Capability-Based Authorization for HEP

Derek Weitzel[1,*], *Brian* Bockelman[1,**], *Jim* Basney[2], *Todd* Tannenbaum[3], *Zach* Miller[3], and *Jeff* Gaynor[2]

[1]University of Nebraska - Lincoln
[2]NCSA
[3]University of Wisconsin-Madison

Abstract. Outside the HEP computing ecosystem, it is vanishingly rare to encounter user X509 certificate authentication (and proxy certificates are even more rare). The web never widely adopted the user certificate model, but increasingly sees the need for federated identity services and distributed authorization. For example, Dropbox, Google and Box instead use bearer tokens issued via the OAuth2 protocol to authorize actions on their services. Thus, the HEP ecosystem has the opportunity to reuse recent work in industry that now covers our needs. We present a token-based ecosystem for authorization tailored for use by CMS.

We base the tokens on the SciTokens profile for the standardized JSON Web Token (JWT) format. The token embeds a signed description of what capabilities the VO grants the bearer; the site-level service can verify the VO's signature without contacting a central service.

In this paper, we describe the modifications done to enable token-based authorization in various software packages used by CMS, including XRootD, CVMFS, and HTCondor. We describe the token-issuing workflows that would be used to get tokens to running jobs in order to authorize data access and file stageout, and explain the advantages for hosted web services. Finally, we outline what the transition would look like for an experiment like CMS.

1 Introduction

At the core of today's grid security infrastructure is the concept of *identity* and *impersonation* through grid certificates. A grid certificate provides you with a globally recognized identification. The grid proxy allows a third party to impersonate you on your behalf. The remote service maps your identity to a set of locally defined authorizations.

Securely accessing remote storage is a requirement for most HEP experiments. Accessing remote storage from a worker node is common in HEP workflows. The majority of access to remote storage is secured with an impersonation grid certificate. The worker nodes are distributed through the grid, and are not fully considered trusted. If the impersonation certificate is compromised, the attacker would have all of the authorizations that are available to the user.

[*]e-mail: dweitzel@cse.unl.edu
[**]e-mail: bbockelm@cse.unl.edu

A capability-based authorization is a restricted token that gives access to only a limited set of authorizations. This token can be restricted to only allow reading of certain paths from certain hosts. Rather than allowing an attacker access to all authorizations that are allowed for the user, the capability-based token only allows access to a very limited set of authorizations.

Capability-based authorization provides an opportunity to reuse technology industry standards and software for High Energy Physics. The industry has moved to capability tokens to access remote resources. Google, Box, Dropbox and many others use capability tokens to access storage and other resources.

2 Background

2.1 OAuth

OAuth is a workflow that allows for user to authorize a third party to perform actions on it's behalf. It is widely used in the internet industry. For example, an OAuth workflow could generate the authorizations to allow a website to store files into a users Dropbox account.

An OAuth workflow requires the user to authorize an agent a small set of authorizations to access a third party service.

3 Implementation

Various technologies had to be modified to utilize capability tokens. Within a workflow, capability tokens need to flow from the user to the the job.

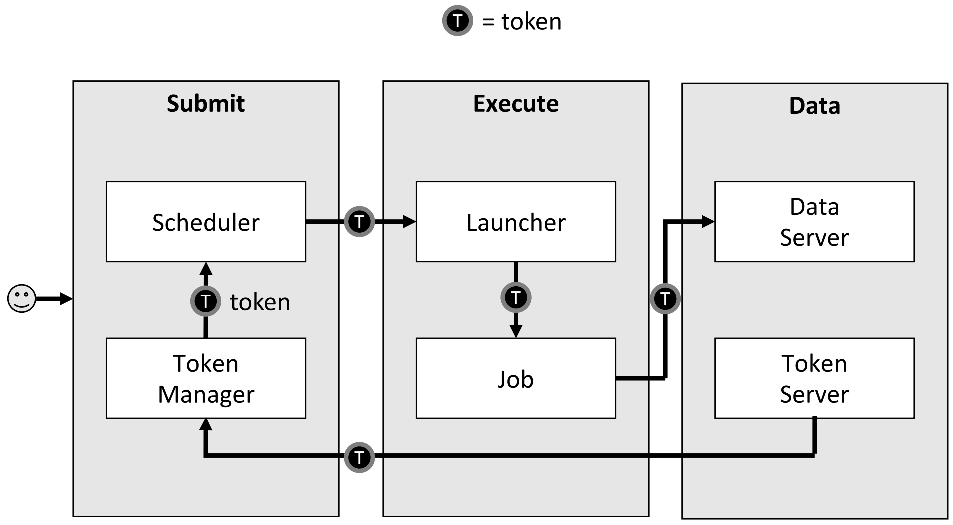

Figure 1. Token flow in a workflow

Figure 1 shows the flow of a token through a workflow. An access token and refresh token is first retrieved from a token server using an OAuth workflow. The token manager will manage the token on behalf of the user, using the refresh token to retrive access tokens when the current token expires. The scheduler will send the token to the execute host. The execute host will make the token available to the job, which will use it to access resources.

