

Using Keycloak for Gateway Authentication and Authorization

Marcus A. Christie*, Indiana University; Anuj Bhandar, Intuit; Supun Nakandala, Indiana University; Suresh Marru, Indiana University; Eroma Abeysinghe, Indiana University; Sudhakar Pamidighantam, Indiana University; and Marlon E. Pierce, Indiana University

*Corresponding author address: Science Gateways Research Center, Pervasive Technology Institute, Indiana University, Bloomington, IN 47408; email: machrist@iu.edu

***Abstract:** Establishing users' identities before they access research infrastructure resources is a key feature of science gateways. With many science gateways now relying on general purpose gateway platform services, the challenges of managing identity-derived features have expanded to include authorization between science gateway tenants, middleware, and third party identity provider services. The latter include campus identity management systems. This paper examines the use of Keycloak as an implementation of an identity management system for Apache Airavata middleware, replacing our previous WSO2 Identity Server-based implementation. This effort raises larger issues that software-as-a-service communities should consider when embedding dependencies on third party software and services, including developing selection criteria and future-proofing systems.*

1. Introduction

Managing user identity is a critical feature for science gateways, which must provide secure and auditable access to restricted resources such as supercomputers, data sets, licensed scientific applications, and for-fee computing clouds. Science gateways must authenticate users, decide if they are authorized to access specific resources, manage expired accounts, and disable compromised accounts.

The basic approach is for a gateway to provide its own user management and authentication system that is an integral part of the gateway's implementation. A gateway that builds over a more general purpose framework such as Drupal or Joomla may use authentication add-ons for

managing users.

Gateway developers today have several additional options. First is the emergence of well-supported authentication services, such as the InCommon Federation that is used by many academic institutions. Facebook, Google, GitHub and other Web-based companies also provide free authentication services that can be integrated into online applications. OpenID Connect [1] has become a popular protocol for Web authentication; it builds over the OAuth 2 authorization protocol [2]. CILogon [3] provides a unifying authentication layer over these different providers. Thus, gateways may outsource user authentication to various services. The gateway may still choose to manage its users internally through a user store (such as an attached database or LDAP server), or it may outsource this as well; a campus-centered gateway may for example connect to a user account system (such as LDAP) managed by the campus cluster providers.

The second important trend has been the emergence of science gateway platform-as-a-service offerings. These are hosted services that can serve multiple gateway tenants simultaneously. Science gateway platforms provide general purpose services such as user management, data management, and job execution, while the gateway tenant provides user interfaces geared towards to a specific user community. Gateway tenants access the gateway platform middleware through secure, network accessible APIs. Various patterns for interactions between gateway tenants and gateway middleware are reviewed in [4], which can be mapped to OAuth 2 authorization grant flows.

This short paper considers how to implement basic interactions between a gateway tenant and Apache Airavata middleware [5]. Apache Airavata is open source software that the authors operate as part of the Science Gateways Platform as a service (SciGaP.org) project. We use Keycloak [6], an open source identity management system, to implement SciGaP’s Identity and Access Management system. Keycloak can authenticate users through a number of different mechanisms, replacing our previous approach based on WSO2 Identity Server (WSOS IS) [7].

2. Gateway Identity Management and Apache Airavata

Apache Airavata is an open source software framework that enables gateway providers to compose, manage, execute, and monitor large scale applications and workflows on distributed computing resources such as local clusters, supercomputers, computational grids, and computing clouds. Apache Airavata provides a programming language-independent API that can be used by one or more remote science gateways. The authors operate a hosted version Apache Airavata and other services (such as Keycloak) needed to run a production version of the system for client gateways.

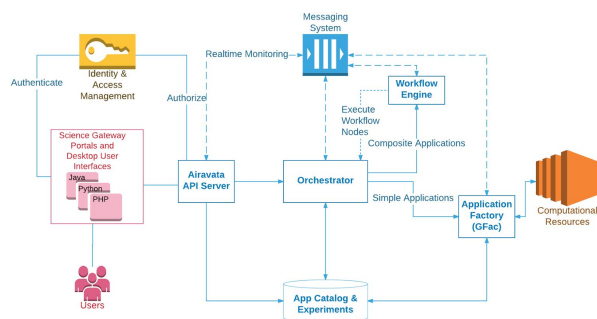


Figure 1: Apache Airavata’s conceptual framework.

