SciAuth: A Lightweight End-to-End Capability-Based Authorization Environment for Scientific Computing

BRIAN AYDEMIR, Morgridge Institute for Research, USA
JIM BASNEY, National Center for Supercomputing Applications, University of Illinois, USA
BRIAN BOCKELMAN, Morgridge Institute for Research, USA
JEFF GAYNOR, National Center for Supercomputing Applications, University of Illinois, USA
DEREK WEITZEL, University of Nebraska-Lincoln, USA

We introduce a new end-to-end software environment that enables experimentation with using SciTokens for capability-based authorization in scientific computing. This set of interconnected Docker containers enables science projects to gain experience with the SciTokens model prior to adoption. It is a product of our SciAuth project, which supports the adoption of the SciTokens model through community engagement, support for coordinated adoption of community standards, assistance with software integration, security analysis and threat modeling, training, and workforce development.

CCS Concepts: • **Security and privacy → Authorization**.

Additional Key Words and Phrases: OAuth, JWT, distributed computing

ACM Reference Format:
Brian Aydemir, Jim Basney, Brian Bockelman, Jeff Gaynor, and Derek Weitzel. 2022. SciAuth: A Lightweight End-to-End Capability-Based Authorization Environment for Scientific Computing. In *Practice and Experience in Advanced Research Computing (PEARC '22), July 10–14, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3491418.3535160

1 INTRODUCTION

SciTokens [14] is a capability-based authorization system for distributed scientific computing, using JSON Web Token (JWT) [8] and OAuth [7] standards. While SciTokens has been incorporated into many software components and used in production in Open Science Grid, it still remains difficult for science projects to perform contained end-to-end experiments to better understand the SciTokens approach prior to adoption. In this article, we present a new end-to-end experimental environment, using popular software components (JupyterHub, HTCondor, GitHub), configured to use SciTokens and packaged in Docker containers to enable community experiments to further SciTokens adoption. We have developed this environment as part of the SciAuth project, which aims to help the scientific community realize the benefits of an interoperable, capability-based ecosystem. The software is publicly available at https://github.com/sciauth/sciauth-lightweight-environment.

2 RELATED WORK

The JWT Profile for OAuth 2.0 Access Tokens [4] provides a standard for an approach to distributed authorization that is being adopted across scientific computing environments and across the Internet. Using JWTs to convey authorization enables interoperability and extensibility through a self-describing token format that can be produced and consumed by a breadth of programming languages and application frameworks. The OAuth 2.0 framework provides interoperable mechanisms for token issuance [7], use [9], validation [11], and exchange [10].

PEARC '22, July 10–14, 2022, Boston, MA, USA

The WLCG Common JWT Profiles [1] extend SciTokens prior work to include authorization for compute resources in addition to data, along with operational guidelines that further enable interoperability and operational security in distributed scientific computing environments. Recent releases of SciTokens libraries add support for the WLCG profiles to enable broader adoption.

GA4GH Passports [13] provide a JWT profile for authorization in genomics and health research infrastructures, including ELIXIR Europe, the National Institutes of Health, and the Autism Sharing Initiative. GA4GH and its partners provide demonstration implementations, but do not provide an end-to-end experimental environment like SciAuth.

3 END-TO-END ENVIRONMENT

The goal of the SciAuth end-to-end environment is to provide an easy to use experimental environment for the SciTokens technology. The environment consists of an integrated set of Docker containers, created from existing software components and configured to produce and consume JWTs to implement basic authorization policies for access to compute and data (e.g., HTCondor workflows and GitHub repositories) via a standard web browser interface built on JupyterHub. Users can log in to the environment using their GitHub, Google, ORCID, Microsoft, or university account via CILogon [3], then experiment with different access policies, security configurations, and system behaviors. Users can also experiment with integrating additional services into the end-to-end environment using the common SciTokens authorization mechanism.

