Early recommendations from the Token Trust and Traceability WG

CHEP 2024, Krakow

Matt Doidge, on behalf of the TTT

What is the Token Trust and Traceability WG? (TTT)

- Instantiated in Summer 2023 with stakeholders from various distributed computing communities; **EGI**, **WLCG** and **AARC**
  - Other groups welcome, some **OSG** participation but meeting timing is difficult.
  - We admit to having a stronger WLCG/Europe bias then we would like - this is unintentional.
- The “spiritual successor” to the **WLCG Traceability and Isolation WG**, with the focus on the new challenges that comes with the move to tokens.
  - The requirements for **traceability** have not fundamentally changed - just the mechanisms involved.
  - The **Isolation** aspect of the previous WG is “solved”, so we don’t need to consider that.

Aims of the TTT.

- The goal of the TTT is to formulate best practice for all aspects of the use of tokens, and inform all parties involved in token use:
    - Service Providers, Token Issuers, Developers, User Groups and beyond.
- This will be done by formulating **Recommendations**, **Procedures** and producing **Policy** Documents.
- Working alongside other groups in the “token space”:
    - The WLCG AuthZ WG and the Grand Unified Token (GUT) WG.

Intermission: What is a Token? What do they replace?

They can be of different types, most common are **Access** Tokens, **ID** Tokens and **Refresh** Tokens.

In our context an Access Token is (usually) a **JSON Web Token** (**JWT**) - a string used as a means of authentication, that string containing 3 base64 encoded blobs, including a JSON describing various **attributes** and a **signature**.

The structure of this JSON matches a **profile**, describing what **capabilities** the token grants.

Access Tokens are given out and signed by an **Issuer**. The Issuer checks if the request is valid for that requestor.

Within the WLCG and other grid environments they replace the **X.509 proxies** that were widely used for many, many years.

```
$ echo $token | cut -d. -f2 | base64 -d | jq
base64: invalid input     # the bits that don’t neatly fit
{
  "wlcg.ver": "1.0",                            # profile version
  "sub": "abc123-ef45-100d-ab23-bedabcdfg", # user (my) ID
  "aud": "https://wlcg.cern.ch/jwt/v1/any", # audience (*)
  "nbf": 1728480846,                            # not before
  "scope": "storage.create:/ storage.read:/",
  "iss": "https://dteam-auth.cern.ch/", # Issuer
  "exp": 1728484446,                            # expiry
  "iat": 1728480846,                            #issued at
  "jti": "0abcdef12-0123-4567-90ab-5678abcd1234", # unique
  "client_id": "fedab01-fed034-0123-a1b2b5"   # requestor
  “wlcg.groups”: [
       “/dteam”,
       “/dteam/NGI_UK”
       ]                                  # group info (made up)
}
```

“Illustration” of an example access token structure.

Differences between OAuth2 tokens and X.509 Proxies

- X.509 proxies are long lived (~0.5-8 days) **“all access passes”,** a user’s proxy gives access to anything the generating certificate holder has permission to.
    - VOMS roles and groups did manage to provide some more granularity.
    - Emphasis on the service endpoint (CE, SE) making the final authorisation decision.
    - Need to be passed around the grid, heavily exposed, but revocable (via the end-entity certificate).
- Access tokens are more fine grained in the authorisation they grant.
    - **Tunable** to activities or workflows, reducing authorisation “over-reach”.
    - (Broadly) **Scopes** describe **what** a token is authorised to do, and **Audiences** describe **where** it allowed to do it.
- There are also **refresh** tokens, complementing access token flows.
        - Refresh tokens are typically smaller than access tokens, and often work “behind the scenes”, embedded within a workflow. They are linked to a specific client and allow it to get new access tokens.
        - The ability to refresh can allow a reduction in access token lifetimes.

Framing the Issues

- JWTs are **not** an AAI panacea
  - Easy to over restrict, or conversely over extend, a token’s purview.
  - **Token introspection** of a token this can lead to stress on the Issuer from the introspection requests of a busy workflow (and require 100% connectivity), but tokens are revocable by the Issuer.
  - **Offline verification** removes this stress, but creates a **need** for a **shorter lifetime** as this means the tokens are **non-revocable** - more frequent refreshes are required for longer workflows.
    - emphasis on **non-revocable**.
  - **Scopes** shift the onus of authorisation decisions (for example who has access to a path) from the resource provider to the token issuer.
    - Possibility for interactions between service configuration and token leading to unexpected behaviour.
    - Conversely possible for an Issuer to hand out scopes too trustingly.
  - Token revocation and user banning not straightforward processes, and revocation is not always possible,
- Optimisation Problem
  - Tokens provide many variables to interact with.
    - With many more types of tokens and flows then we had before with X.509 proxies.
  - No “one size fits all” - even within a community.

