Pre-Signed URLs for the grid

Christophe Haen
17/05/2023
DOMA BDT

About tokens

- We move to them because they are industry standard
- And they are great for what they are meant for, we should use them !
- But the industry does not use them for files access/transfers !
  - Why would we ?
- To date, we still don’t know exactly how they would work
  - Permission granularity undecided
  - JWT schema flaws
  - FTS/DIRAC/Rucio token exchange mechanisms
  - IAM scalability: order of kHz
  - Capability/role based approach ?

Pre-signed URLs

- From AWS docs: *"By default, all S3 objects are private. Only the object owner has permission to access them. However, the object owner can optionally share objects with others by creating a presigned URL, using their own security credentials, to grant time-limited permission to download the objects."*
- Another AWS docs: *"In essence, presigned URLs are a bearer token that grants access to those who possess them."*
- Simply put, the authorization to perform specific actions (download, upload, etc) become part of the URLof the file, and this can just be passed around.
  - The signature can also go in the header
- THAT is the industry standard for file access/transfers
  - Same techno when sharing files with OwnCloud, Google docs, etc
  - AWS S3 is the defacto standard (V4 signing process)

How it works (very imprecise, but to get the idea)

- Proper example
- A client (e.g. DIRAC) has credentials (e.g. AccessKey/SecretKey) to a storage (e.g. Gridka disks)
- A User (e.g. Batman) wants to access a file at Gridka https://gridka.de/lhcb/batman/myIdCard.jpeg but with no credentials
- Batman asks DIRAC a URL to download this file
- DIRAC knows Batman, so forge a presigned URL for him
- https://gridka.de/lhcb/batman/myIdCard.jpeg?accessKey=<diracAccessKey>&signature=<giberish>
  - The signature is basically {“timestamp”: “2023-06-01 12:00:00”, “verb”: “Get”} hashed with the DIRAC secret key

Advantages of presigned URLs

- REAL industry standard for data
- No complex token exchange procedures
- Requires much less development work from FTS/DIRAC/RUCIO
- Makes the integration of Commercial Cloud Storage in our grid world trivial
  - They would be just another storage, as they use pre-signed URLs
- Reduces possible frictions/conflicts of profiles between
  - DIRAC/Rucio
  - IAM/EGI Checkin
- Remove huge load from IAM
  - URLs are signed by the client directly, and they are cheap

Advantages of presigned URLs (2)

- Manage permission from the namespace (DIRAC/Rucio) not OIDC provider
- No more discussion on the model

Presigned URLs are not the silver bullet

- They are not meant for every use cases
  - XCache proxy
  - Xroot redirector
  - ?
- But as far as standard data operations are concerned, they are the go-to solution
  - FTS would need to support only this
  - Let’s try it !!

Yes, I know...

- This proposal comes late
- A lot of work has been done at the storage level for token

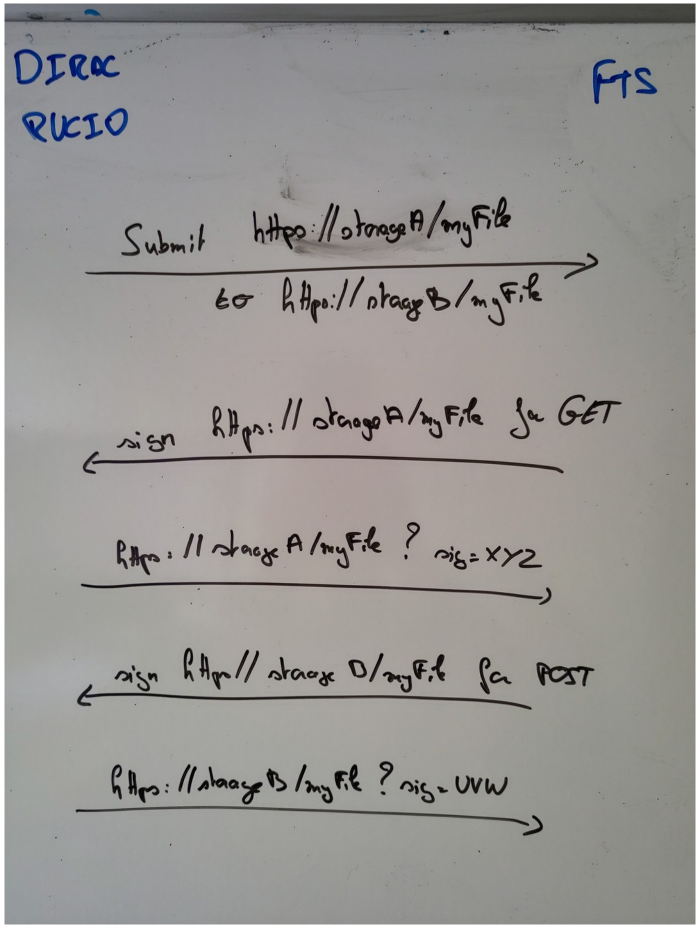
How to make sure we fail

- Take a long time to implement
- Solve everything with it
- Incompatible with S3
- Invent new signature algorithms
- Make differences between DIRAC & Rucio