3.1 Overall Architecture

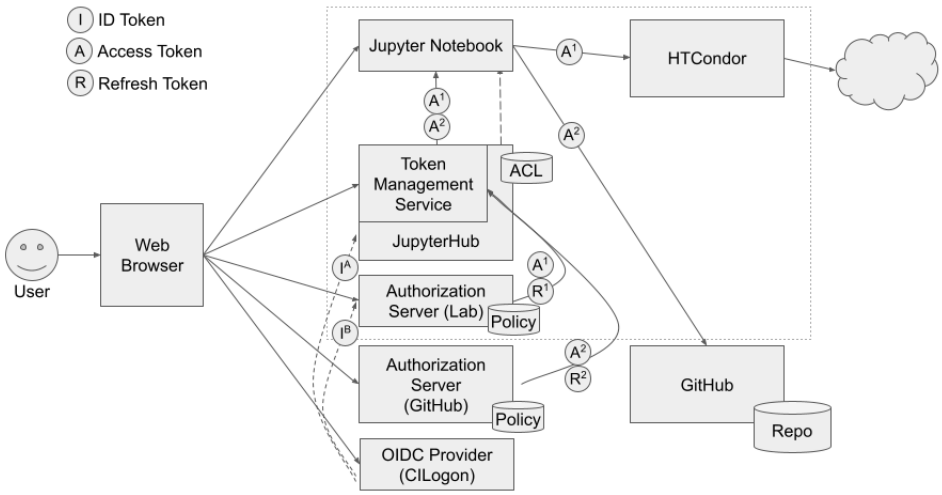

Fig. 1. The Architecture of the SciAuth End-to-End Environment

Figure 1 illustrates the overall system architecture of the SciAuth end-to-end environment. It includes an OpenID Connect Provider (CILogon), which issues ID Tokens for authentication, and two Authorization Servers (Lab and GitHub), which issue Access Tokens for authorization and Refresh Tokens for refreshing the Access Tokens. JupyterHub is configured to use the OpenID Connect Provider for authentication, with an access control list (ACL) that grants or denies access to the JupyterHub environment based on the ID Token contents (user identifier, attributes, and group memberships). JupyterHub is also configured with a Token Management Service to obtain tokens from the two Authorization Servers and to refresh those tokens as needed. When the user

launches a Jupyter Notebook, the Token Management Service delivers the Access Tokens into the Notebook environment, so the Notebook can submit jobs to HTCondor (using the Lab Access Token) and access GitHub repositories (using the GitHub Access Token). The Lab Authorization Server is also configured to use the OpenID Connect Provider for authentication, while the GitHub Authorization Server uses GitHub accounts for authentication.

The Docker containers provide the Lab Authorization Server, JupyterHub (with Token Management Service), example Jupyter Notebooks, and HTCondor services. These components are shown inside a box in Figure 1 to illustrate that they constitute the end-to-end environment. The environment connects to CILogon and GitHub as external services.

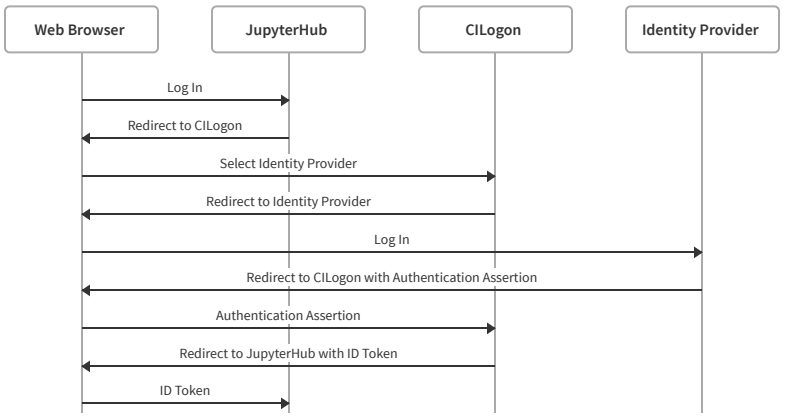

Fig. 2. Logging in to JupyterHub using an ID Token

Figures 2, 3, and 4 illustrate the sequence of operations for logging in to JupyterHub, obtaining Authorization Tokens for HTCondor and GitHub, and submitting HTCondor jobs and accessing GitHub repositories through the Jupyter Notebook.

