Access control, REST and sessions

Johan Peeters
independent software architect
Interactions with REST APIs are stateless

*each request contains all ... information necessary ... to understand the request*

Motivation:

1. scalability
2. processing need not understand interaction semantics (service orchestration)
3. services may be dynamically rearranged
4. cacheable
5. security

Fielding, PhD dissertation, p. 93
Stateless interaction means:
no sessions!
How do you do e-commerce without state?

Client

GET /products
read (list)

GET /products/123
read

POST /custs/456/orders
create

789
POST 123

xyz

/custs/456/orders/789
create
E-commerce is stateful

State is in

- the resources
- the client

but interaction is stateless.

No sessions!
What about access control?

Sensitive resources require a valid access token

Access token informs authZ decision
What is a valid access token?

Issued by an authorization server

5. call with access token

AuthZ Server

trust

/custs/456/orders/789
How does the client obtain an access token?

1. redirect URL
2. authN and consent dialog
3. 302 #access & identity token
4. #access & identity token
5. call with access token

trust

Browser

AuthZ Server

Client

/custs/456/orders/789
What if...

token is stolen?

client goes rogue?

user loses trust in the client?

...
Revoke the access token!

So-called reference tokens
Downside: additional round trip
Revocation requires detection,
so tokens should only be valid for a short period
Short-lived tokens

Shorter access token lifetime → smaller window of opportunity for attacker

but requiring the user to authenticate frequently is anathema

Could we use OAuth 2 refresh tokens?

Yes, but...
A novel idea:

sessions

between user agent and authorization server
This is great because...

users only have to log in once per session

only session implementation in the authZ server
But important problems remain

1. users should not even be aware of a new access token request
2. how do users log out?
Silent authentication

hidden iframe

- makes token request with `prompt parameter set to none`
- receives access token
- sends it to parent with HTML5 `postMessage()`

brittle?
Log out

OAuth revocation endpoint? Perhaps partially

Some authZ servers provide a proprietary /logout endpoint

3 OIDC drafts:

- back-channel logout
- session management
- front-channel logout

Struggling with Single Log Out
Conclusions and recommendations

- keep your APIs RESTful and stateless - thus no sessions
- sessions between client and authZ server avoid need to re-authenticate
  - caveat: silent authentication is clunky
  - perhaps refresh tokens are not so bad
- Single Log Out may be a good deal more complex than Single Sign On
About me

- Security architect
- Founder of secappdev.org
- Consultancy
- Bespoke development
- Lecturer at EhB

https://www.johanpeeters.com
@YoPeeters
yo@johanpeeters.com