Cloud Bursting Galaxy: Federated Identity and Access Management

Vahid Jalili[*], Enis Afgan[†], James Taylor[‡], and Jeremy Goecks[§]

December 13, 2018

Abstract

Motivation: Large biomedical datasets, such as those from genomics and imaging, are increasingly being stored on commercial and institutional cloud computing platforms. This is because cloud-scale computing resources, from robust backup to high-speed data transfer to scalable compute and storage, are needed to make these large datasets usable. However, one challenge for large-scale biomedical data on the cloud is providing secure access, especially when datasets are distributed across platforms. While there are open Web protocols for secure authentication and authorization, these protocols are not in wide use in bioinformatics and are difficult to use for even technologically sophisticated users.
Results: We have developed a generic and extensible approach for securely accessing biomedical datasets distributed across cloud computing platforms. Our approach combines OpenID Connect and OAuth2, best-practice Web protocols for authentication and authorization, together with Galaxy (https://galaxyproject.org), a web-based computational workbench used by thousands of scientists across the world. With our enhanced version of Galaxy, users can access and analyze data distributed across multiple cloud computing providers without any special knowledge of access/authorization protocols. Our approach does not require users to share permanent credentials (e.g., username, password, API key), instead relying on automatically-generated temporary tokens that refresh as needed. Our approach is generalizable to most identity providers and cloud computing platforms. To the best of our knowledge, Galaxy is the only computational workbench where users can access biomedical datasets across multiple cloud computing platforms using best-practice Web security approaches and thereby minimize risks of unauthorized data access and credential use.
Availability and Implementation: Freely available for academic and commercial use under the open-source Academic Free License (https://opensource.org/licenses/AFL-3.0) from the following Github repositories: https://github.com/galaxyproject/galaxy and https://github.com/galaxyproject/cloudauthz
Contact: jalili@ohsu.edu, goecksj@ohsu.edu

1 Introduction

Genomics is expected to be an exabase-scale Big Data domain by 2025, posing greater data acquisition and storage challenges than astronomy, YouTube, and Twitter—the other major generators of Big Data [22]. Big data in genomics has led to substantial advances in several areas, including developmental biology, hu-

[*]Department of Biomedical Engineering, Oregon Health and Science University, OR, USA
[†]Department of Biology, Johns Hopkins University, Baltimore, MD, USA
[‡]Department of Biology, Johns Hopkins University, Baltimore, MD, USA
[§]Department of Biomedical Engineering, Oregon Health and Science University, OR, USA

man evolution, and precision medicine. However, genomics data has become widely distributed because DNA sequencers are easily available and inexpensive, making it simple for laboratories to generate their own data. Distributed genomics data poses increasing challenges for making effective use of the data, especially for the increasingly important tasks of data integration and joint analysis.

The emergence of cloud-based solutions such as Amazon Web Services (AWS) and Microsoft Azure have paved the path for online, scalable, cost-effective, secure, and shareable Big data persistence and analysis with a growing number of researchers and laboratories hosting (publicly and privately) their genomics Big data on cloud-based services [15]. Most computational analyses of genomics data requires complex workflows that include many steps and analysis tools. Computational workbenches such as Galaxy [1] and GenomeSpace [19] make it simple to create and execute analysis workflows. Consequently, these systems need to seamlessly access cloud-based genomics data to execute workflows.A prerequisite is to mutually satisfy principals (users and services) about each other's identity and privileges. Given that genomics data, especially those coming from clinical samples, are often highly sensitive data, as they may contain protected health information (PHI) or personally identifiable information (PII) [8], protocols are needed to implement secure authentication and authorization.

The few systems that allow leveraging cloud-based storages for workflow execution (e.g., GenomeSpace) and data transfer (e.g., Globus [13]), rely on locally storing user credentials for a given cloud service provider. For instance, to transfer data to/from a cloud service provider, Globus prompts for user credentials with the provider which are then cached and used to access the resources. Delegating user's private credentials to a third-party service is a privacy and security risk for the user and a liability risk for the application service provider. Additionally, revoking these access credentials requires manual intervention, and initially obtaining credentials requires users to be familiar with the technical details of the given cloud provider, hindering adoption. These challenges highlight the need for a robust solution to securely delegate access to cloud-based resources.

We have devised an approach for federating identity and access management and implemented it in the Galaxy project. The approach enables a secure and seamless delegation of privileges without sharing principal's login and/or access credentials. This approach makes it possible for Galaxy users to securely access genomics data across a wide variety of cloud computing platforms. The advantages of our approach are twofold: first, leveraging OpenID Connect (OIDC), it outsources user authentication to trusted identity providers—a user can login to Galaxy using an existing third-party identity (e.g., a Google account) without having to explicitly create a Galaxy user account. This feature paves the path for the Galaxy-as-service model [2] where a user can login to all Galaxy servers using a single identity. In addition to simplifying the login process, this model protects users' identity and credentials should any Galaxy server suffer a security breach.

The second advantage of our approach is that a user can securely grant Galaxy authorization to access genomic data on the cloud. Previously, users needed to share their permanent cloud credentials with Galaxy [3], which is problematic because those credentials grant Galaxy the same privileges as the user, Galaxy must store and secure those credentials, and users must manually obtain those credentials. We have developed an approach that leverages *CloudAuthz* (`https://github.com/galaxyproject/cloudauthz`) for on-demand and automatic generation of temporary access credentials to assume minimum delegated privileges.