Each step of this workflow required modifications software that is used on the Grid. HTCondor [1] was enhanced in order to retrieve and refresh tokens on behalf of the user. XRootD [2] and CVMFS [3] where modified to accept and validate tokens to access data.

XRootD and CVMFS both use the SciTokens library [4], described in 3.1 to validate and authorize tokens.

3.1 SciTokens Library

SciTokens [5] provides a library that is used to create and validate scitokens. A SciToken is a JSON Web Token [6] with attributes that describe how a token can be used.

```
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "key-rs256"
}
{
  "scope": "read:/protected",
  "aud": "example.com",
  "iss": "https://demo.scitokens.org",
  "exp": 1539652082,
  "iat": 1539651482,
  "nbf": 1539651482,
  "jti": "76a3c07f-1e9d-4441-a3e2-6f37dbe26327"
}
```

Figure 2. Example SciToken

Figure 2 shows an example SciToken. The special attribute `scope` is used to describe how the token can be used. In this example, the token can be used to read from the resource at `/protected`.

The library can be used to validate the signature of the token. It can also be used to query the token for authorizations.

3.2 XRootD

XRootD is a high performance data server. We modified XRootD to pass an authorization header to a plugin. The client would use the authorization header to pass the SciToken to XRootD. We wrote a plugin [7] that would use the SciTokens library to validate the incoming token.

The plugin uses Boost [8] to connect the C++ API of XRootD to the Python API of the SciTokens library. XRootD passes the token to the plugin which creates an Access Control List of directories this token is allowed to access. XRootD will cache the ACL until the token expires.

Figure 3 shows an example HTTP request including the token. The token is used in the `Authorization` line, and it is treated as a bearer token [9]. The bearer token is encoded in base64 encoding [10]. XRootD will pass the `Authorization` header to the token plugin for validation and to create the ACLs.

```
GET /store/path HTTP/1.1
Host: storage.site1.com
Authorization: Bearer eyJ0eXAiOiJKV1...
```

Figure 3. Example HTTP Request with Token

3.3 CVMFS

Cern-VM FileSystem (CVMFS) is a read-only global data source. It is mounted on execute hosts as a traditional filesystem, making it easier to use than a remote storage system.

CVMFS has the capability to use external sources as the data repository. The metadata for files such as size, name, and checksums are stored within the CVMFS namespace. The contents of the files are stored on external servers reachable from CVMFS through an HTTP interface. When a user requests the contents of a file, CVMFS will request the data from a list of potential external HTTP servers.

In order to authenticate with the external servers, a token must be passed from the user to the data server. A token is stored within the user's environment and pointed to by an environment variable. When a request for data that is protected by tokens is requested by CVMFS, it will start a separate process that will retrieve the token from the user's environment and validate it. Once CVMFS has the token, it will use it to authenticate with the remote server and request the data.

3.4 HTCondor

HTCondor is a job and workflow management system. Support for Tokens, specifically OAuth 2.0 [11] tokens, where added to different components in the system, as seen in Figure 4.
Figure 4 shows the token flow within the HTCondor system.

1. The user first submits the job using the tool `condor_submit`.

2. The user is redirected to the token server and identity provider, which authorizes the users and creates a token for with the authorization requested.

3. The token and the refresh token is stored by the `condor_credd`.

4. The token is sent to the job through the `condor_shadow` and the `condor_shadow`.

5. The user's job can use the token to access data on the data server.

The `condor_credd` manages the token credential throughout the lifetime of the workflow. Periodically, it will use the refresh token to retrieve a new token from the token server. The token is only valid for a short time, while the refresh token could be valid for months. The `condor_credd` will refresh the token before the expiration and send the updated token to the jobs.

