.initialize('Public key');

//Using popup (option 1)
OAuth.popup('facebook', function(accessToken, err) {
  //handle error with err
  //use accessToken in your API request
});

//Using redirection (option 2)
OAuth.redirect('facebook', "callback/url");
```

We support 50+ OAuth providers.



Developers before OAuth.io

stubborn programmer

- 1 Go to official download page
- 2 Download the lib
- 3 Install the lib



Get early access to OAuth.io beta

Your email

Sign up

Follow @oauth_io 224 followers

powered by [webshell](#)



Google Developers

Google Accounts Authentication and Authorization X

Search

dbaler@gmail.com
Sign out

HomeProductsConferencesShowcaseLiveGroups

Google Accounts Authentication and Authorization +1 1.8k

Choosing an Auth Mechanism

- Authentication
 - OpenID
 - OAuth 2.0**
 - Mobile App Authentication
- API Authorization
 - OAuth 2.0
 - Login**

Using OAuth 2.0 for Login

Google APIs use OAuth 2.0 for authentication and authorization. You can also choose to use Google's authentication system as a way to outsource user authentication for your application. This can remove the need to create, maintain, and secure a username and password store.

Note: If you are planning to provide a "sign-in with Google" feature, we recommend using [Google+ Sign-in](#), which provides the OAuth 2.0 authentication mechanism along with additional access to Google desktop and mobile features.

Sign in with Google

This article describes how you can outsource user authentication to Google and gain access to a user's profile.

The Google endpoints described here align with the [OpenID Connect](#) specification, which at the time of this writing, is in early draft stage. For reference, the OpenID Connect specification is very similar to the OAuth 2.0 protocol. These Google endpoints will update as the specification

OpenID Connect Flows

- **OpenID Connect builds on top of OAuth2**
 - Authorization Code Flow
 - Implicit Flow
- **Adds some new concepts**
 - ID Token
 - UserInfo endpoint
- **..and some additional protocols, e.g.**
 - discovery & dynamic registration
 - session management

<http://openid.net/connect/>

OpenID Connect: The Players

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint



User Agent



Client

Step 1a: Authorization Request

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint

GET /authorize?
client_id=webapp&
redirect_uri=https://webapp/cb&
scope=openid profile&
response_type=code&
state=123



User Agent



Client

Scopes & Claims

- OpenID defines a set of standard scopes and claims

Scope	Claims
profile	name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, and updated_at.
email	email, email_verified
address	address
phone	phone_number, phone_number_verified
offline_access	requests refresh token

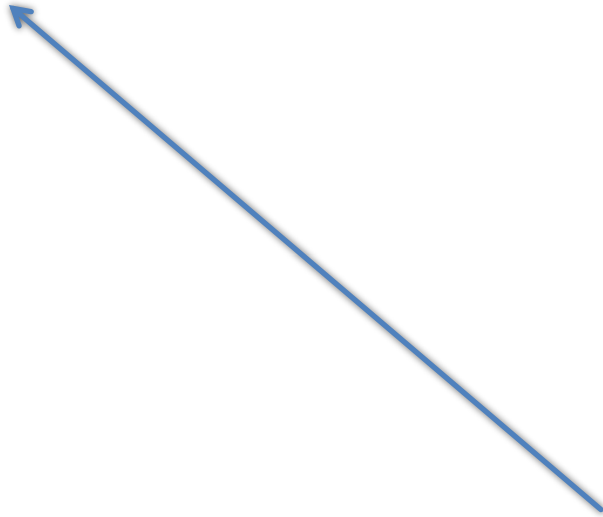
Step 1b: Authentication

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint



User Agent



Client

Step 1c: Consent

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint



Application **WebApp** asks
for permission to access your profile



User Agent



Client

Step 1d: Authorization Response

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint

GET /cb?
code=abc&
state=123



User Agent



Client

Step 2a: Token Request

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint

POST /token

Authorization: Basic (client_id:secret)

grant_type=authorization_code&
authorization_code=abc&
redirect_uri=https://webapp/cb



User Agent



Client



Step 2b: Token Response

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint

```
{  
  "access_token" : "abc",  
  "id_token": "uvw",  
  "expires_in" : "3600",  
  "token_type" : "Bearer",  
  "refresh_token" : "xyz"  
}
```



User Agent



Client

ID Token

- **JWT that contains claims about the authentication event**
 - Issuer (iss)
 - Subject (sub)
 - Audience (aud)
 - Expiration (exp)
- **Client must validate the ID token at this point**

Step 3a: UserInfo Request

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint

GET /userinfo

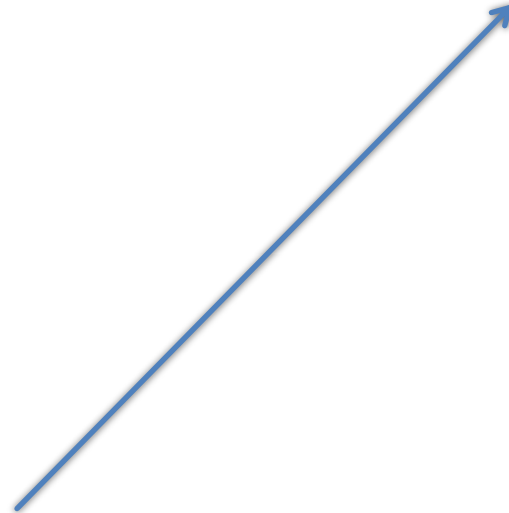
Authorization:
Bearer access_token



User Agent



Client



Step 3b: UserInfo Response

Identity Provider

Authorization Endpoint

Token Endpoint

UserInfo Endpoint

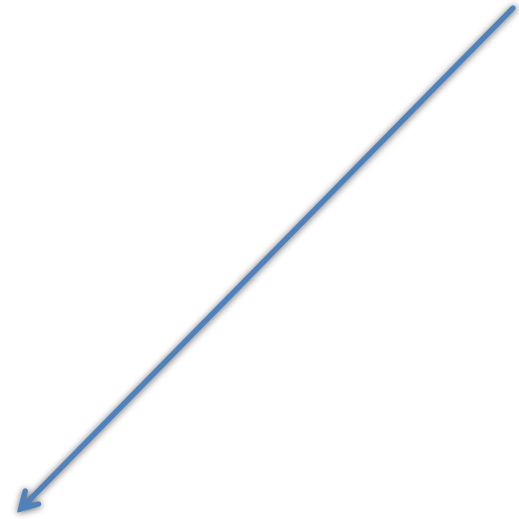
```
{  
  "sub": "248289761001",  
  "name": "Jane Doe",  
  "email": "janedoe@example.com"  
}
```



User Agent



Client



Summary

- **OpenID Connect standardizes how authentication with OAuth2 works**
 - standard scopes and claims
 - token type is JWT
 - ID token
 - UserInfo endpoint
- **Goal is to allow a client to use an arbitrary OpenID Connect provider without code modifications**
 - as opposed to how it works with homegrown OAuth2 authentication today
- **Not done yet, but "basic profile" pretty stable**
- **Additional specs are under development**