Our approach is implemented in the Galaxy framework, making it possible for Galaxy users to access and analyze cloud-based genomics data in their workspaces. Galaxy users can access cloud data that they own or have access to by specifying a provider name (e.g., AWS) and a resource name (e.g., an Amazon Simple Storage Service (S3) bucket), but share neither login nor access credentials (see section 2). Being able to perform these steps without requiring the user to supply their permanent access credentials has the following advantages:

- Galaxy need not use, store, or protect user access credentials, and users can authenticate using available Web identity providers such as Google. Thus our approach is both user-friendly and secure;

- User identity is exposed to Galaxy via security tokens that cannot be exploited to impersonate them. This point is guaranteed by contrivances such as *audience* claim (claim is "piece of information asserted about an entity" [20]) in the ID token of OIDC protocol [20];

- Authorization to data follows the principle of least privilege, so Galaxy only has access to given datasets and not users' account information;

- Delegated privileges (exposed using short-term authentication/authorization tokens) issued for and assumed by Galaxy, are independent from a user's credentials, hence their scope can be restricted independently;

- Can leverage the role-based access control model [21], which enables segregating duties and provide a principal with the least privileges required to perform its authorized action;

- The short-term authentication/authorization tokens issued for Galaxy are refreshed automatically and can be revoked by the resource owner, either from the IdP or the resource provider;

- The privileges issued for a principal—a Galaxy server in this case—cannot be assumed by a different client (e.g., another Galaxy server, or a different web app). This is guaranteed by OIDC protocol.

1.1 Motivating Application

Cloud-based services have become a ubiquitous storage solutions due to their scalability, availability, and cost efficiency, which makes them an ideal solution for genomics Big data storage that are commonly studied in a collaborative setting. Accordingly, a growing number of datasets are publicly hosted on cloud service providers. For instance, *Tabula Muris* is a single-cell transcriptomic data comprising more than 100,000 cells of 20 organs and tissues of *Mus musculus* [7] and it is publicly hosted on AWS (accessible through the following Amazon Resource Name: arn:aws:s3:::czb-tabula-muris). In addition to *Tabula Muris*, 87 additional datasets exist from various disciplines that are all publicly available via the AWS *registry of open data* (`https://registry.opendata.aws`). Among them are *The Human Microbiome Project* (arn:aws:s3:::human-microbiome-project), *The International Cancer Genome Consortium* (arn:aws:s3:::oicr.icgc.meta/metadata), *Nanopore Reference Human Genome* (arn:aws:s3:::nanopore-human-wgs), and *1000 Genomes* (arn:aws:s3:::1000genomes).

A common scenario that demonstrates the need for robust authentication and authorization to cloud-based genomics data is joint analysis of public and private datasets. For instance, an active area of research in precision oncology is using omics data to statistically learn biomarker signatures to guide selection of therapies [16, 23, 6, 24] most likely to be effective for a particular tumor. In these studies, both private as well as public datasets such as TCGA require authorized access. Currently this kind of research requires putting private and public datasets on the same institutional computing cluster or cloud computing platform and running analyses on that cluster/platform. Moving omics data is costly, difficult, can be insecure depending on how access credentials are used. Our approach greatly simplifies joint analyses by providing a secure way for users to access data on one or more clouds. With our enhanced Galaxy, users can securely access and combine both private and public datasets onto a single Galaxy server where it can be analyzed together. This server can live behind an institutional firewall or on a commercial cloud computing platform and hence provide flexibility about where the final analysis is run.

Large-scale collaboration is another scenario where secure cloud-based authorization to genomics data is critical. For instance, collaborating labs across different institutes can host their data on the cloud and grant each member of those labs read (and write) access to the data. The challenge, however, is the ability to readily access that data for analysis. Typi-

