OAuth 2.0 and Internet Standard Protocols

Torsten Lodderstedt

Deutsche Telekom AG

What shall we aim for?

"… make OAuth the authorization framework of choice for any internet standard protocol, such as WebDAV, IMAP, SMTP or SIP."

http://www.ietf.org/mail-archive/web/oauth/current/msg07758.html

- Why? Because it is
  - Secure
  - Easy to use
  - Scalable
  - General purpose (and by no means limited to 3rd party delegation)

OAuth 2.0 Adoption

- A lot of productive implementations exist ☺
- Standard protocols using OAuth 2.0
  - OpenId Connect
  - OpenSocial
  - Open Mobile Alliance RESTful APIs
  - UMA
  - …
- BUT perception of OAuth seems to be: best-suited for protecting deployment-specific APIs

Is there anything missing?

Life of a client – A Walkthrough

Example

- Access documents on a Website

https://www.example.com/

- using
  - CURL and
  - Web Browser
- BEARER authentication scheme

(1) discover the environment

1. End-user runs ***curl*** with some URL referring to his documents

```
curl https://www.example.com/documents/
```

2. Web server answers

```
HTTP/1.1 401 Authorization Required
WWW-Authenticate: BEARER
    realm="https://www.example.com/documents"
```

- What‘s next?
  - How does the client (gets to) know the authorization servers endpoint URLs?
  - How does the client learn the authorization server‘s capabilities?

(1) discover the environment (contd.)

- Discover the authorization server (Options)
  1. Resource‘s HTTP response may directly carry information
  2. Application protocol specific discovery
  3. Domain-specific discovery protocols
  4. Full-fledged, generic discovery protocol
- Discover the authorization server‘s capabilities
  - endpoint URLs
  - supported extensions (e.g. revocation or registration)
  - supported grant types

(1) discover the environment (contd.)

- Assumptions:
  - authorization: https://as.example.com/authz
  - token: https://as.example.com/tokens
  - grant types: resource owner password credentials and authorization code

What‘s missing?

- Discover authorization server

(2) Introduce client to server

- Anonymous client is the only available option currently
  - acceptable for resource owner password credentials (CURL)
  - but what about authorization code or implicit typically used by native and browser apps?
- Assuming the user now tries to access the documents using a browser, the user consent would look like

Some anonymous client is asking for permission to access your files at *https://www.example.com/documents/*

?

(2) Introduce client to server (contd.)

- User must be supported in co-relating application usage and authorization process, e.g.

Firefox is asking for permission to access your files at https://www.example.com/documents/

(2) Introduce client to server (contd.)

- Required data: name, URL, …
- How to publish this data? Some options:
  1. Dynamic client registration
     - would also allow to setup client id and secret (or other credential)
  2. Authorization request parameters
     - comparable to user agent header
  3. …

What‘s missing?

- Discover authorization server
- Publish client meta data

(3) request authorization

GET /authz?response_type=code&client_id=abc&
state=xyz &redirect_uri=cust://oauth&scope=???
Host: as.example.com

- What would be an appropriate scope value?

  scope=„GET“ or scope=„HTTP_GET“ or scope=„WebDAV_GET“?

- Would be consistent with today‘s standard practice!
  - Most implementations handle resources implictly, scopes represent API types, permissions, and/or operations
  - Viable option for single service providers and environments operating a single service per API/protocol type
- But what about web servers? (or mail servers, file servers, …)
- Moreover, it does not allow to control access to (sub)sets of resources, such as directories

(3) request authorization (contd.)

- What about this?

  scope= https://www.example.com/documents/#GET

- Respective authorization request:

  GET /authz?response_type=code&client_id=abc&
  state=xyz&redirect_uri=cust://oauth&scope=https%3A%2F%2
  Fwww.example.com%2Fdocuments%2F%23GET
  Host: as.example.com

(3) request authorization (contd.)

- Need to come up with a sustainable concept of how to use scopes (Options)
  1. Best practices document
  2. Design guideline
  3. Standard track document defining scope scheme for HTTP-based resources
  4. …

What‘s missing?

- Discover authorization server
- Publish client meta data
- Scope design guideline

(4) Access resources

- Let‘s go now … but wait, can the client really trust in www.example.com?
- How does it know this server is the legitimate consumer of the access token?
- What if it is a counterfeit resource server?

http://tools.ietf.org/html/draft-ietf-oauth-v2-threatmodel-01#section-4.6.4

- Threat prevention through well-known addresses and HTTPS server authentication no longer viable

(4) Access resources (contd.)

- Alternative threat prevention needed (Options)
  1. Put actual resource server‘s URL into token and validate on legitimate server
  2. Proof of possession (e.g. MAC)
  3. Auth server might verify resource server URL and, if required, refuse request
  4. Authz server might announce to the client the valid resource server endpoints
  5. …

What‘s missing?

- Discover authorization server
- Publish client meta data
- Scope design guideline
- Countermeasure against counterfeit resource servers