Ideal bucket list

- Direct upload
- Direct download
- Direct removal
- Check metadata
- Third Party Copy
- Tape operations
- Xroot stream read from job directly from the end storage

How it could look like in the end


DIRAC
RUCIO
FTS
Submit https://storageA/myFile
to https://storageB/myFile
sign https://storageA/myFile for GET
https://storageA/myFile?sig=XYZ
sign https//storageD/myFile for POST
https://storageB/myFile?sig=UVW

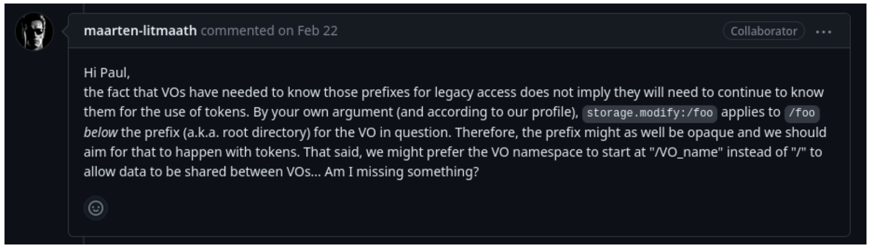
- FTS could
  - Have no storage credentials, and ask DIRAC/Rucio to sign on the fly (preferred solution ?)
  - Have storage credentials and use them
  - Have storage credentials and sign its own URLs
    - Why would it ?
- A job would just ask DIRAC/(Rucio) to sign

How do we get there pragmatically

- Stick as much as possible to the existing solutions
- Stick to HTTPs to start with
- Use AWS V4 signature and Boto3 library
  - Ensure we keep compatibility with cloud transparently
  - Support pre-signed URLs for all our basic operations (upload/download/remove/stat)
- DIRAC/Rucio should offer a unified and minimalist interface to FTS:
  - Same interface as the library to make transparent to FTS the use of local credentials ?
  - Different interface to reduce the number of calls
  - Shared secret for auth between FTS and DIRAC/Rucio

Few pitfalls

- AWS V4 expects (bucket,key), we have path.
  - Maarten's proposal makes a lot of sense to me (Bucket: SE/RSE, Key: LFN)
  - Host header http only :-(
  - Echo style ?
  - Fake bucket, full path as key ?
- TPC is not an S3 concept
- Tape operations
  - Can we get away by just signing each url in the requests ?
  - Maybe AWS Glacier Deep Archive has the solution ?
  - Keep it for a second step


maarten-litmaath commented on Feb 22 Collaborator

Hi Paul,
the fact that VOs have needed to know those prefixes for legacy access does not imply they will need to continue to know them for the use of tokens. By your own argument (and according to our profile), `storage.modify:/foo` applies to `/foo` *below* the prefix (a.k.a. root directory) for the VO in question. Therefore, the prefix might as well be opaque and we should aim for that to happen with tokens. That said, we might prefer the VO namespace to start at "/VO_name" instead of "/" to allow data to be shared between VOs... Am I missing something?

Proposed goal

- “Be able to perform by the 01/09/2023 a third party copy between 2 disk storages using FTS and pre-signed URLs. FTS should ask DIRAC/Rucio for signature. DIRAC/Rucio should use BOTO to perform the operation.”
  - Agreed by FTS, EOS, DIRAC, Rucio providing the community shifts efforts from token-based to presigned urls based solution
- If we can do that, there’s no reason why we can’t fully transition to pre-signed URLs for the use cases we listed earlier
- If for this goal, we do anything not standard (except the TPC), we do it wrong
- Relatively low hanging fruit, but allows to setup basic infrastructure, and more independent tests can take place (e.g. scaling)

Conclusion

- Presigned URLs addresses the great majority of our use cases
- They can be implemented in a much shorter time scale than tokens
- We should not miss the opportunity to adopt a real standard and ease commercial cloud transparent integration
- LHCb successfully runs a similar pattern in non grid context with EOS tokens