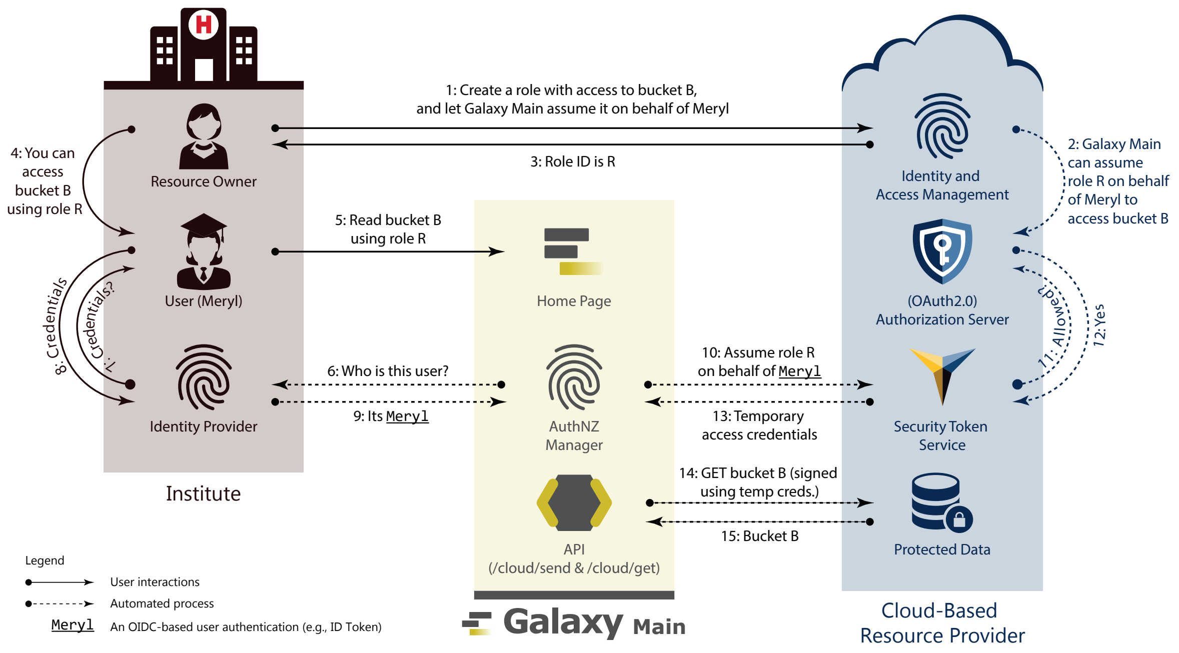
1: Create a role with access to bucket B, and let Galaxy Main assume it on behalf of Meryl
3: Role ID is R
2: Galaxy Main can assume role R on behalf of Meryl to access bucket B
4: You can access bucket B using role R
Resource Owner
User (Meryl)
Identity Provider
Institute
5: Read bucket B using role R
Home Page
6: Who is this user?
9: Its Meryl
7: Credentials?
8: Credentials
AuthNZ Manager
API (/cloud/send & /cloud/get)
10: Assume role R on behalf of Meryl
13: Temporary access credentials
14: GET bucket B (signed using temp creds.)
15: Bucket B
11: Allowed?
12: Yes
Identity and Access Management
(OAuth2.0) Authorization Server
Security Token Service
Protected Data
Cloud-Based Resource Provider
Galaxy Main
Legend
User interactions
Automated process
Meryl An OIDC-based user authentication (e.g., ID Token)

Figure 1: Galaxy adopts and integrates best-practice Web protocols to access secured data stored on cloud platforms (discussed in details in Section 2). In this approach, a *resource owner* shares protected data with collaborators (*User*) leveraging the role-based access model [21] and OpenID Connect protocol (OIDC). Accordingly, a resource owner defines a *role* with (read or write) access to protected data (e.g., see Figure 2), and specifies a Galaxy instance (defined using OIDC *audience* ID) that can assume the *role* upon presenting the user's identity token issued by their specified institute (OIDC IdP) for that Galaxy instance (e.g., see Figure 3). Upon successfully assuming the *role*, Galaxy receives cloud-provider-specific temporary credentials, and uses them to sign API requests to protected data. Note that following the OIDC requirements, all the discussed communications are TLS-protected (see Section 3.1). Additionally, a *resource owner* and *user* are not required to belong to a same trust group (e.g., institute).

cally, this is accomplished by either downloading the data onto on-premises resources or "mounting" it on cloud-based compute platforms. With Galaxy being widely adopted as a scalable, transparent, and reproducible data analysis platform, it is essential to enable Galaxy users to load their cloud-hosted, private data into their Galaxy *history* in an attestable and auditable manner. Accordingly, a resource owner can share cloud-hosted data with a user who is authenticated using their social or institutional identities. The user can then login to Galaxy using their specified identity, and request copying shared data into their *history*. Having analyzed the data, the user can request copying analysis results from the Galaxy *history* to the cloud-hosted storage, which enables them to share the analysis results with their collaborators. An illustration of this scenario is given in Figure 1, and a detailed discussion of the method is available in Section 2. For example, it is now possible to authenticate with Galaxy using a Google identity. Given a one-time, out-of-band setup where that identity is associated with an AWS S3 bucket role, the authenticated Galaxy user can seamlessly download and upload data to a private S3 bucket. All this is done without ever prompting the user for their AWS credentials.

2 Methods

Linking a cloud-based storage to Galaxy without requiring user credentials is realized by leveraging the OIDC protocol and the *CloudAuthz* library; this is implemented as a two-step authentication and authorization procedure. Authentication allows a user's identity to be validated while authorization verifies the privileges the given user has.

2.1 User authentication

Galaxy leverages the *authorization code flow* of the OIDC protocol to authenticate (and authorize) a user. In this flow, a user's identity is first verified by an IdP, then Galaxy receives security tokens (e.g., ID token and access token) from the IdP, which contains claims about the authentication and authorization of the user. The tokens are represented in cryptographically-signed JavaScript Object Notation (JSON) Web Token, JWTs, which ensures their integrity and immutability. In general, this flow is a two-step procedure described as follows.

First, the admin of a Galaxy instance sets up the instance for the *authorization code flow* by registering the instance with the IdP, and obtains security credentials for the instance (e.g., *Client ID* and *Client secret* as provided by Google). These credentials are used to ensure the authenticity of communications between the parties. For instance, an identity token issued for a user contains an *audience* claim (the *Client ID* of that Galaxy instance), which ensures that the token is issued for and can be used by the specified audience only.

Second, Galaxy authenticates a user (who wants to login to Galaxy using their third-party identity) by sending a request to an IdP. Among other information, the request incorporates:

- Security tokens of the Galaxy instance obtained when registering the instance (e.g., client ID and client secret as used with Google);
- Redirect URL; to be called upon successful authentication;
- Anti-forgery claims (e.g., *state* and *nonce* to prevent respectively cross-site request forgery (XSRF) and replay attacks).

