Macaroons and dCache

... or delegating in a cloudy world

Paul Millar

LSDMA AAI Topical Meeting (2015-09-30)
In LSDMA WP1 we've focused more on federated identity management, which is Authentication.

This talk is about the second 'A': Authorisation.
Quick recap: which is which?

Authentication

Authorisation
Authorisation without authentication?
DLCL use-case: photon portal

- User's web-browser
- Authentication
- Portal
- User DB
- HTTP/WebDAV
- Pool
- Storage fabric: dCache

Users

DLCL domain-specific layer

- Login
- Req. download
- Stream data
- Redirect
- Req. download
- Req. download
- Stream data
Desired: client downloads directly

How to authorise this?
One solution: a bearer token
What are bearer tokens?

**Bearer token** is something the user presents with a request so the server will authorise it. There's no interaction between client and server.

Examples of bearer tokens:

- HTTP BASIC authn, anything stored as a cookies.

Counter-examples:

- X.509 credential,
- SAML,
- Kerberos.
Bearer tokens for download authz

- Redirection should work **without JavaScript**,
- Simple: **embed token** in redirection URL.

http://webdav.example.org/path/to/file?authz=\(<\text{TOKEN}\>\\n
(There are nicer ways of embedding the token, but the URL is the only thing we can control)

- **Complete token** always sent with the request.
- What can we do to stop someone **stealing** this token?
  
  ... or make the token useless if they steal it.
Introducing Macaroons
Macaroons 101

- Macaroon is a **bearer token**.
- Macaroon contains zero or more **caveats**.
- Each caveat **limits** something:
  - **who** can use it, or
  - **what** they do with it.
- Anyone can **add** a caveat to a macaroon:
  - Create a new macaroon that is more limited.
- Nobody can **remove** a caveat from a macaroon.
Example: 3rd-party copy

Portal

Target Storage Service

Source Storage Service

only WRITE, only from <IP addr>, only for 10 minutes.

Rogue Service
3rd party caveats – extra cool!

- A 1st party caveat can be satisfied by the client.
- A 3rd party caveat requires proof from some other service; e.g.
  - only fred@facebook,
  - only members of VO ATLAS,
  - only if not part of a denial-of-service attack.
- The proof is another macaroon: a discharge macaroon.
Example: download w/ 3rd-party caveat

(some details have been glossed over)
Discharge macaroons

• The client proves it satisfies a 3rd party caveat by having a **discharge macaroon**.

• The original macaroon is only useful with a **valid** discharge macaroon.

• The discharge-macaroon can have **caveats**:
  
  Short-lived discharge macaroon can be used to simulate X.509’s certificate revocation list.
  
  The discharge macaroon can have 3rd-party caveats.
Solution revisited: macaroons

User's web-browser

Authentication

Portal

Add caveats to macaroon

User DB

HTTP/WebDAV

Pool

Req. download

Redirect

Req. macaroon

Supply macaroon

Stream data

Login
For what else are macaroons good?

Private Sharing!
Enabling sharing: a new interface

- **Create** a macaroon:
  Need to know the macaroon to access the file.
- **List** macaroons:
  Facilitate sharing files.
- **Facilitate** adding caveats:
  Purely in-browser or server-side?
  Third-party caveats? (e.g., member-of-ATLAS caveat)
- **Destroy** macaroons:
  Unclear if this really makes sense.
Thanks for listening.