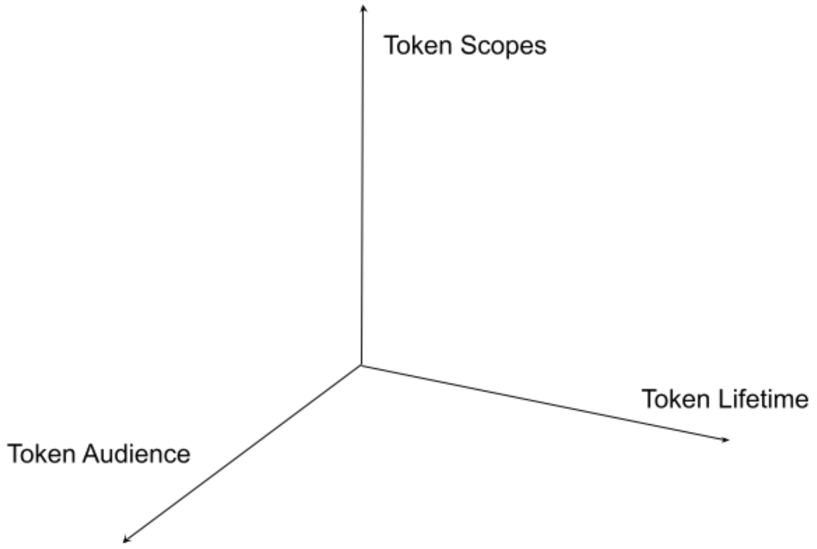
Orthogonal(-ish) Axes: What, Where, How Long For.

Whilst not capturing everything, a useful way of visualising some of the **tunable token attributes** (but this is more than a 3-Dimensional problem).

The “goal” is to get a vector in “token-trust space” with as small a magnitude as you can and still meet your **operational needs**. The closer to the “origin” the better.

These considerations are made **per workflow**, and are ultimately a form of **risk analysis**.

X.509 proxies would exist almost “off the charts” on all 3 axes!



The “units” of the axes correspond roughly to:

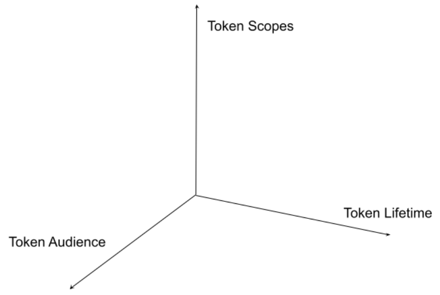
- “Broadness” and “Power” of Scopes.
- Number and “Sensitivity” of Audiences
- Time

Token Risk Analysis

There is no “one size fits all” solution - the most logical approach will be to produce a framework to facilitate **Risk Analysis** and Management **per workflow**.

Two hypothetical examples:

- **read only** access to a set of files.
    - Risk dependent on the sensitivity of the data
    - e.g. “Innocuous” data at a specific site could have a very long access token lifetime.
- **delete** access to data
    - Inherently higher risk, again dependent on the uniqueness/importance of the data.
    - Stricter Audience, Strict Scope, Shorter Lifetime.



Analysing risks from **Data Management** workflows are the “low hanging fruit” for these exercises. Studying the Impact for other resources (such as compute) is less intuitive, but equally important.

Token Lifetime

The AARC Community have been working on: **AARC-G081 “Recommendations of Token Lifetimes”**.

- This one “variable” stands out as it is universal and, on the surface, easily tunable.
- A rough rule of thumb, the **higher the risk the shorter you want the token lifetime**.
    - But Risk comes from many factors - from the chances of credential exposure, whether that token is revocable, and the scope and capabilities of that token.

However:

Lifetime has a direct effect on the **load** caused to the Issuing service, so the above considerations cannot be taken in a vacuum.

As noted in the document and discussed several times within our own meetings, within the grid we’ve worked for years with long lived, although revocable, proxy lifetimes - there’s plenty of room for improvement.

Traceability and Logging

The requirement for **Traceability**, detailed by our predecessor WG, remains.