3.2 Authorization Server

We developed a lightweight issuer [6] for the end-to-end environment, based on prior SciTokens work, to issue tokens using OAuth. The issuer is packaged as a Docker container. The user provides configuration to describe how it is run and the policy it should enforce.

3.2.1 Configuration. The authorization server requires several configuration parameters.

- Host certificate to enable HTTPs connections
- Private keys to sign issued SciTokens
- Client id and Client secret to authenticate with CILogon service
- Public hostname of the issuer that clients can use to request tokens
- Issuer policy file (described below in 3.2.2).

The configuration is given to the issuer through either environment variables or mounted into the container as a file or directory.

3.2.2 Authorization Server Policy. Policy in the authorization server is configured through a JSON file mounted into the Docker container at runtime. The policy file determines each user's scopes in the generated SciTokens, keyed on the *CILogon User Identifier* of the users by default.

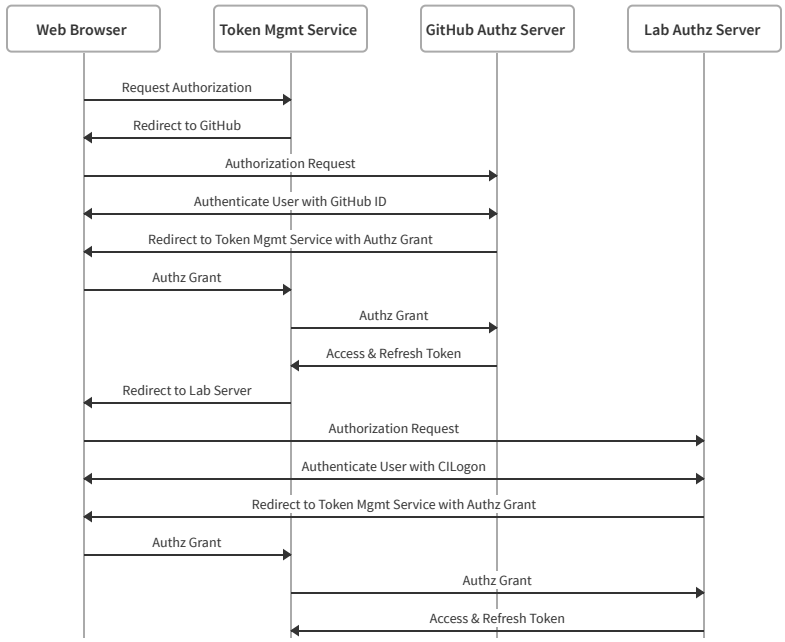

Fig. 3. Obtaining Authorization Tokens

Example Policy File

```
{
  "http://cilogon.org/serverA/users/12345": {
    "eduPersonEntitlement": [
      "compute.write compute.read compute.cancel compute.modify"
    ],
    "audience": "https://htcondor.localdomain/"
  },
}
```

The provided configuration issues SciTokens with permission to submit jobs to the local HTCondor container. Multiple users can be listed in the policy file, each with their own scopes and audience, or more advanced policies can be configured using boolean logic and regular expressions.

3.3 JupyterHub Token Management Service

The Token Management Service is a Python package [2] that can be installed alongside JupyterHub to enable it to act as an OAuth client for fetching and refreshing access tokens on a user's behalf. Users authorize the service to fetch tokens via a basic web UI, and they can retrieve tokens via either the web UI or from inside their Jupyter notebook via a web API.

Setting up the service consists of installing the Python package, configuring JupyterHub to start the service, and configuring the service with one or more OAuth clients for token issuers. The end-to-end environment builds-in the first two of these steps and provides a template file for the third, with placeholders for CILogon and GitHub registration parameters.

