dCache.org

Macaroons and SciTokens:

Paul Millar
(on behalf of the dCache team)

dCache Workshop 2019 at Madrid, Spain; 2019-05-21
https://indico.desy.de/indico/event/22170

Nordic e-Infrastructure Collaboration

eXtreme DataCloud

LSDMA

Fermilab

DESY.

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

Macaroon cheat sheet

- Bearer token: easy to give to someone
- Available in all supported dCache versions
- Any user can request a macaroon
  - Available via WebDAV door
  - User-friendly version available in dCacheView

    (dCache v4.2 or newer)

Macaroon current usage

- **SurfSARA**,

  (see Onno's talk)
- **ATLAS/University of Oslo**,

  BOINC client uploads results using a macaroon
- WLCG **HTTP-TPC**,

  HTTP client (embedded within HTTP server) is given a macaroon to authorise the transfer.
- DESY's **EOSC-FaaS compute platform**

  Event describing the arrival of a new file includes macaroons to read data and upload results.

Recent changes: Issue ID

- Previously, the logged ID was from the macaroon's hash (e.g., **`1lG2P7wH`**)
- Turns out that the hash isn't a great ID
  - Problems for auditing macaroon usage, prevents revocation
- Solution in dCache v5.1, the Issue ID (iid) caveat.
  - Optional, macaroons without an iid still work.
  - Must be first caveat, if present.
  - Value is a unique ID for this macaroon.
- dCache now logs a compound ID: iid and hash (e.g., **`uMvbGcWH:clhxzw0t`**)

Recent changes: macaroon request logging

- Support back-ported to stable branches in October 2018.

```
level=INFO ts=2019-05-19T17:50:28.988+0200
event=org.dcache.webdav.request request.method=POST
request.url=https://prometheus.desy.de:2443/VOs/dteam/tpc-
test/https/domatest/file25_fb5546d2-49d2-4fd3-b15b-f94a02f0448a
request.macaroon-request="{\"caveats\":
[\"activity:DOWNLOAD,LIST\"], \"validity\": \"PT157M\"}"
response.code=200 response.macaroon-id=uMvbGcWH:clhxzw0t
socket.remote=[2001:1458:d00:14::14d]:56000 user-
agent="x509_token_issuer/@devel@ neon/0.0.29"
user.dn="CN=1558242063,CN=2583931595,CN=1913499342,CN=Thomas
Beermann,CN=722011,CN=tbeerman,OU=Users,OU=Organic
Units,DC=cern,DC=ch" user.mapped=1001:1001
```

Possible future macaroon changes

(feedback, please)

Possible future changes: client ID

- **Opaque** (to dCache) data
- Multiple caveats of this type are allowed

  dCache **just ignores** all these values
- Holds (very short) client-specific metadata

  Could use the Issue-ID to bind data to macaroon.
- Possible uses:
  - Rucio/FTS: record some internal state.
  - Dynafed: to record from which endpoint is the macaroon.

Possible future changes: secret management

- Create separate set of secrets **per user**.

  Removing a secret affects only one user
- Update **secret creation** policy:
  - When creating a secret, duration a fixed amount larger than macaroon
  - Prefer creating a new secret if the lifetime of the existing secret (with the shortest remaining lifetime) is much longer than the macaroon.

Possible future changes: invalidation

- Want to allow users to **invalidate** their macaroons
- Three ways to invalidate a macaroon:
  - a) **Record the IssueID** (and secret ID) in a list,
  - b) Remove **specific secret** for this macaroon,
  - c) Remove **all user's secrets** created before specific time.
- Choose **least destructive** option, given what the client presents.
- Requires **"real" authentication**, not presenting a macaroon.

Possible future changes: macaroon store

- When creating a macaroon, optionally ask dCache to **remember it**.

    Number of stored macaroons will be limited.

- Allow a user to **query** what macaroons they have created (maybe with a short note, explaining why)
- Can use this information to facilitate **invalidation**.

SciTokens

SciTokens: the problem

- Grid is currently **X.509 based**, using VOMS to identify groups of people
- **Mapping** between X.509 and site-local resource identifiers (typically uids / gids) is "often lossy and difficult to predict".
- Updating **permissions** can be a slow, manual process.
- Jobs often run with delegated credentials that have **full permissions**
  - A stolen credential can do anything the user is allowed to do.

SciTokens: the solution

- VOs will have a **central service** that authorises user activity.