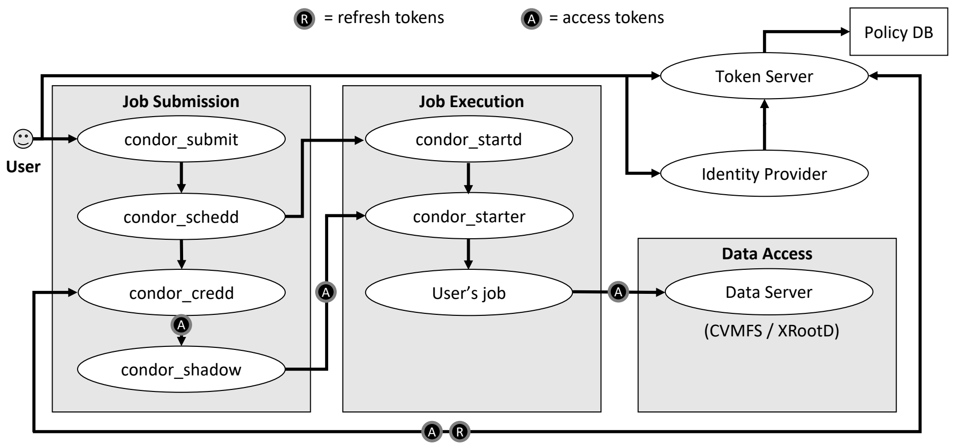

Figure 4. Token flow through HTCondor system

4 Evaluation

There are many advantages for webservers when using tokens rather than certificates for authentication.

When authenticating with a certificate, the authentication happens at the connection layer between the client and server. In most cloud services, the SSL connection is proxied and terminated at the boundary before the web server receives the request. A token is instead passed in the headers of the request which is sent through any proxies to the receiving web server.

5 Transition of CMS

A transition of a large organization like CMS to tokens will take many changes. I will break down the changes into workflows and user interfaces, data access during execution, and site to site data transfers.

The user interface for submitting jobs will add another step to acquire the transfer token. Currently, submission frameworks require the user to generate a grid proxy certificate. Instead of creating a grid proxy, the submission framework will redirect the users to a server that will follow the OAuth workflow to acquire an access token. The users will have to use a browser to acquire the token.

In contrast to the current submission frameworks, rather than trusting the host certificate on the submission node, the framework will need to trust the host itself. The OAuth workflow requires registration of each submission host in order to trust the client that the access token is issued.

The workflow will copy the access token to the execution host for use by the job. The token flow will follow Figure 1.

As previously mentioned, data servers map an identity to a set of locally defined authorizations. For capability based tokens, the authorizations will be in the tokens. Therefore the data servers must be configured consistently across the organization.

Data transfer servers must be modified to forward tokens for site to site transfers. Third party copy requires a token to authorize reading from one data server, and a token to write to another server. The data server must understand the third party copy command, as well as how to pass a token to another server. Additionally, the service coordinating the transfer must know how to acquire and refresh the tokens.

6 Conclusions

The capability tokens will change current processing and data services. Using capability tokens can increase security by distributing limited tokens, allowing processing to run in less trusted resources than is acceptable with current identity tokens.

References

[1] D. Thain, T. Tannenbaum, M. Livny, Concurrency - Practice and Experience **17**, 323 (2005)

[2] A. Dorigo, P. Elmer, F. Furano, A. Hanushevsky, WSEAS Transactions on Computers **1** (2005)

[3] P. Buncic, C.A. Sanchez, J. Blomer, L. Franco, A. Harutyunian, P. Mato, Y. Yao, *CernVM–a virtual software appliance for LHC applications*, in *Journal of Physics: Conference Series* (IOP Publishing, 2010), Vol. 219, p. 042003

[4] D. Weitzel, B. Bockelman, *scitokens/scitokens: Use scope attribute, updated from scp* (2018), `https://doi.org/10.5281/zenodo.1308902`

[5] A. Withers, B. Bockelman, D. Weitzel, D. Brown, J. Gaynor, J. Basney, T. Tannenbaum, Z. Miller, *SciTokens: Capability-Based Secure Access to Remote Scientific Data*, in *PEARC '18: Practice and Experience in Advanced Research Computing* (2018), `https://doi.org/10.1145/3219104`

[6] M. Jones, J. Bradley, N. Sakimura, Tech. rep., IETF (2015), `https://tools.ietf.org/html/rfc7519`

[7] B. Bockelman, D. Weitzel, *scitokens/xrootd-scitokens: Flexible Authorization Handling* (2018), `https://doi.org/10.5281/zenodo.1206218`

[8] B. Dawes, D. Abrahams, R. Rivera et al., URL http://www. boost. org **35**, 36 (2009)

[9] M. Jones, D. Hardt, Tech. rep., IETF (2012), `https://tools.ietf.org/html/rfc6750`

[10] S. Josefsson, Tech. rep., IETF (2006), `https://tools.ietf.org/html/rfc4648`

[11] D. Hardt, Tech. rep., IETF (2012), `https://www.rfc-editor.org/rfc/rfc6749.txt`