The SciTokens Authorization Model: JSON Web Tokens & OAuth

Jim Basney <jbasney@ncsa.Illinois.edu>

Brian Bockelman <bbockelm@cse.unl.edu>

This material is based upon work supported by the National Science Foundation under Grant No. 1738962. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

SciTokens Project

- The SciTokens project, starting July 2017, aims to:
  - Introduce a ***capabilities-based*** **authorization infrastructure** for distributed scientific computing,
  - Provide a **reference platform**, combining CILogon, HTCondor, CVMFS, and XRootD, and
  - **Implement specific use cases** to help our science stakeholders (LIGO and LSST) better achieve their scientific aims.

Identity-based Authorization

- At the core of today's grid security infrastructure is the concept of *identity* and *impersonation*.
  - A grid certificate provides you with a globally-recognized identification.
  - The grid proxy allows a third party to impersonate you, (ideally) on your behalf.
  - The remote service maps your identity to some set of locally-defined authorizations.
- We believe this approach is fundamentally wrong because it exposes too much global state: identity and policy should be kept locally!

Capability-based Authorization

- We want to change the infrastructure to focus on *capabilities*!
  - The tokens passed to the remote service describe what authorizations the bearer has.
  - For traceability purposes, there may be an identifier that allows tracing of the token bearer back to an identity.
  - Identifier != identity. It may be privacy-preserving, requiring the issuer (VO) to provide help in mapping.
- Example: “The bearer of this piece of paper is entitled to write into /castor/cern.ch/cms".

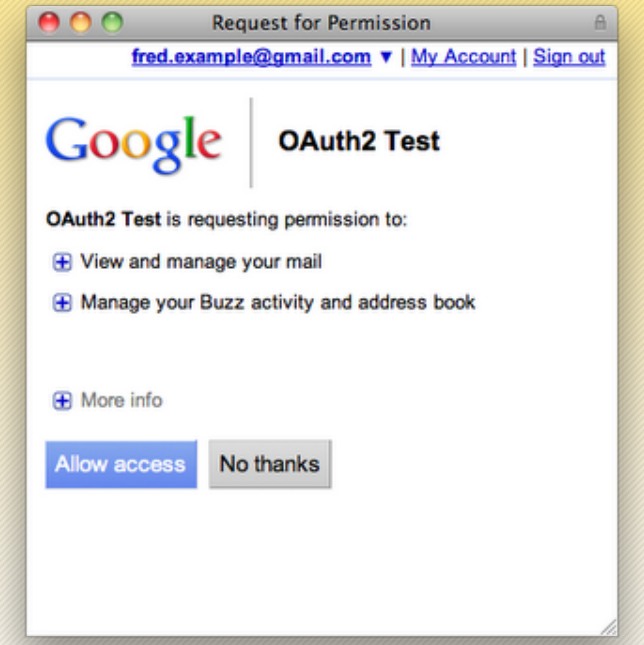
Capabilities versus Impersonation

- If GSI took over the world, an attacker could use a stolen grid proxy to make withdrawals from your bank account.
- With capabilities, a stolen token only gets you access to a specific authorization (“stageout to /store/user at Nebraska”).
- SciTokens is following the **principle of least privilege** for distributed scientific computing.

The World Uses Capabilities!