Figure 1 provides a high level summary of Apache Airavata. Users authenticate to a gateway tenant, which in turn interacts with Apache Airavata’s API server. The API server routes requests to the appropriate internal components, which provide access to data and metadata, execute applications on remote resources, and move data

between resources.

The general interaction between Web-based gateways and Apache Airavata middleware is shown in Figure 2; see also [4]. When a user logs in, the gateway contacts the Authorization Server, which redirects to a federated authentication service such as CILogon. After a successful authentication, the Authorization Server returns an access token to the gateway. The gateway can use this access token to request services through the API server. The API server validates the access token and may make additional authorization decisions.

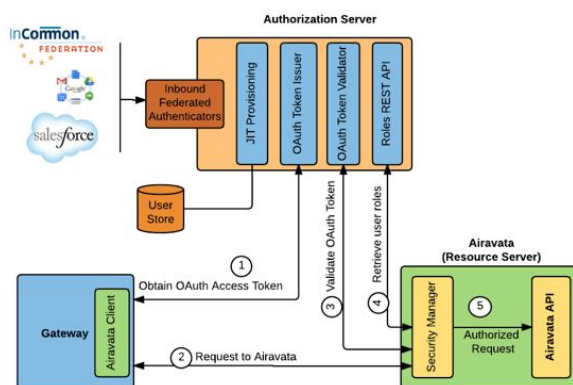


Figure 2: Gateway-middleware integration using OpenID Connect and OAuth2.

3. Implementation Details and Related Considerations

In previous work, we implemented Figure 2 using WSO2 IS as the Authorization Server. We were motivated however to find a replacement service when we encountered difficulties integrating with CILogon. In theory, WSO2 IS supports external identity providers, but in practice we encountered difficulties configuring the Public Key Infrastructure (PKI) trust store so that it would accept the signing Certificate Authority (CA) of CILogon’s SSL certificate.

Keycloak offers a feature set very similar to WSO2 IS. Keycloak provides a user store and administrative functions for administering users, including user roles. Keycloak also supports identity federation for identity providers that support OpenID Connect or SAML.

The CILogon service provides an OpenID Connect service including a self-service ability to

register clients and OpenID Connect endpoints for retrieving user information, etc. Once an administrator registers a web portal with CILogon, the client ID and client secret can be used to configure setup a CILogon identity provider in Keycloak.

Users who authenticate via CILogon are automatically provisioned in Keycloak's user store. During the signin process, Keycloak, once the user is authenticated, redirects back to the web portal with an authorization code. The web portal can then use the authorization code to get an access token and retrieve the user's profile from Keycloak. This user profile (first name, last name, email address) will match the details that Keycloak itself retrieves from CILogon.

Users are assigned to one or more roles to grant them access to different subsets of Apache Airavata API methods. Keycloak exposes a REST API that allows managing a user's roles. This functionality is exposed in the web portal to users with the "admin" role. Admins are able to manage the roles assigned to a user. Typically new users are assigned to a role which has no API access and a decision must be made by an admin user as to which role(s) to assign the user.

User authorization occurs in the Apache Airavata API server, which securely brokers requests to other Airavata services. When the web portal calls an API server method, it passes the user's access token. The API server first makes a call to Keycloak to verify the access token is valid. If the access token is valid a second call is made to Keycloak to get a list of roles that are assigned to the user. The API server has a list of API methods that are accessible to each role. The request is authorized if the user has a role that can access the given API method.

Another design goal we had was to use Keycloak as a backend service so that the user is never exposed to the Keycloak user interfaces. The reason for not exposing the user to the Keycloak user interface is simply to avoid needing to build user trust in this additional authentication service and thus avoid user confusion.

When a user is doing a standard username-password login, this is accomplished by using a Resource Owner Password Credentials

grant flow by which the web portal directly submits the username and password to Keycloak and gets a code that can be exchanged for an access token. When a user is logging in through CILogon, the web portal redirects to Keycloak with a special query parameter indicating to Keycloak which identity provider to redirect to for the user's authentication. Thus, the web portal redirects to Keycloak, which immediately redirects to CILogon, and the user never sees the Keycloak login page.

