SciTokens: Demonstrating Capability-Based Access to Remote Scientific Data using HTCondor

Alex Withers
NCSA
alexw1@illinois.edu

Brian Bockelman
Morgridge Institute for Research
bbockelman@morgridge.org

Derek Weitzel
University of Nebraska-Lincoln
dweitzel@cse.unl.edu

Duncan Brown
Syracuse University
dabrown@syr.edu

Jason Patton
University of Wisconsin-Madison
jpatton@cs.wisc.edu

Jeff Gaynor
NCSA
gaynor@illinois.edu

Jim Basney
NCSA
jbasney@illinois.edu

Todd Tannenbaum
University of Wisconsin-Madison
tannenba@cs.wisc.edu

You Alex Gao
University of Illinois
yougao2@illinois.edu

Zach Miller
University of Wisconsin-Madison
zmiller@cs.wisc.edu

ABSTRACT

The management of security credentials (e.g., passwords, secret keys) for computational science workflows is a burden for scientists and information security officers. Problems with credentials (e.g., expiration, privilege mismatch) cause workflows to fail to fetch needed input data or store valuable scientific results, distracting scientists from their research by requiring them to diagnose the problems, re-run their computations, and wait longer for their results. SciTokens introduces a capabilities-based authorization infrastructure for distributed scientific computing, to help scientists manage their security credentials more reliably and securely. SciTokens uses IETF-standard OAuth JSON Web Tokens for capability-based secure access to remote scientific data. These access tokens convey the specific authorizations needed by the workflows, rather than general-purpose authentication impersonation credentials, to address the risks of scientific workflows running on distributed infrastructure including NSF resources (e.g., LIGO Data Grid, Open Science Grid, XSEDE) and public clouds (e.g., Amazon Web Services, Google Cloud, Microsoft Azure). By improving the interoperability and security of scientific workflows, SciTokens 1) enables use of distributed computing for scientific domains that require greater data protection and 2) enables use of more widely distributed computing resources by reducing the risk of credential abuse on remote systems.

In this extended abstract, we present the results over the past year of our open source implementation of the SciTokens model and its deployment in the Open Science Grid, including new OAuth support added in the HTCondor 8.8 release series.

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07.
https://doi.org/10.1145/3332186.3333258

CCS CONCEPTS

• **Security and privacy → Authorization**;

KEYWORDS

OAuth, capabilities, distributed computing

ACM Reference Format:
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 *Practice and Experience in Advanced Research Computing (PEARC '19), July 28-August 1, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3332186.3333258

1 INTRODUCTION

We introduced the SciTokens model last year in [12]. In this extended abstract, we present the implementation and deployment experience we have gained since then, including new OAuth support added in the HTCondor 8.8 release series.

SciTokens applies the well-known principle of capability-based security to remote data access. Rather than sending unconstrained identity tokens with compute jobs, we send capability-based access tokens. These access tokens grant the specific data access rights needed by the jobs, limiting exposure to abuse. These tokens comply with the IETF OAuth standard [5], enabling interoperability with the many public cloud storage and computing services that have adopted this standard. By improving the interoperability and security of scientific workflows, we 1) enable use of distributed computing for scientific domains that require greater data protection and 2) enable use of more widely distributed computing resources by reducing the risk of credential abuse on remote systems.

As illustrated in Figure 1 from [12], our SciTokens model applies capability-based security to three common domains in the computational science environment: Submit (where the researcher submits and manages scientific workflows), Execute (where the computational jobs run), and Data (where remote read/write access to

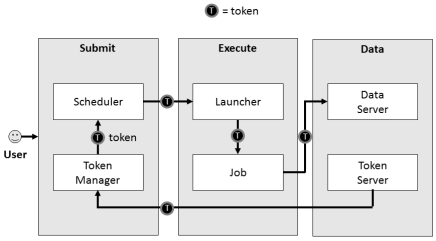

Figure 1: The SciTokens Model

Figure 2: Different Token Types

scientific data is provided). The Submit domain obtains the needed access tokens for the researcher's jobs and forwards the tokens to the jobs when they run, so the jobs can perform the needed remote data access. The Scheduler and Token Manager work together in the Submit domain to ensure that running jobs have the tokens they need (e.g., by refreshing tokens when they expire) and handle any errors (e.g., by putting jobs on hold until needed access tokens are acquired). The Data domain contains Token Servers that issue access tokens for access to Data Servers. Thus, there is a strong policy and trust relationship between Token Servers and Data Servers. In the Execute domain, the job Launcher delivers access tokens to the job's environment, enabling it to access remote data.