  This is **NOT** VOMS: it does *not* assert group membership

- The central service can be as **fine-grain** as desired; e.g., authorising users on a file-by-file bases.
- The central service **issues tokens** ("SciTokens") that say what a user is allowed to do.
- Services **verify** that the presented SciToken is valid, from a trusted SciToken server, and that the requested operation matches what the user is allowed to do.
- In reality, SciTokens are just **OAuth2** tokens with some restrictions :-)

SciTokens: under the hood

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiI
sImtpZCI6ImtleS1yczI1NiJ9.eyJpc3MiOiJ
odHRwczovL2RlbW8uc2NpdG9rZW5zL
m9yZyIsImV4cCI6MTU1MjQzMzEwMSwi
aWF0IjoxNTUyNDMyNTAxLCJuYmYiOjE
1NTI0MzI1MDEsImp0aSI6IjRmOWE1MD
U2LTZhMzMtNDgwMC1hYzgyLTJlYmQ3
YWM3NzVmZCJ9.a1vDM243xUow6vSa
VEgv3QAfzIxqqw2TqfPhu4HeOCk-
3USiFmEzgAMuOPIzZdp5V6q4n2vlHmE
G2idp2eRt0T5G95DiKJ4B79OD0M3wlvB
mciWrmDOWXLPqvOYz08fCKRq2RgLkh
9-K43qsAdgl-
szD_3QcoFzoxKHOm3wTlfPCTSAYQzJt
N67Lv65tCsUNz-OR4gvobGoh

HEADER
(crypto parameters)

```
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "key-rs256"
}
```

PAYLOAD
(claims)

```
{
  "iss": "https://demo.scitokens.org",
  "exp": 1552433101,
  "iat": 1552432501,
  "nbf": 1552432501,
  "jti": "4f9a5056-6a33-4800-ac82-2ebd7ac775fd"
}
```

SIGNATURE

SciTokens: under the hood

```
{
  "iss": "https://demo.scitokens.org",
  "exp": 1552433101,
  "iat": 1552432501,
  "nbf": 1552432501,
  "jti": "4f9a5056-6a33-4800-ac82-2ebd7ac775fd"
}
```

Claims in this token

iss	which server issued this token
exp	when the token expires
iat	when token was issued
nbf	token is not valid before
jti	unique identifier for this token; for replay protection, traceability.

Some other possible claims

sub	an identifier (possibly opaque) for the user
aud	audience: which service is supposed to accept the token.
scp	(scope) what operations are allowed
scope	new, preferred way of writing scope claims

SciTokens: under the hood

Scope operations

`read`	Read data from a resource
`write`	Write data into a resource

Path qualifiers

`read`	Read data from within /
`read:/foo`	Read data from within /foo
`write:/data/dir1`	Write data into /data/dir1

Combining elements in scope

`read write:/results` Can read all files, but write only into `/results`

```
{
  "iss": "https://demo.scitokens.org",
  "scope": "read write:/results",
   [...]
  "jti": "4f9a5056-6a33-4800-ac82-2ebd7ac775fd"
}
```

SciTokens and WLCG

- WLCG **AuthZ working group** is investigating token-based approach for WLCG
- Closely based on SciToken, but making some **recommendations**; e.g., include a user identifier in token, to support user-level banning.
- SciToken will likely being **updated**, based on these recommendations.

SciTokens: support in dCache

- Fetch (and cache) **public keys** from the issuer, SciTokens are verified "offline".
- dCache supports **multiple issuers**. The token describes from which issuer it is.
- Tokens from one SciToken issuer are mapped to a **fixed set** of principals.

This is equivalent to a VO has a single "group account".

```
gplazma.scitoken.issuer!dteam = https://dteam.org
[…] user:dteam
```

SciTokens: ‘aud’ claim support in dCache

- OAuth2 has an optional `aud` (audience) claim.
  - Describes which service should accept this token
  - Provides some protection against a token being used where it shouldn’t
- By default, dCache accepts all tokens, whether or not they have an `aud` claim.
- The SciToken plugin may be configured to accept only specific `aud` claim values. The configuration value is a space-separated list.

```
gplazma.scitoken.audience-targets = dcache.example.org
```

SciTokens: ‘jti’ claim support in dCache

- OAuth2 has an optional `jti` (JWT ID) claim, value is unique.
- Used for audit / problem tracking
  - dCache records the `jti` as a principal.
- Avoid replay attacks:
  - Store last n values, configured globally or per issuer

```
gplazma.scitoken.token-history = 0

gplazma.scitoken.issuer!dteam = https://dteam.org
/data/dteam -tokenHistory=100 […]
```

  - SciTokens probably requires this switched off!

SciTokens: 'sub' claim support in dCache

- OAuth2 has an optional `sub` (subject) claim.

  Identifies the user, but may be pseudonomous.

- Used for audit / problem tracking
  - dCache records the `sub` as a principal.
- dCache requires a SciToken to have either a `jti` or a `sub` claim (or both).

SciTokens: ‘scope’ claim support in dCache

- OAuth2 has an optional **`scope`** claim.

  If present, this should limit what activities are allowed; for example,

  `read:/public read:/users write:/users/paul`

- We need to map these scope paths to dCache path.
- Simple approach: just add a prefix.

  For example: prefix `/data/dteam` + scope path `/public` → `/data/dteam/public`

- Configured as part of the issuer:

```
gplazma.scitoken.issuer!dteam = https://dteam.org
/data/dteam […]
```

SciTokens: next steps

- SciToken support is included in **dCache v5.1**
  - Simple testing by Brian B (and myself): it works.
  - Prometheus is configured with SciToken support for dteam and CMS.
  - I'm trying to get prometheus tested with other software.
- **Other software** is being updated to support SciTokens
  - Condor is the next target.
- Will likely see **changes** from WLCG AuthzWG.
- Might need to support **issuing macaroons** from SciTokens.

Thanks for listening!