SCI TOKENS

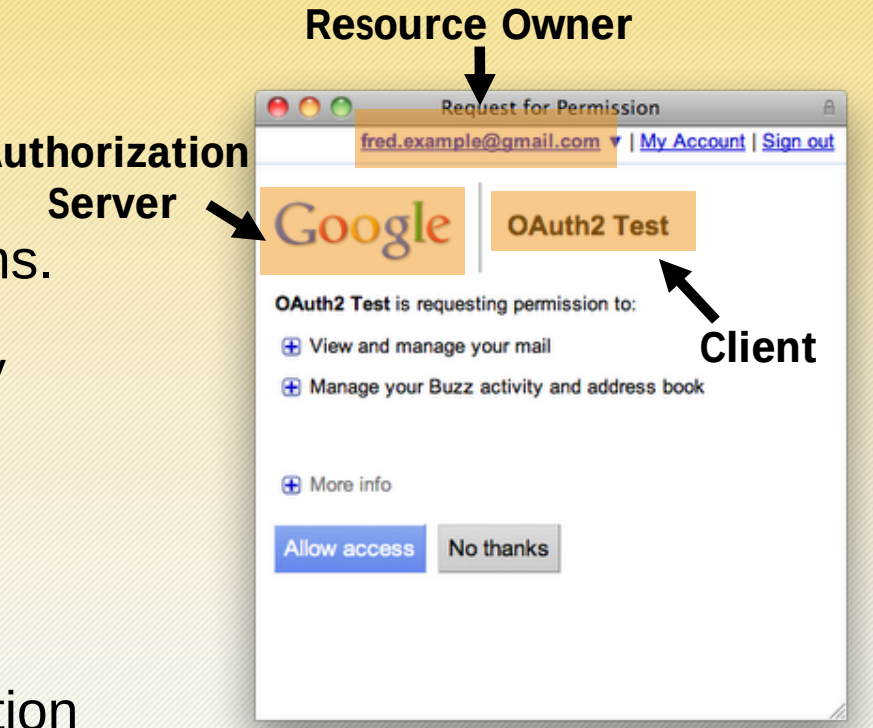
- The rest of the world uses capabilities for distributed services.
  - The authorization service creates a token that describes a certain capability or authorization.
  - Any bearer of that token may present it to a resource service and utilize the authorization.
- The primary way this is implemented is through OAuth2.
- When you click “allow access” on the right, the **client** at “OAuth2 Test” will receive a token. This token will permit it to access the listed subset of Google services for your account.
- OAuth2 is used by Microsoft, Facebook, Google, Dropbox, Box, Twitter, Amazon, GitHub, Salesforce (and more) to allow distributed access to their identity services.