The SciTokens model adopts token types from OAuth (see Figure 2). Users authenticate with identity tokens to submit jobs (workflows), but identity tokens do not travel along with the jobs. Instead, at job submission time the Token Manager obtains OAuth refresh tokens with needed data access privileges from Token Servers. The Token Manager securely stores these relatively long-lived refresh tokens locally, then uses them to obtain short-lived access tokens from the Token Server when needed (e.g., when jobs start or when access tokens for running jobs near expiration). The Scheduler then sends the short-lived access tokens to the jobs, which the jobs use to access remote data.

The remainder of our extended abstract is organized as follows. In Section 2, we provide an overview of HTCondor's support for SciTokens. In Section 3, we describe recent HTCondor updates related to SciTokens. In Section 4, we describe our Open Science Grid deployment experience. In Section 5, we describe our new C/C++ implementation. In Section 6, we describe our OAuth server updates. Lastly, in Section 7 we describe our interoperability efforts with related work, and we conclude in Section 8.

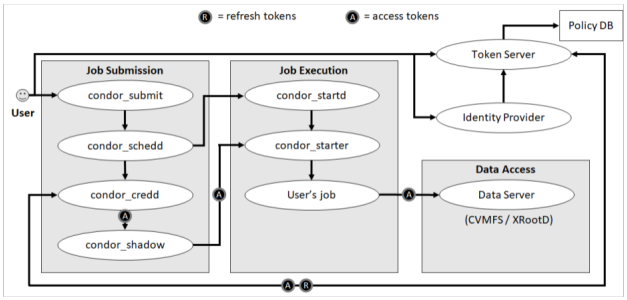

Figure 3: The SciTokens System Architecture

2 HTCONDOR IMPLEMENTATION

As the component that actually executes a scientific workflow, HTCondor serves as the linchpin that ties together all the SciTokens components. To best communicate our HTCondor approach, we first present a walk-thru of how HTCondor orchestrates the component interactions upon submission of a job, followed by a discussion of integration points.

As illustrated in Figure 3 from [12], the process begins when the researcher submits the computational job using the condor_submit command (or more likely using Pegasus or similar workflow frontend that then runs condor_submit). As part of the submission, the researcher specifies required scientific input data and locations for output data storage in the condor_submit input file. For example, in a LIGO PyCBC [10] submission, the researcher will specify a set of data "frames" from the LIGO instrument that are the subject of the analysis. Then condor_submit authenticates the researcher to the token_server(s) to obtain the tokens needed for the job's data access; as an optimization, condor_submit may first check for any locally cached tokens from the researcher's prior job submissions. The token_server determines if the researcher is authorized for the requested data access, based on the researcher's identity and/or group memberships or other researcher attributes. If the authorization check succeeds, the Token Server issues an OAuth refresh token back to condor_submit, which stores the refresh token securely in the condor_credd, and sends the job information to the condor_schedd. Since condor_submit gathers all the needed data access tokens, there is no need to store any identity credentials (e.g., passwords, X.509 certificates, etc.) with the job submission, thereby achieving our goal of a capability-based approach.

The next phase of the process begins when the condor_schedd has scheduled the job on a remote execution site. The condor_schedd communicates with the condor_startd to launch the job, establishing a secure communication channel between the condor_shadow on the submission side and the condor_starter on the remote execution side. The condor_starter then requests access tokens from the condor_shadow for the job's input data. The condor_shadow forwards the access token requests to the condor_credd, which uses its stored refresh tokens to obtain fresh access tokens from the token_server. The condor_credd returns the access tokens to the condor_shadow which forwards them securely to the condor_starter

which provides them to the researcher's job. Note that only access tokens are sent to the remote execution environment; the longer-lived refresh tokens remain secured in the submission environment which typically resides at the researcher's home institution. Lastly, the job uses the access tokens to mount CVMFS filesystem(s) to access scientific data. CVMFS verifies each access token to confirm that the token was issued by its trusted token_server and that the token's scope includes access to the scientific data being requested. If verification succeeds, CVMFS grants the requested data access. If the access token needs to be refreshed, the condor_starter makes another request back to the condor_shadow.