Upon a successful authentication, an IdP sends an authorization *code* and the *state* token to the Galaxy instance. The Galaxy instance uses state token to validate the authenticity of the redirect message, and associate the authorization code with a user of the Galaxy instance. Then the Galaxy instance exchanges the authorization code for an ID token and a refresh token (which can be used to refresh an expired ID token) from the IdP.

2.2 Authorization Grant

In general, cloud-based resource providers leverage the role-based access control (RBAC) model [21] to grant authorization. However, each resource provider implements a proprietary procedure to authorize a client to assume a *role*. For instance, while an AWS role can be assumed using access key and secret key, a client has to provide subscription ID, client ID, client secret, and tenant ID to assume a role (service principal) on Microsoft Azure. However, the presented method is generic and can be used on any RBAC and OIDC-compliant resource provider. The following sections explain the method on AWS and Azure for defining and assuming a role, where a role is defined via the resource provider's web portal and it is assumed in Galaxy leveraging *CloudAuthz*.

2.2.1 AWS Temporary Credentials

An AWS *role* is an identity that can be assumed by an OIDC Relying Party (RP)—Galaxy in our scenario—on behalf of a user who is authenticated by an IdP. A role has certain permissions to specific resources (e.g., read access to a S3 object) that are defined using policies attached to it (e.g., see Figure 2). Through the AWS identity and access management web portal, a resource owner defines a role and a policy, and attaches the policy to the role. The resource owner

```
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
         "s3:ListAllMyBuckets",
         "s3:GetObject"
      ],
      "Resource": "*",
      "Condition": {
         "IpAddress": {
            "aws:SourceIp": "1.2.3.4"
         }
      }
   }]
}
```

Figure 2: A sample part of an AWS policy, which can be attached to a role to enable it to *read* buckets and download objects from S3, if the request is made from a server with `1.2.3.4` IP address. *Sid*: statement ID.

then defines a *trust* relation for the role, which defines the principals who are authorized to assume the role. The trust relation is defined using the *audience* ID of an RP, and authorizes the RP to assume the role on behalf of an IdP-authenticated user (e.g., see Figure 3). The *audience* ID is a required claim of an ID token that AWS security token service (STS) uses to assert if the token presented for assuming a role is issued for the RP defined in the trust relation. This mechanism prevents assuming a role using an ID token that is issued for a RP other than the one defined in the role's trust relation.

A user defines an AWS role for Galaxy using its Amazon resource name (ARN). Galaxy assumes the role by submitting a request to Amazon STS, which contains the role ARN and the user's ID token. AWS STS asserts the authenticity of the request by verifying with the IdP if the ID token is not expired and is issued for the audience specified in the ID token, and if the audience is trusted to assume the role. After a successful validation AWS STS responds to Galaxy, which among other information includes *access key ID*, *secret access key*, and *session token*. These credentials can be used to assume delegated privileges (e.g., read an AWS bucket) as defined in the policy attached to the role (see Figure 4). The temporary credentials are automatically refreshed by Galaxy, and can be restricted (update policy) and revoked by the resource owner.

```
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Effect": "Allow",
      "Principal": {
         "Federated": "accounts.google.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
         "StringEquals": {
            "accounts.google.com:aud": "8936...apps.googleusercontent.com"
         }
      }
   }]
}
```

Figure 3: An example of a trust relationship defined for an AWS role, which allows a Galaxy instance, identified by the `8936...apps.googleusercontent.com` (part of the client ID), to assume the role in exchange of a user's ID token issued for that Galaxy instance by Google.

2.2.2 Azure Service Principal

Azure resource manager leverages RBAC model [21] to enforce permissions. The actions an Azure role is authorized to perform are defined by its *permissions* and *scope*. Azure defines several built-in roles (e.g., the *Storage Blob Data Reader* role has read access to data and containers of Blob storage), and allows defining custom roles using Azure PowerShell or Azure CLI. An Azure role is assigned to a *security principal*, which defines a user, a group of users, or a *service principal*. A service principal is an identity used by applications or services. Accordingly, to authorize a Galaxy instance to access protected Azure resources, the resource owner defines a service principal and assigns an appropriate role to it.

A client can assume a service principal leveraging the *client credentials grant* flow of OAuth 2.0 protocol. This is a non-interactive flow and it is specifically designed for application-to-application communication. In this flow, client authentication (*ID* and *secret* of the service principal) is used as the authorization grant. Accordingly, neither the resource owner nor a Galaxy user is asked for a consent when client attempts to assume a service principal by a client, hence, this flow should be established between confidential clients only.