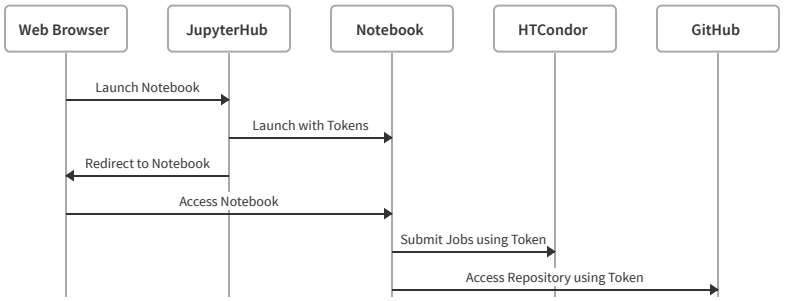

Fig. 4. Accessing the Jupyter Notebook with External Resources

3.4 Resource Providers

Since the JupyterHub Token Management Service uses the OAuth standard, it can integrate with a wide range of providers. The initial release of the SciAuth environment contains two integrations:

(1) **HTCondor Software Suite**. The HTCondor software suite provides a range of compute services. HTCondor's SchedD daemon exposes the ability to provide job management (submission, file transfer, removal) functionality over a remote network connection. HTCondor's underlying network protocol, CEDAR [12], includes a flexible negotiation of authentication method and has recently been extended to include authentication based on SciTokens. This provides the JupyterLab session with the ability to submit jobs to remote HTCondor systems as illustrated with token A_1 in Figure 1.

(2) **GitHub**. The OAuth model is used as an authorization mechanism across a wide range of commercial service providers, so while SciTokens has been integrated in several community services, the SciAuth environment is not limited to them. As illustrated for token A_2 of Figure 1, the token management service can acquire tokens from a service like GitHub, have it placed into the Jupyter environment, and enable the user to access a GitHub-based source repository from their notebook.

We plan to include additional integrations in in future releases. For example, the XRootD software framework is commonly use in the High Energy Physics community to move data between sites [5]. It provides support for multiple data transfer protocols, including the `xrootd` protocol and HTTP. XRootD now has an HTTP plugin for SciTokens, and the `xrootd` protocol has a new authentication mechanism (termed "ZTN") that adds support for tokens.

3.5 Deployment

Our end-to-end environment is implemented as a Docker Compose setup that is designed to be deployed locally on any host where the user can start and stop Docker containers. Deploying the environment consists of cloning the Git repository at https://github.com/sciauth/sciauth-lightweight-environment and following the provided instructions.

By default, our setup requires the user to be able to register an OIDC client with CILogon, for authenticating users to JupyterHub and the lightweight token issuer, and an OAuth application with GitHub, for providing GitHub Access Tokens. Aside from filling in configuration templates with these OAuth client IDs and secrets, other configuration is handled automatically.

4 CONCLUSIONS AND FUTURE WORK

In conclusion, the SciAuth end-to-end experimental environment addresses the need for hands-on access to SciTokens-enabled components prior to planning adoption across a scientific computing

environment. Since SciTokens is a technology for authorization in distributed systems, a true evaluation environment requires multiple interconnected services. The SciAuth packaging of service containers using Docker Compose eases deployment for experimenting with different authorization policies, security configurations, and performance modes. JupyterHub provides a standard web interface, avoiding the need for less friendly command-line interfaces.

Future work for the SciAuth project includes the development of training materials, using the end-to-end environment for hands-on exercises, for both in-person training events and online self-paced tutorials. For updates, please visit our project web site at https://sciauth.org/.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant numbers 1738962 and 2114989. The lightweight issuer is derived from prior work by Duncan Brown for the SciTokens project.