Three-Legged Authorization

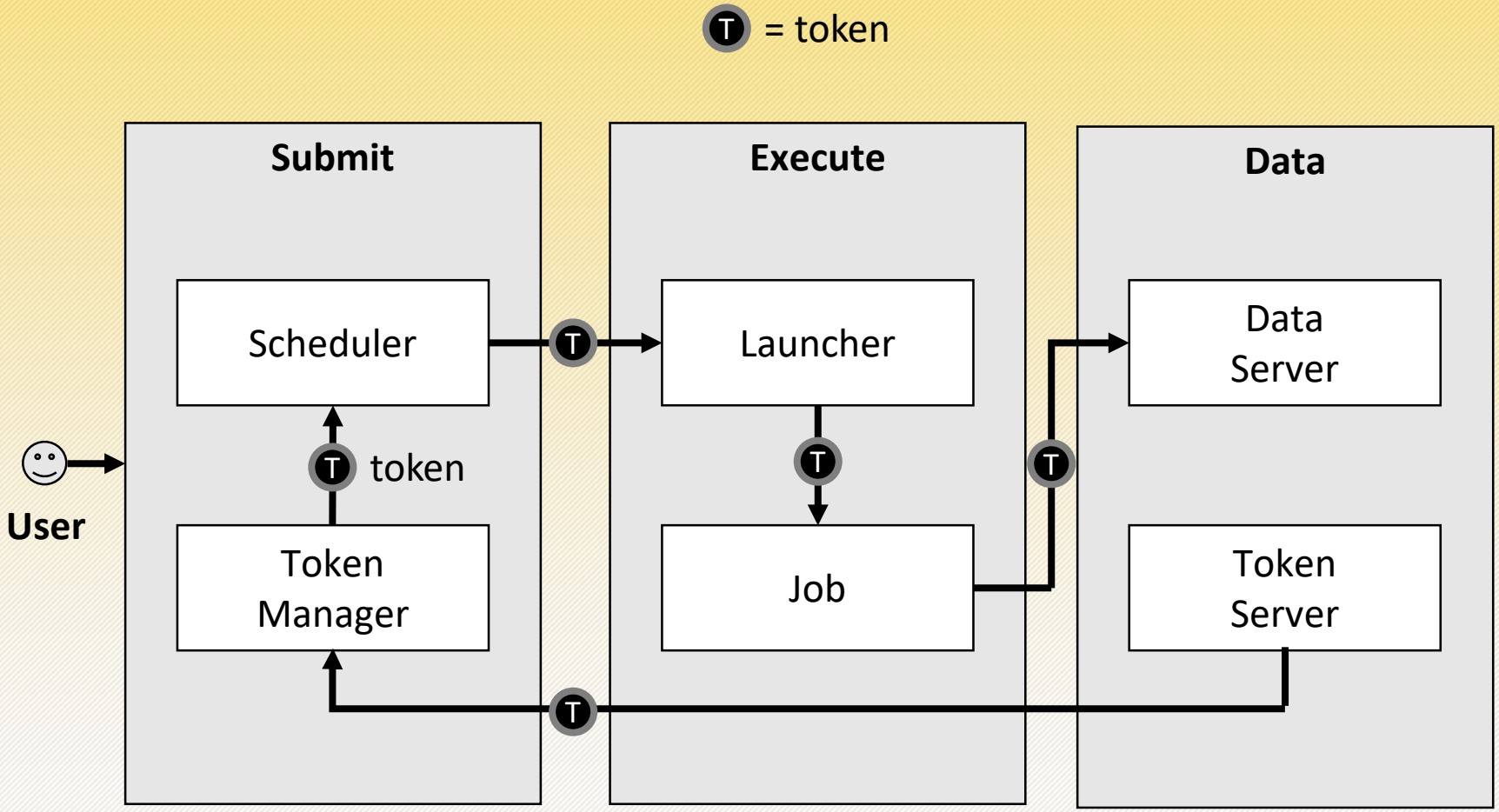
SCI TOKENS

- In OAuth2, there are three abstract entities involved in the authorization workflow:
  - **Authorization server** issues capabilities (tokens).
  - The **resource owner** (end-user) approves authorizations.
  - The **client** receives tokens. Often, this is the third-party website or smartphone app.
- Once the token is issued, it can be used at the **resource server** to access some protected resource.
  - In the Google example, Google runs both the authorization and resource servers.