To assume an Azure role, a client requests an ac-

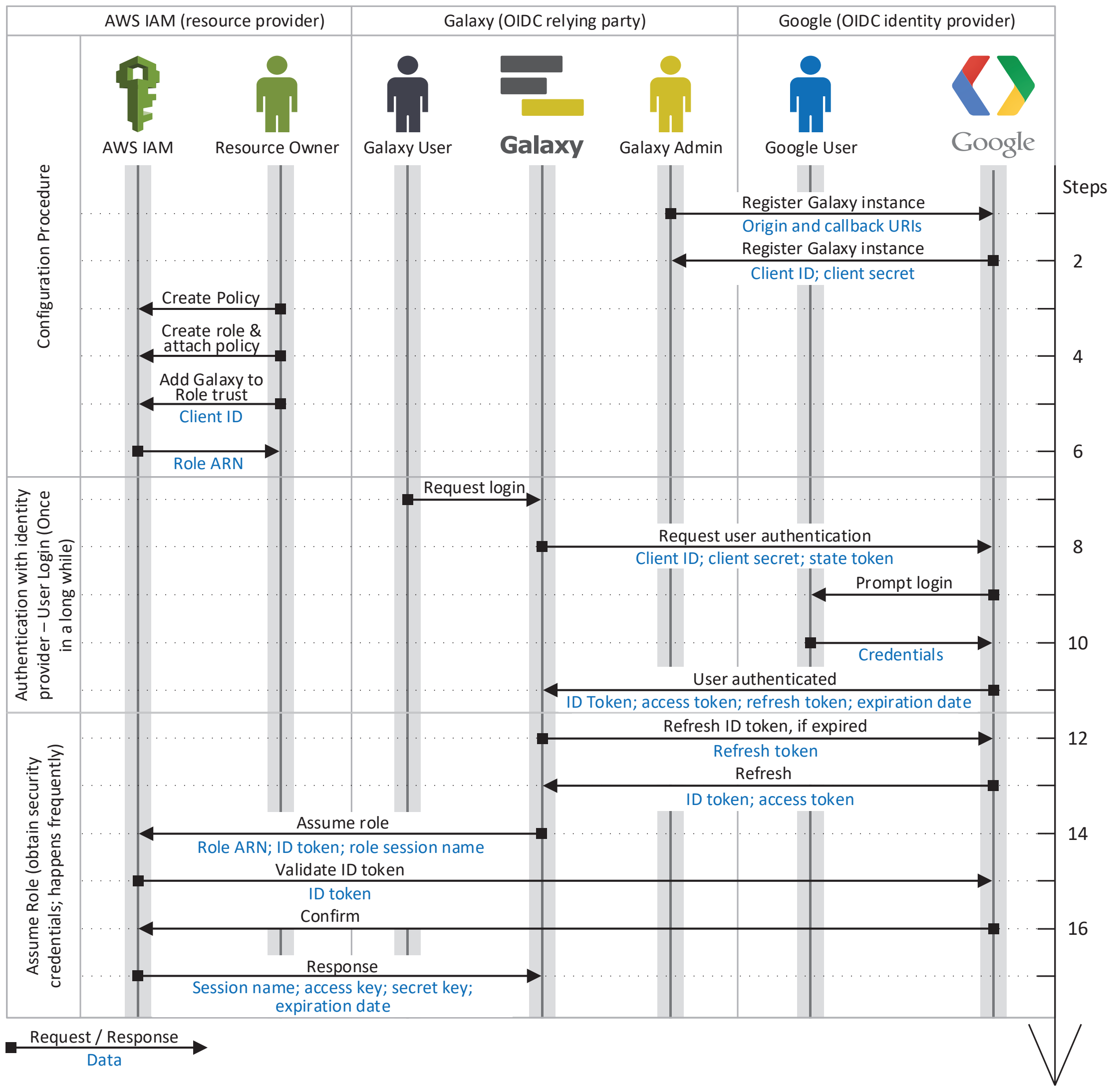
AWS IAM (resource provider)
Galaxy (OIDC relying party)
Google (OIDC identity provider)
AWS IAM
Resource Owner
Galaxy User
Galaxy
Galaxy Admin
Google User
Google
Steps
Configuration Procedure
Register Galaxy instance
Origin and callback URIs
Register Galaxy instance
Client ID; client secret
Create Policy
Create role & attach policy
Add Galaxy to Role trust
Client ID
Role ARN
Authentication with identity provider – User Login (Once in a long while)
Request login
Request user authentication
Client ID; client secret; state token
Prompt login
Credentials
User authenticated
ID Token; access token; refresh token; expiration date
Assume Role (obtain security credentials; happens frequently)
Refresh ID token, if expired
Refresh token
Refresh
ID token; access token
Assume role
Role ARN; ID token; role session name
Validate ID token
ID token
Confirm
Response
Session name; access key; secret key; expiration date
Request / Response
Data
2
4
6
8
10
12
14
16

Figure 4: Identity federation and authorization grant in the proposed protocol for AWS.

cess token from Azure's authorization server using its client credentials (see Figure 5). Upon a successful client authentication, the authorization server issues an access token for the client. The access token is issued for the application, independent from a user, and grants the client with privileges as defined by the role attached to the service principal. Following the client credentials grant flow, Azure's authorization server does not provide a *refresh token*; hence, an expired access token is refreshed by repeating the authorization process. Additionally, the authorization is revocable by removing the service principal, or changing its *secret*, or updating the role's assigned to it.

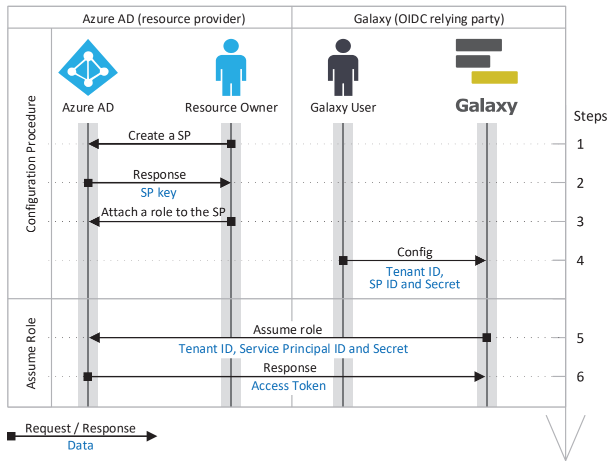

Figure 5: Identity federation and authorization grant in the proposed protocol for Azure. SP: service principal.