Note that SciTokens can leverage additional aspects of HTCondor, such as the fact that the condor_shadow can be made explicitly aware if the job is staging input data, accessing data online while the job is running, or staging output data. We allow the job submission to state three different sets of access tokens, which will only be instantiated at file stage-in, execution, and file stage-out, respectively. This enables long running jobs, for instance, to fetch a very short-lived write token for output that will only be instantiated once processing has completed. We also adjust the granularity of access token restrictions; for instance, the condor_shadow may request fresh access tokens for each job instance, allowing the token to be restricted in origin to a specific execution node. Alternatively, for greater scalability, access tokens can be cached at the credd and shared across all condor_shadow processes serving jobs that need the same data sets. Finally, we are investigating scenarios in which the data service and its accompanying token service is not fixed infrastructure, but instead is dynamically deployed upon execute nodes, perhaps by the workflow itself. In this scenario, the token service could be instantiated with a set of recognized refresh tokens a priori.

3 HTCONDOR UPDATES

In the past year our OAuth-enabled CredMon progressed from a research prototype to a supported component of HTCondor 8.8 [9]. The CredMon is a plug-in to the condor_credd that implements support for OAuth credentials. Since we use the OAuth standard, it was relatively straightforward to support both SciTokens and Box.com credentials in the same CredMon, so HTCondor jobs can access files both in CVMFS (using SciTokens) and in Box.com folders (using Box tokens).

OAuth support in HTCondor requires a lightweight web server on the submission node, to enable users to authorize credential issuance from the token server (e.g., SciTokens or Box.com) to the condor_credd. Packaging and configuring this web server component for successful deployment by HTCondor users was a source of multiple lessons learned over the past year. Of special note is the challenge of transferring credentials from the web application to the condor_credd, which are running under different service accounts for proper isolation. When the OAuth protocol delivers temporary credentials (the OAuth "code") to the web application, the web application writes the credentials to a roundevous directory that the condor_credd can read from to obtain longer-lived refresh tokens and access tokens. Thus, the more sensitive credentials are not exposed to the web application.

4 OSG DEPLOYMENT EXPERIENCE

Open Science Grid (OSG) has been an early adopter of the SciTokens model. Over the past year, 13 OSG users have used SciTokens credentials to secure almost two million StashCP uploads across over two thousand servers at 60 unique sites.

OSG currently uses SciTokens with HTCondor in "Local CredMon Mode". In contrast to HTCondor's "OAuth CredMon Mode", the "Local CredMon Mode" configuration uses a SciTokens credential issuer that's local to the HTCondor submit node, that issues credentials according to project-specific policies set by the submit node administrator. Since this mode does not use OAuth, the submitter does not see an OAuth consent screen, but instead HTCondor transparently adds the needed SciToken credentials to the user's job environment to enable project-specific StashCache access. This mode still relies on HTCondor's end-to-end credential management capabilities, for sending access tokens along with the jobs and refreshing tokens as needed.

This early OSG deployment experience has been especially helpful for working out SciTokens packaging and upgrade path details, since we have gone through upgrades of the SciTokens software (including file-server plug-ins) and credential profiles. Rolling out these updates across the distributed OSG infrastructure has required the SciTokens project to think about compatibility and versioning from the start.

5 C/C++ IMPLEMENTATION

In addition to our Java [4] and Python [11] SciTokens implementations, which we described in [12], we have added a C/C++ implementation [1], which enables improved performance for our CVMFS and XrootD integrations.

It has also enabled development of a SciTokens Apache module, which implements an Apache user authentication type. The module is developed using the SciTokens authorization helper for CVMFS and the JWT Apache authentication module. In the module, the token is retrieved from the current request with the Apache Portable Runtime (APR) library and verified with functions in the SciTokens C/C++ library.

The SciTokens C/C++ implementation has also enabled support for SciTokens as a native HTCondor authorization method (e.g., for authorizing access to the condor_schedd). SciTokens C++ RPMs will soon be included in the OSG software distribution.