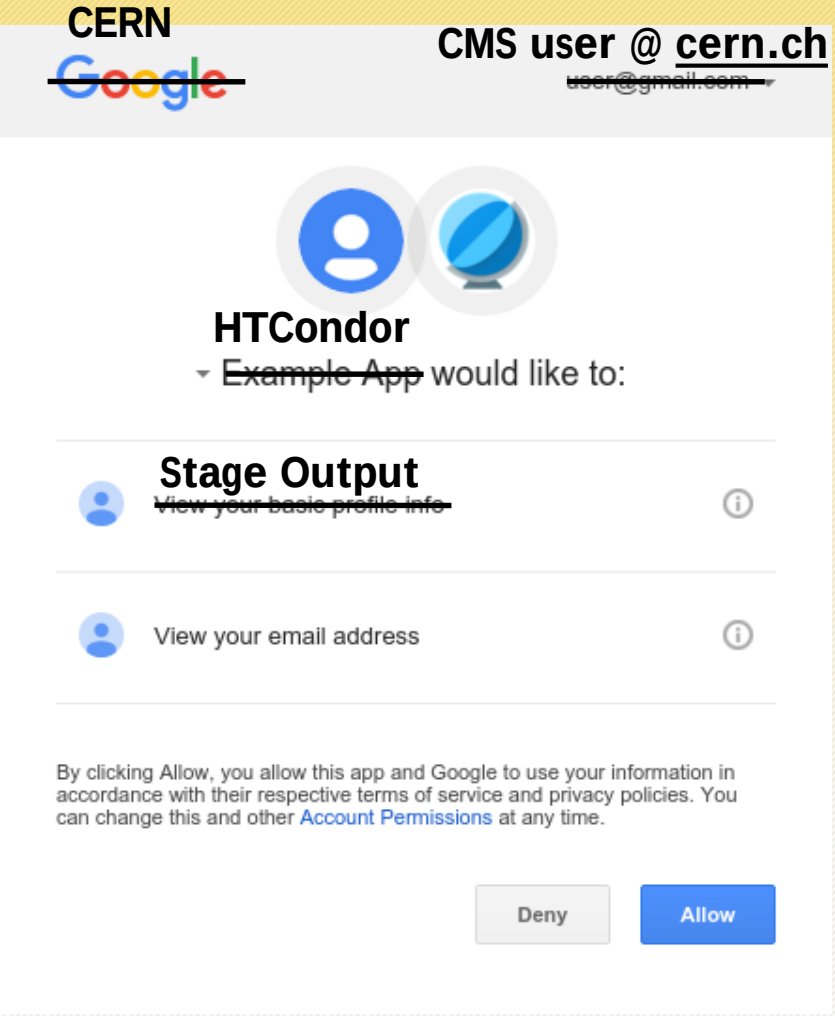
SciTokens Model

- Integrating an OAuth2 client on the HTCondor submit host
- Enhancing CILogon to support OAuth2 with VO-defined scopes
- Enhancing HTCondor to manage token refresh, attenuation, and delivery to jobs
- Enhancing data services (CVMFS, Xrootd) to allow read/writes using tokens instead of grid proxies



End-Goal

- The end-goal is this →
- The first time you use HTCondor, you navigate to a web interface and setup your desired permissions.
  - On every subsequent `condor_submit`, HTCondor will transparently create the access token for you. *User sees nothing*.
- Replace CERN, usernames, and authorization as desired.
  - **Goal**: our first use of OAuth2 will be to stageout from payload jobs to Box.