3 Results

Galaxy federates users identity and authentication using the OIDC protocol, which is the current industry standard. Accordingly, an identity provider authenticates a user to a Galaxy instance using temporary identity token that can be refreshed by that Galaxy instance only using a refresh token. Tokens represent *claims* about the authenticated users, and to verify the integrity of the tokens, they are represented in cryptographically *signed* JSON Web Tokens (JWTs). (A JWTs token is different from a cryptographically *encrypted* token, JWEs, where the claims in the token are encrypted and hidden from unauthorized parties.) Galaxy uses the authentication tokens to obtain cloud-native credentials to sign requests to a cloud-based resource provider's API and access protected resources (a detailed discussion is postponed to Section 3.2.2). To simplify the process, manual intervention of users is minimized (see figures 1, 4 and 7) and the tokens/secrets are never handed-out to end users.

In the remainder of this section, we discuss the advantages of the proposed method, related security challenges, and compare existing authentication and authorization protocols with our choices.

3.1 Countermeasures against eavesdropping attack

The eavesdropping attack is a type of the man-in-the-middle attack where the attacker sniffs and relays communication between parties (e.g., between Galaxy and AWS) and steals sensitive information such as identity tokens and/or access credentials. A common practice that we leverage to effectively thwart eavesdropper revolves around two principles; first, cryptographically secured communication channel between the parties. The OAuth2.0 protocol mandates transmitting tokens using Transport Layer Security (TLS) protocol (see sections 10.3 and 10.4 at `tools.ietf.org/html/rfc6749`). Accordingly, we recommend employing TLS to secure the communication between Galaxy and both identity and resource providers.

Second, OIDC identity and access tokens, and cloud-native credentials (generated by *CloudAuthz*) are short-term tokens with least privileges, which shortens the time-frame during which an eavesdropper can impersonate a Galaxy user when the TLS connection is exploited and tokens/credentials are stolen. The maximum age of tokens and cloud-native credentials is configurable in Galaxy by instance admins, and we recommend setting it to their minimum values (the default value is 3600 seconds). The `exp` claim of JWTs sets the expiration time of tokens; and since the tokens are cryptographically signed, the value of `exp` claim (among other claims) cannot be changed without invalidating the token. The expired tokens can be refreshed only by trusted parties using their secrets (e.g., *audience* ID and secret) and refresh tokens.

3.2 State-of-the-art of Fine-grained Medical Data Access Control in Cloud Computing

As the interest in data-driven healthcare continues to intensify, data security and privacy become imperative, which demands more robust and transparent data governance. Additionally, with the proliferation of biobanks and comparative data analysis methods, data sharing across institutes is becoming essential, which escalates data governance challenges. This scattering of datasets across organizations impedes data usage because accessing each requires a researcher to separately apply for access through a *data access committee*. The challenges are twofold, first, there has been a great deal of controversy evolving around *consent*. The Global Alliance for Genomics and Health (GA4GH) is fostering "consent codes" to facilitate data sharing, which divides data access conditions into nineteen empirical "categories" and "requirements" for consistent interpretation of data access and consent [11]. However, lack of consensus to the legal and ethical appropriateness of existing strategies hinders their adoption [5].

Second, determination and automatic enforcement of data usage restrictions and user authorizations. The Data Use Oversight System (`duos.broadinstitute.org`) is an attempt to define and enforce an ontology of data access restrictions. The GA4GH has launched a pilot study, named "library card" [4], to standardize a role-based authentication and authorization of researchers by augmenting the widely-adopted protocols such as OIDC; it is envisioned to encode "bona fides" of researchers as a set of standardized *claims*. Additionally, GA4GH has defined "registered access", an empirical data access model that leverages "consent codes" and "library card" for authentication, attestation, and authorization of researchers access to protected data [9, 10]. The different protocols for authentication and authorization are discussed in the following sections.

3.2.1 User Authentication Protocols

A principle component to data sharing and their access control is user identification and authentication across institutes and resource/service providers. There has been a number of protocols developed for this purpose, such as Lightweight Directory Access Protocol (LDAP), Security Assertion Markup Language (SAML), OASIS WS-* (Security, Trust, and Federation), OAuth, and OpenID Connect (OIDC). These protocols are used in widely adopted services such as *Shibboleth*, which leverages SAML protocol to enable *single sign-on* across organizations. For instance, using Shibboleth, a university X (Shibboleth Identity Provider) with subscription to a publisher Y (Shibboleth Service Provider) can enable its students to login to Y and access subscription-required articles.

OIDC is the state-of-the-art authentication and authorization protocol supported by major social identity providers and cloud-based resource providers such as Amazon, Google, and Microsoft. Accordingly, we use the OIDC protocol for Galaxy's identity and access management (see Figure 6), and the remainder of this section is scoped to OIDC-based approaches. Leveraging OIDC-certified libraries (see `openid.net/developers/certified/` for their list) any platform can act as an OIDC identity provider; however, to allow users to login to Galaxy using their institutional or social identities, and security challenges of implementing and maintaining an OIDC IdP for developers and Galaxy instance admins (e.g., counterfeits weaknesses covered in `tools.ietf.org/html/rfc6819`), we are akin to rely on external identities.