- e.g. a **Resource Admin** needs to be able to deduce **who** accessed their resource, **what** actions were performed, and **where** the access comes from.
    - At the very least to the extent of being able to control access to, for example, a subject/user combination
    - This is as much a requirement on, for example, the individual middleware developers.
    - Need for clear logging of token transactions.
- There are also mundane **accounting** considerations.
    - Vital for services to know what to do with the token.
    - Requires well known "hooks" or **identifiers** within the token.
- Within WLCG the most commonly used of these is simply the **Issuer**.
    - **groups** exist as a concept but there's still conversations being had there.
    - In the event of an Incident, there is the requirement on the issuer to "fill in the blanks" ("actual" user ID etc) from the information provided by sites.

Information such as a specific user's identity might not be available within a token (only a sub-id), so the Issuer may need to be contacted if this is required - need a process for this.

As an aside, a pitfall of tokens being reasonably **short strings** is that they're easily **leaked**, via sharing, log exposure or recording in the wrong place.

Revocation, Banning and Incident Response

With X.509 we had a few banning mechanisms.

- Certificate Revocation Lists (CRLs) and within WLCG/EGI the Argus servers as well.

Within the token paradigm much of the ability to identify and act upon a user (e.g. ban them) is with the Issuer as the first point of contact.

- Need to define the **responsibilities** of the **Issuer** in this context. Issuers need to be aware of this and have in place:
  - Contact Policies
  - Incident Response Procedures
  - User Banning and Revocation Processes (such as client credentials and refresh tokens)
- All of which need to be defined - although some existing Issuers have established working methods (and a good track record).
  - Some token implementations have methods to store contact (and policy) information within the token metadata.
- And need to take into account **if** a token is revocable.
  - May be limited to preventing the refresh of a “bad” token.

Trust and Education

An important factor of **Trust** is **Understanding**

- Humans tend not to trust what they don't know.
- Tokens are still viewed as new and mysterious in many groups.
- For WLCG sites, encourage hands on experience, for example using the **dteam IAM server**
    - Requires some adjustments to the current documentation which is dev-focussed.
- Service configuration to enable tokens still in flux
    - Many areas of best Practice are yet to be defined.
    - Onus of this on the developers.
- Methods to enable inspection of tokens.
    - These exist, simply a matter of documenting.

Need to **develop**, **collate** and **curate** documentation.

In Summary

- Basic requirements for Trust and Traceability have not changed with the move towards Tokens, the methods available to achieve these aims have.
- Tokens provide multiple, orthogonal attributes to consider: including **Scope**, **Audience** and **Lifetime**.
    - As well as other factors for consideration, such as token location, storage and transfer.
- Best practice for Tokens, due to their “tunable” nature, needs to be defined **per workflow** and after a process of **Risk Analysis**.
    - The details of such a process are still being laid down, but progress has been made, in particular with **Token Lifetimes**.
- “Token-aware” Incident Response procedures, including the roles and responsibilities of Token Issuers.
- Awareness of the risks of Token Exposure, both by admins and devs.
- Need for Good Documentation throughout the token space.

Round up

The TTT is still in its early days, and welcomes more participation if you'd like to join in or just have questions for us. The TTT currently meets once a month, targeting the 4th Tuesday.

We're just looking at the tip of the iceberg of what needs to be done.

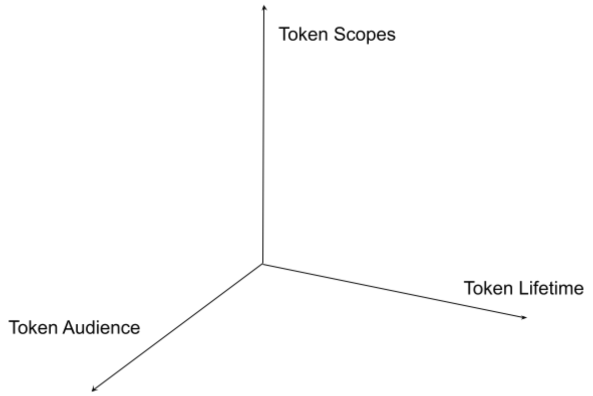
"With Tokens we have the potential for better AAI within our infrastructures than we have ever had in the past."

github: https://github.com/TTT-WG/TTT-WG

cern egroup: token-trust-and-traceability-wg

or contact Matt directly.

THANK YOU FOR YOUR TIME!


Token Scopes
Token Lifetime
Token Audience