SCITOKENS

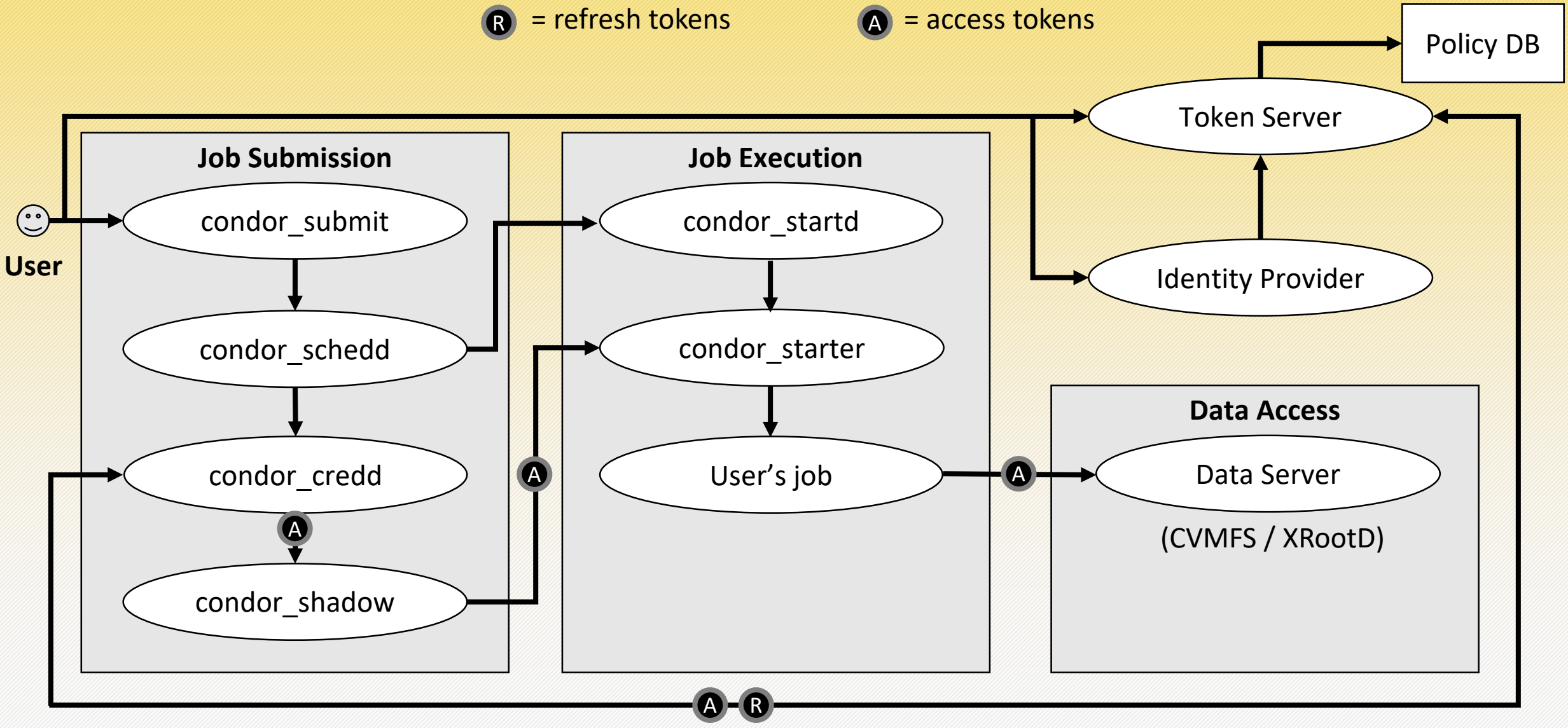
~~USER MANAGEMENT OF FILES~~

~~PASSWORD IN TERMINAL~~

~~SCITOKENS-PROXY-INIT~~

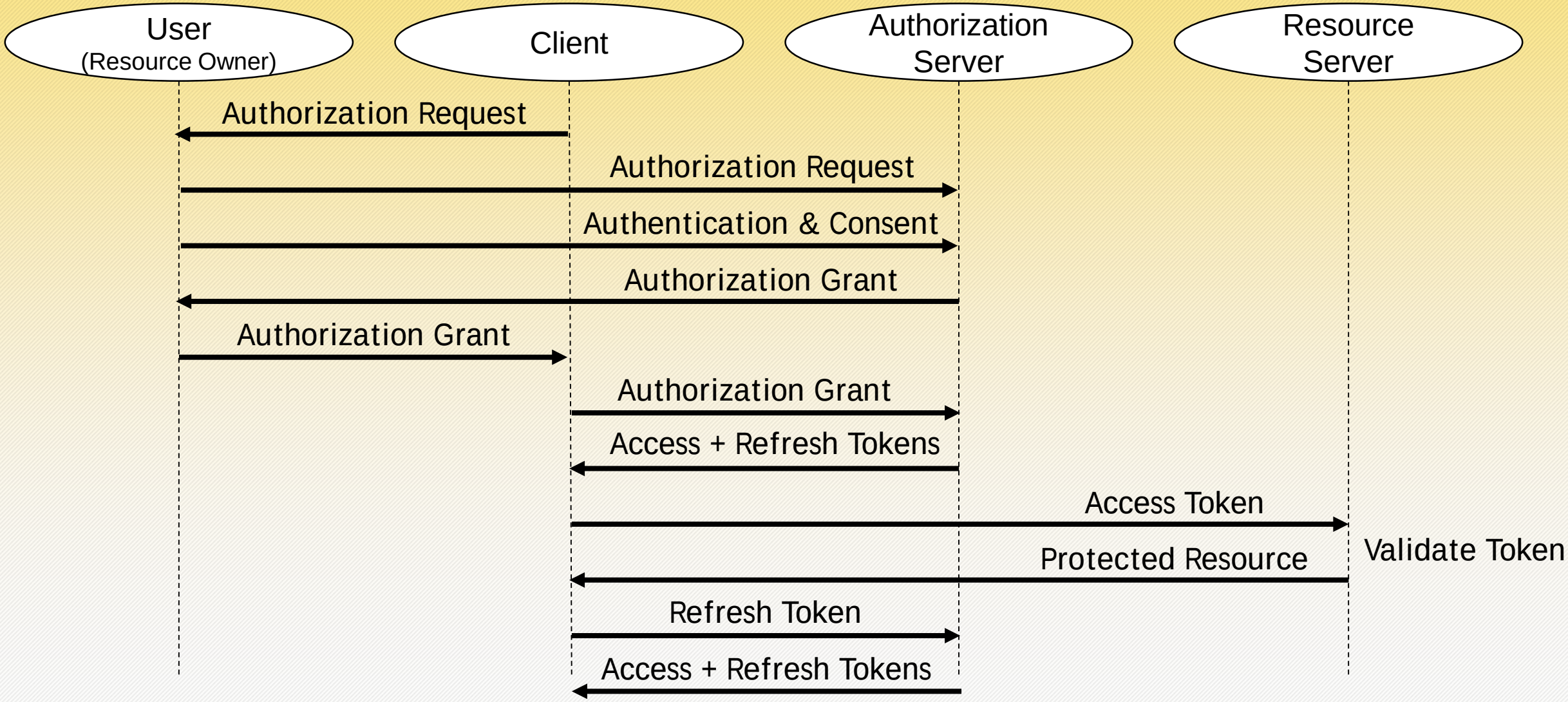
~~COPY / PASTE~~

Architecture



OAuth2 Authorization Framework

SCI TOKENS



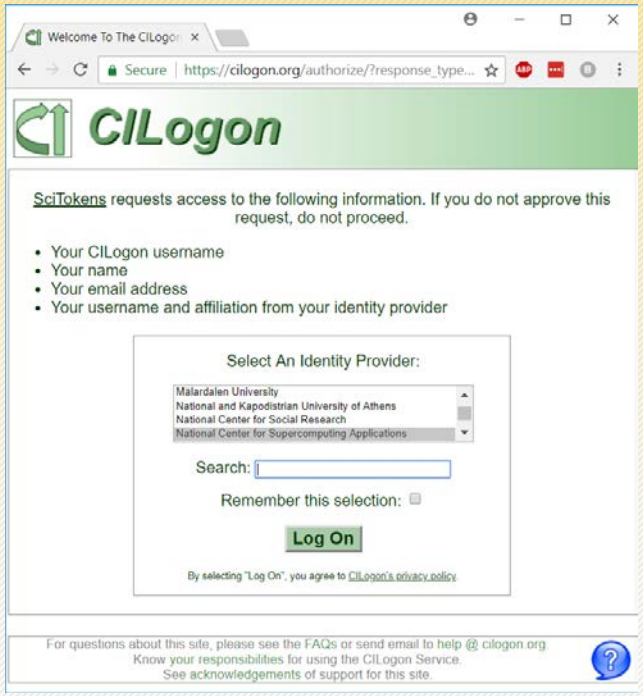
CILogon and SciTokens

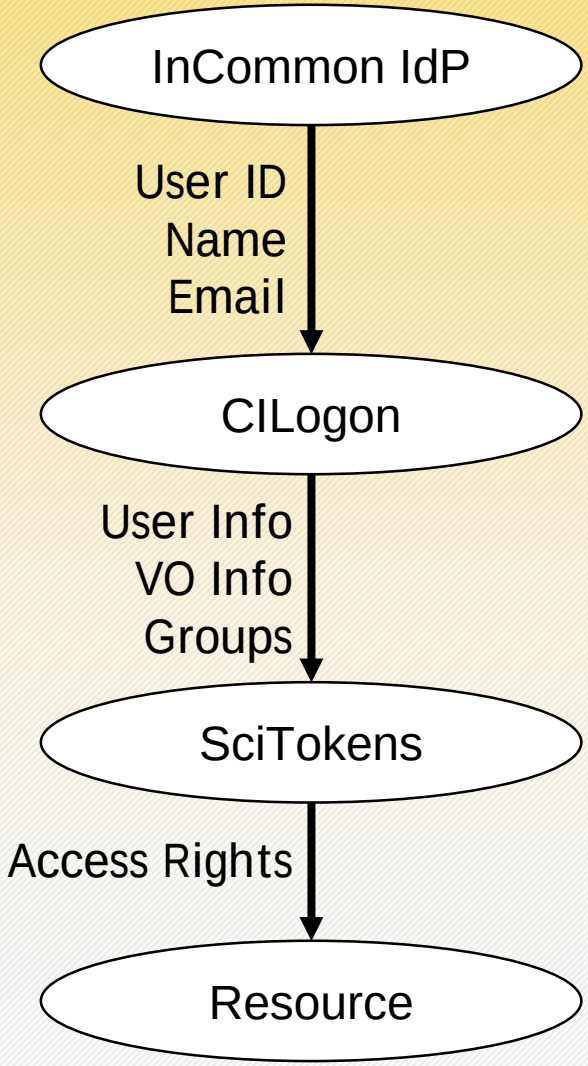
CILogon

- Federated Identity Management
- OpenID Connect
- ID Tokens

SciTokens

- Federated Authorization
- OAuth 2.0
- Access Tokens





Tokens for Distributed Science Infrastructures

- Distributed science infrastructures are distinct from a “resource server” like Google because they are not run by a single central entity.
- Hence, unlike Google, we can’t use opaque random strings for the token. We need something that allows for **distributed verification**.
  - Given a token, a storage service can determine it is valid.
  - Analogously, given a proxy chain and a set of trust roots, you can determine the GSI proxy is valid.
- Goal: Sites set aside some area for each VO; VOs manage the authorizations within these “VO home” areas.

JWT in action!

- Free tokens! Navigate to https://demo.scitokens.org to get your free tokens!
- This demo illustrates the access token format we're working on.
  - Utilizes JSON Web Tokens (JWT) as the access token format.
  - Various RFCs provide clear guidance on how to verify token integrity.
  - Adds a few domain-specific claims for receiving access to storage.
- The tokens are base64-encoded and can be used as part of a curl command to use protected resources.

Example Token, Decoded

SCI TOKENS

- The decoded token contains multiple scopes - basically filesystem authorizations.
- The audience narrows who the token is intended for.
- The issuer identifies who created the token; value used to locate the public keys needed to validate signature.
- The subject is an opaque identifier for the resource owner. In this case, it also happens to be the identity.
- The expiration is a Unix timestamp when the token expires. A typical lifetime is 10 minutes.