There exists two architectural approaches for user authentication using their external identities: *direct* and *brokered* [18, 14]. In the *direct* authentication pattern, a Galaxy instance (client) directly establishes a *trust* relation (i.e., communicate following a standard protocol such as OIDC) with an IdP, acting as an RP, where the IdP issues identity tokens with `aud` claim being the audience ID of that Galaxy instance. The *direct* authentication is a decentralized pattern, where users are authenticated to different Galaxy instances independently. Using a decentralized pattern, a breached *trust* relation is isolated and cannot affect other *trust* relations. Additionally, admins of Galaxy instances can independently choose IdPs following their institutional policies. However, to interface with multiple IdPs following this pattern, Galaxy needs to implement every IdP-specific *trust* relation.

The *brokered* pattern leverages an authentication broker; an intermediary service of a *single sign-on* architecture that establishes a *trust* relation between multiple IdPs and service providers, and it is *trusted* by both parties independently. (In other words, a broker can use different authentication and authorization protocols to communicate with IdP and Galaxy.) A broker may decouple parties using an internal user identity, and vouches for the user by issuing its own identity tokens to the clients (see Figure 7) [18]. An advantage of this design is the ability to impersonate a user by masking their login username by the internal identity of the broker to Galaxy. Additionally, an authentication broker can negotiate *trust* between Galaxy and IdPs, which removes the need for direct relation with IdPs. However, the brokered pattern is a centralized approach, where users are authenticated to various Galaxy instances using shared identities. Accordingly, a brokered pattern establishes a single point of failure and a central breach point; a security and liability concern [12]. If compromised, it can jeopardize the security of users on all connected Galaxy instances. If it fails, none of the parties can communicate; however, with the cost of increased design complexity, this problem can be mitigated by significant number of redundant and mirrored brokers. Additionally, some Galaxy instances may not be commissioned to interface brokers or use shared identities due to institutional policies and *consent* concerns.

In spite of the plain core premise of *direct* and *brokered* patterns (as described in [18]), the specification has a myriad of options and variations, which makes it difficult to draw clear boundaries between the available implementations. Some IdPs provide libraries for *direct* user authentication, for instance Google (`developers.google.com/identity/protocols/OAuth2`) and Microsoft (`docs.microsoft.com/en-us/azure/active-directory/develop/reference-v2-libraries`). Python Social Auth (`github.com/python-social-auth/social-core`) implements IdP-specific trust relations for common social identity providers and exposes them via a common interface, which simplifies using the *direct* authentication pattern with multiple IdPs. A decent number of services are available for the *brokered* pattern authentication, spanning from commercial products such as Amazon Cognito (`aws.amazon.com/cognito/`) and Okta (`www.okta.com`), to free and open-source services such as Keycloak (`www.keycloak.org`), CILogon (`www.cilogon.org/oidc`), and Fence (`github.com/uc-cdis/fence`).

Galaxy needs to authenticate users in a heterogeneous environment of authentication and authorization approaches. An authentication broker can negotiate trust between Galaxy and IdPs (and service providers), which removes the need for direct relation with IdPs. Meanwhile, using libraries such as Python Social Auth, Galaxy can establish a trust with multiple IdPs through a common interface, without the need for an IdP-specific implementation. Accordingly, we leverage the *direct* authentication pattern, and use Python Social Auth to establish a *trust* relation with IdPs.

3.2.2 Protocols for User Authorization to Cloud-Based Resources

Cloud-based resource providers commonly implement two methods to authorize access rights to secured resources: *signed URLs*, and *cloud-native credentials*. *Signed URLs* grant a party in possession of the URL with particular access to specific resources determined at the URL generation by the resource owner. Signed URLs allow resource owners to grant temporary access to users who are not required to be authenticated by the resource provider. While signed URLs simplify data sharing, it is challenging to audit the access to the data shared using signed URLs. Additionally, the enforcement of a fine-grained access control on a large scale would be challenging since the URLs are generated on a per-resource basis.

Cloud-native credentials, as the name implies, are provider-specific secrets to sign programmatic requests to the provider (e.g., Application Programming Interface (API) requests). Providers commonly offer long and short-term credentials. A resource owner can obtain long-term secrets from the provider's portal, and use them to authorize a client's (e.g., a web or native app) access to secured resources. However, such credentials are commonly obtained via a manual intervention that demands a degree of familiarity with the resource provider's portal. Additionally, long-term nature of such credentials commonly persuades scenarios where the tokens are embedded or distributed in applications, which increase the risk of tokens being hijacked.

Accordingly, some cloud-based resource providers (e.g., Amazon and Microsoft) implement *security token service* (STS), specifications of which is defined as part of OASIS WS-Trust and WS-Federation protocols [17]. The STS that commonly maps to (OAuth2.0) authorization server and is intended to be used by native and web apps, issues short-term security credentials upon a successful assertion of user authentication (see Figure 1). Since such tokens are emitted on-the-fly as per API requests, obtaining them does not require a manual intervention of the resource owner. Additionally, the short-term credentials are not valid after their limited lifetime, whose advantages are twofold, first it does not necessitate revoking them when they are no longer needed. Second, if (e.g., as a result of a design flaw) short-term credentials are distributed or embed with applications, they cannot impose security risks after their limited lifetime.

However, both long and short-term secrets grant a client with the same level of privileges as the resource owner. Accordingly, resource providers leverage RBAC [21] and allow resource owners to define *roles*. A *role* (or a *service principal* in Azure terminology) is an identity that can be assumed by clients using specified users authentication, whose privileges are defined independent from the resource owner. Coupling STS with RBAC, resource providers emit short-term credentials per successful assertion of user authentication.