Based on our past experience with third party identity providers like WSO2 IS, a major design goal with the Keycloak integration was to create abstractions for all the functionalities needed by Airavata for managing users and their roles. Calls to the Keycloak REST API are made indirectly through defined interfaces, providing a layer of separation between the core Airavata code and the Keycloak API. This design will shield Airavata from creating provider-specific dependencies and enable easy replaceability. As a part of this initiative, we were successful in developing interfaces for Tenant Management, client configuration and user management. Each of these interfaces will also help automate the process of creating a new gateway without the need of an administrator to manually configure the new tenant identity provider's admin console.

4. Related Work

Science gateway security has been examined in [3][8]. These works are the basis of the current paper but do not consider the details of multi-tenanted science gateway middleware.

Our previous work [4] considered a larger number of gateway-middleware integration use cases. The basic case, depicted in Figure 2, assumes the user store is attached to the Authorization server. Other cases include user stores attached to the gateway and user stores provided by external services (such as an LDAP server maintained by a department). [4] also examined other OAuth2 grant flow cases such as those that occur when integrating desktop interfaces.

The Globus Auth [9] service is a

software-as-a-service system that implements many of the same capabilities as Keycloak and WSO2 IS. Globus Auth additionally provides support for groups and is integrated with other Globus services such as file transfer. Keycloak and WSOS IS are open source software that can be used by gateways and middleware operators to offer identity management services. It is possible to integrate these with Globus Auth as well by making Globus Auth a trusted identity provider. This would enable an Apache Airavata-based gateway to use Keycloak-based identity management services and Globus file transfer services.

5. Conclusions

This effort touches on two larger challenges faced by science gateways. First, gateways no longer need to implement all required capabilities themselves. We refer to this as the “build versus buy” decision, in which a gateway development team decides to use a third party piece of software or service rather than build everything they need from scratch. Identity management over the last decade has matured significantly, and there are a number of high quality solutions that a gateway can choose from.

Although the “buy” option (that is, use a third party software or service) has many advantages, it is inevitable that the gateway will need to replace a solution over time. In our case, WSO2 IS worked well for our initial use cases, and we put it successfully into production, but it failed when we had new requirements. Furthermore, even though WSO2 IS is open source software, we realized that the modifications needed were too much of a burden to implement and maintain, and we had concerns about WSO2’s open source governance model.

Replacing IS with Keycloak, even though the products have similar capabilities, was complicated by the use of WSO2 IS-specific code within our reference implementation gateway. Keycloak also does not support XACML, which we used in WSO2 IS to make access control decisions on API access. Instead the API server uses Keycloak’s REST API to apply a simple role based access

control as described above. This complication could have been ameliorated by providing more “wrapping” coding to isolate implementation details.

6. Acknowledgments

This work is supported by NSF Award ##1339774.

7. References

- [1] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B. and Mortimore, C., 2014. OpenID connect core 1.0. *The OpenID Foundation*, p.S3.
- [2] Hardt, D., 2012. The OAuth 2.0 authorization framework.
- [3] Basney, J., Fleury, T. and Gaynor, J., 2014. CILogon: A federated X. 509 certification authority for cyberinfrastructure logon. *Concurrency and Computation: Practice and Experience*, 26(13), pp.2225-2239.
- [4] Nakandala, S., Gunasinghe, H., Marru, S. and Pierce, M., 2016. Apache Airavata Security Manager: Authentication and Authorization Implementations for a Multi-Tenant eScience Framework.
- [5] Marru, S., Gunathilake, L., Herath, C., Tangchaisin, P., Pierce, M., Mattmann, C., Singh, R., Gunarathne, T., Chinthaka, E., Gardler, R. and Slominski, A., 2011, November. Apache airavata: a framework for distributed applications and computational workflows. In *Proceedings of the 2011 ACM workshop on Gateway computing environments* (pp. 21-28). ACM.
- [6] Keycloak: <http://www.keycloak.org/>
- [7] WSO2 Identity Server: <http://wso2.com/identity-and-access-management>
- [8] Welch, V., Barlow, J., Basney, J., Marcusi, D. and Wilkins-Diehr, N., 2007. A AAAA model to support science gateways with community accounts. *Concurrency and Computation: Practice and Experience*, 19(6), pp.893-904.
- [9] Tuecke, S., Ananthakrishnan, R., Chard, K., Lidman, M., McCollam, B. and Foster, I., 2016. Globus Auth: A research identity and access management platform. In *12th IEEE International Conference on e-Science*.