6 OAUTH SERVER IMPROVEMENTS

We have updated our Token Server to support a more flexible policy language, including per-client policies based on SAML attributes and LDAP queries. We have also added support for OAuth token revocation [6], dynamic client registration [7, 8], and mobile clients [3].

7 RELATED WORK

The SciTokens project has benefited from participation in the WLCG Authorization working group, including involvement in the development of a WLCG profile for JSON Web Tokens [2] that is compatible with SciTokens.

The SciTokens project has also engaged in interoperability testing with other JWT implementations in the scientific community, including INFN IAM[1] and dCache[2] services.

8 CONCLUSIONS

The JSON Web Token and OAuth standards provide a solid foundation for distributed, capability-based authorization for scientific workflows. By enhancing existing components (CILogon, CVMFS, HTCondor, XrootD) to support the SciTokens model, we have provided a migration path from X.509 identity-based delegation to OAuth capability-based delegation for existing scientific infrastructures.

All SciTokens code is open source and published at https://github.com/scitokens. The HTCondor CredMon is also open source, published at https://github.com/htcondor/scitokens-credmon. Visit https://scitokens.org/ for the latest information about the SciTokens project.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1738962.

REFERENCES

[1] Brian Bockelman and Derek Weitzel. 2019. scitokens/scitokens-cpp (Version v0.3.0). (May 2019). https://doi.org/10.5281/zenodo.2656677
[2] A. Ceccanti, L. Cornwall, D. Crooks, B. Bockelman, D. Groep, D. Kelsey, N. Liampotis, M. Litmaath, M. Salle, H. Short, and R. Wartel. 2019. *WLCG Common JWT Profiles*. Technical Report.
[3] W. Denniss and J. Bradley. 2017. *OAuth 2.0 for Native Apps*. RFC 8252. http://www.rfc-editor.org/rfc/rfc8252.txt
[4] Jeff Gaynor. 2018. SciTokens for Java. (Nov. 2018). https://doi.org/10.5281/zenodo.1479838
[5] D. Hardt. 2012. *The OAuth 2.0 Authorization Framework*. RFC 6749. http://www.rfc-editor.org/rfc/rfc6749.txt
[6] T. Lodderstedt and M. Scurtescu. 2013. *OAuth 2.0 Token Revocation*. RFC 7009. http://www.rfc-editor.org/rfc/rfc7009.txt
[7] J. Richer, M. Jones, J. Bradley, and M. Machulak. 2015. *OAuth 2.0 Dynamic Client Registration Management Protocol*. RFC 7592. http://www.rfc-editor.org/rfc/rfc7592.txt
[8] J. Richer, M. Jones, J. Bradley, M. Machulak, and P. Hunt. 2015. *OAuth 2.0 Dynamic Client Registration Protocol*. RFC 7591. http://www.rfc-editor.org/rfc/rfc7591.txt
[9] HTCondor Team. 2019. HTCondor 8.8.2. (April 2019). https://doi.org/10.5281/zenodo.2636523
[10] Samantha A Usman, Alexander H Nitz, Ian W Harry, Christopher M Biwer, Duncan A Brown, Miriam Cabero, Collin D Capano, Tito Dal Canton, Thomas Dent, Stephen Fairhurst, Marcel S Kehl, Drew Keppel, Badri Krishnan, Amber Lenon, Andrew Lundgren, Alex B Nielsen, Larne P Pekowsky, Harald P Pfeiffer, Peter R Saulson, Matthew West, and Joshua L Willis. 2016. The PyCBC search for gravitational waves from compact binary coalescence. *Classical and Quantum Gravity* 33, 21 (2016), 215004. http://stacks.iop.org/0264-9381/33/i=21/a=215004
[11] Derek Weitzel and Brian Bockelman. 2018. scitokens/scitokens: v1.0.2. (March 2018). https://doi.org/10.5281/zenodo.1187173
[12] Alex Withers, Brian Bockelman, Derek Weitzel, Duncan Brown, Jeff Gaynor, Jim Basney, Todd Tannenbaum, and Zach Miller. 2018. SciTokens: Capability-Based Secure Access to Remote Scientific Data. In *Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18)*. ACM, New York, NY, USA, Article 24, 8 pages. https://doi.org/10.1145/3219104.3219135

[1] https://iam.infn.it/
[2] https://www.dcache.org/