REFERENCES

[1] Mine Altunay, Brian Bockelman, Andrea Ceccanti, Linda Cornwall, Matt Crawford, David Crooks, Thomas Dack, David Dykstra, David Groep, Ioannis Igoumenos, Michel Jouvin, Oliver Keeble, David Kelsey, Mario Lassnig, Nicolas Liampotis, Maarten Litmaath, Andrew McNab, Paul Millar, Mischa Sallé, Hannah Short, Jeny Teheran, and Romain Wartel. 2019. *WLCG Common JWT Profiles.* Zenodo. https://doi.org/10.5281/zenodo.3460258

[2] Brian Aydemir. 2022. *scitokens-jupyter 0.0.3.* Zenodo. https://doi.org/10.5281/zenodo.6425179

[3] Jim Basney, Heather Flanagan, Terry Fleury, Jeff Gaynor, Scott Koranda, and Benn Oshrin. 2019. CILogon: Enabling Federated Identity and Access Management for Scientific Collaborations. *PoS* ISGC2019 (2019), 031. https://doi.org/10.22323/1.351.0031

[4] V. Bertocci. 2021. *JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens.* RFC 9068. IETF. https://doi.org/10.17487/RFC9068

[5] Brian Bockelman, Andrew Hanushevsky, Oliver Keeble, Mario Lassnig, Paul Millar, Derek Weitzel, and Wei Yang. 2019. Bootstrapping a new LHC data transfer ecosystem. In *EPJ Web of Conferences*, Vol. 214. EDP Sciences, EDP Sciences, France, 04045. https://doi.org/10.1051/epjconf/201921404045

[6] Duncan Brown, Derek Weitzel, and Jeff Gaynor. 2022. *scitokens/lightweight-issuer: First release.* Zenodo. https://doi.org/10.5281/zenodo.6418252

[7] D. Hardt. 2012. *The OAuth 2.0 Authorization Framework.* RFC 6749. IETF. https://doi.org/10.17487/RFC6749

[8] M. Jones, J. Bradley, and N. Sakimura. 2015. *JSON Web Token (JWT).* RFC 7519. IETF. https://doi.org/10.17487/RFC7519

[9] M. Jones and D. Hardt. 2012. *The OAuth 2.0 Authorization Framework: Bearer Token Usage.* RFC 6750. IETF. https://doi.org/10.17487/RFC6750

[10] M. Jones, A. Nadalin, B. Campbell, J. Bradley, and C. Mortimore. 2020. *OAuth 2.0 Token Exchange.* RFC 8693. IETF. https://doi.org/10.17487/RFC8693

[11] M. Jones, N. Sakimura, and J. Bradley. 2018. *OAuth 2.0 Authorization Server Metadata.* RFC 8414. IETF. https://doi.org/10.17487/RFC8414

[12] Zach Miller, Dan Bradley, Todd Tannenbaum, and Igor Sfiligoi. 2010. Flexible session management in a distributed environment. *Journal of Physics: Conference Series* 219, 4 (2010), 042017. https://doi.org/10.1088/1742-6596/219/4/042017

[13] Craig Voisin, Mikael Linden, Stephanie O.M. Dyke, Sarion R. Bowers, Pinar Alper, Maxmillian P. Barkley, David Bernick, Jianpeng Chao, Mélanie Courtot, Francis Jeanson, Melissa A. Konopko, Martin Kuba, Jonathan Lawson, Jaakko Leinonen, Stephanie Li, Vivian Ota Wang, Anthony A. Philippakis, Kathy Reinold, Gregory A. Rushton, J. Dylan Spalding, Juha Törnroos, Ilya Tulchinsky, Jaime M. Guidry Auvil, and Tommi H. Nyrönen. 2021. GA4GH Passport standard for digital identity and access permissions. *Cell Genomics* 1, 2 (2021), 100030. https://doi.org/10.1016/j.xgen.2021.100030

[14] Alex Withers, Brian Bockelman, Derek Weitzel, Duncan Brown, Jason Patton, Jeff Gaynor, Jim Basney, Todd Tannenbaum, You Alex Gao, and Zach Miller. 2019. SciTokens: Demonstrating Capability-Based Access to Remote Scientific Data using HTCondor. In *Proceedings of the Practice and Experience in Advanced Research Computing* (Chicago, IL, USA) *(PEARC '19)*. ACM, New York, NY, USA, Article 118, 4 pages. https://doi.org/10.1145/3332186.3333258