HEADER: ALGORITHM & TOKEN TYPE

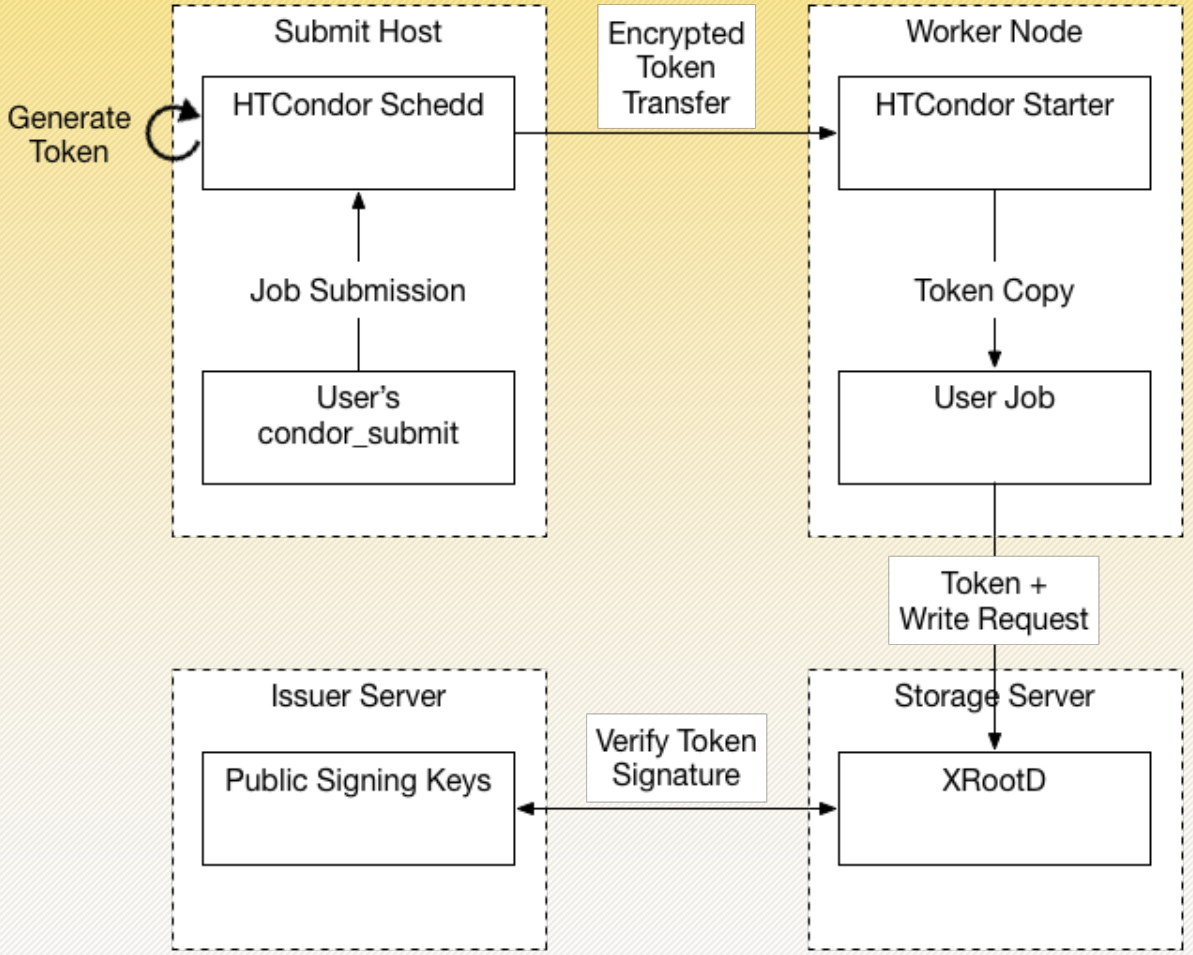

```
{
  "typ": "JWT",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "scope": "read:/protected write:/store/u25321",
  "aud": "https://demo.scitokens.org",
  "iss": "https://demo.scitokens.org",
  "sub": "bbockelm@cern.ch",
  "exp": 1526954997,
  "iat": 1526954397,
  "nbf": 1526954397,
  "jti": "78c44ce9-62bb-43e8-a7a6-f035f7ebd42b"
}
```

Early results on OSG

- We have been able to get a basic end-to-end token-based auth{z,n} workflow working for the OSG VO submit service.
- *This includes* patches to Xrootd to validate tokens presented via HTTP and to write files out with the correct Unix user permissions.
- **Cheats**:
  - instead of using OAuth2 to generate the token, we keep a signing key on the submit host.
  - only one token needed.
  - submit host and storage server owned by OSG.



Wait, I've seen this before!

- If you're from ALICE and getting a sense of déjà vu — you're right!
  - The capability-based infrastructure is precisely the authorization infrastructure used by ALICE for the **past decade**.
  - SciTokens takes this **successful model**, recasts it using modern web protocols, and utilizes OAuth2 workflows to issue the tokens.
- The use of common protocols and workflows means that we have a large number of battle-tested libraries we can leverage (spend our time doing other stuff besides writing the basics!).
- Using JWT-formatted access tokens is somewhat-commonplace among web companies.
  - We *think* SciTokens is unique in using JWT access tokens for distributed verification in a federated infrastructure.

Status & Next Steps

- So far we have:
  - Version 1.0 of Python and Java libraries
  - Simple HTCondor OAuth client implementation
  - XRootD token validation plugins
  - Token-based CVMFS access
  - X509-to-SciToken translation service
  - 3rd-party HTTPS FTS transfers authorized with SciTokens
- Next steps:
  - Use Java library for a dCache authorization plugin
  - Release plugin for CVMFS support
  - More fine-grained token management in HTCondor
  - Integration with LIGO LDAP
  - Enhancing HTCondor token support with OAuth flows

Thanks!

Visit
https://scitokens.org/
for more info.

Any questions?