Some resource providers (e.g., Amazon) use federated identities and enable assuming a role using authentications issued for specified clients by determined IdPs (see Figure 3). Leveraging this model, a client can obtain short-term credentials on behalf of users who are not necessarily part of the resource owner's cloud subscription account, which is advantageous for sharing data without adding collaborators to a cloud subscription account.

Therefore, Galaxy obtains user authorization to protected cloud-based resources, leveraging RBAC model, from resource provider's STS. Since resource providers expose STS and RBAC differently, Galaxy uses *CloudAuthz* that provides a common interface to various resource providers, it supports AWS and Azure, and Google is under development (see Figure 6). Amazon Cognito provides AWS STS and RBAC functionality for native and web apps, and Fence implements an interface to Google and AWS STS.

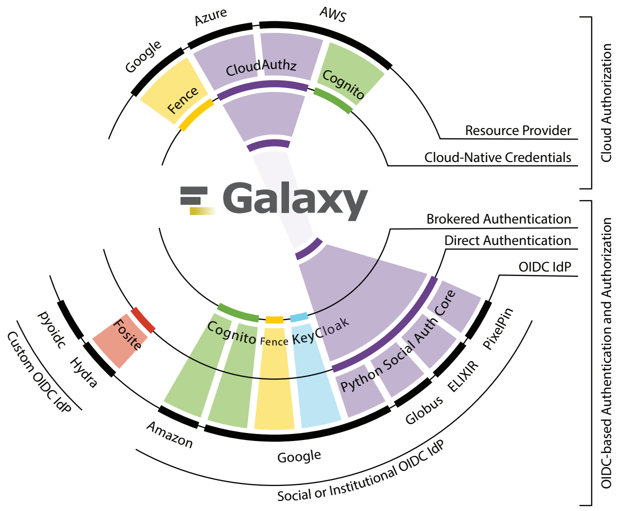

Figure 6: Illustrates a subset of available methods and implementations for user authentication and authorization grant to cloud-based resource providers, and the back-ends each method supports. The figure is scoped to only OIDC-based authentication and authorization grant using cloud-native credentials. The method and implementations we use in Galaxy are highlighted in *purple*, which are Python Social Auth for user authentication, and *CloudAuthz* for granting cloud authorization.

4 Discussion

In this paper, we have described a robust, generalized, and secure approach for accessing biomedical data across multiple cloud computing platforms. We use best-practice Web approaches so that user credentials are never requested, transmitted, or stored by principles other than resource or identity providers. In general, leveraging the RBAC

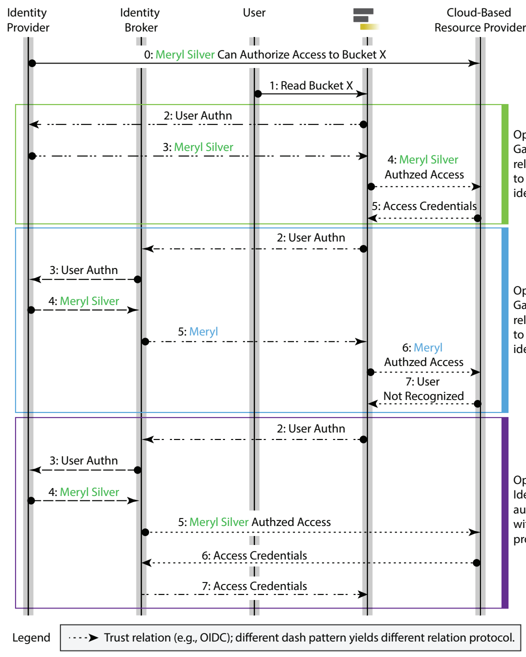
Identity Provider
Identity Broker
User
Cloud-Based Resource Provider
0: Meryl Silver Can Authorize Access to Bucket X
1: Read Bucket X
2: User Authn
3: Meryl Silver
4: Meryl Silver Authzed Access
5: Access Credentials
Option 1: Galaxy is the relying party to the identity provider.
2: User Authn
3: User Authn
4: Meryl Silver
5: Meryl
6: Meryl Authzed Access
7: User Not Recognized
Option 2: Galaxy is the relying party to the identity broker.
2: User Authn
3: User Authn
4: Meryl Silver
5: Meryl Silver Authzed Access
6: Access Credentials
7: Access Credentials
Option 3: Identity broker authzes access with resource provider.
Legend
Trust relation (e.g., OIDC); different dash pattern yields different relation protocol.

Figure 7: Illustrates three patterns of user authentication and cloud authorization. The *Option 1* is based on *direct* authentication protocol, which we currently implement. The *Option 2* is based on brokered authentication pattern, and since methods implementing this protocol can map an authenticated user to a local identity (see steps 4 and 5 of *Option 2*: the broker emits its own authentication instead of relaying the IdP's proof), this protocol cannot be used for authorization grant to cloud-based resource providers. The *Option 3* also follows *brokered* authentication pattern, but since it also provides authorization grant service (Amazon Cognito is such a broker), it can be used as an alternative to *Option 1*.

model and the OIDC protocol, the proposed method securely grants clients with authorization to protected cloud-based resources by mutually satisfying the principals about each other's authentication and authorization. Additionally, the proposed method follows the principle of least privilege, and allows the resource owner to revoke and restrict a client's authorization at any time, and independent from other granted authorizations. The proposed method is implemented in the Galaxy platform, which is used across the world for large-scale biomedical analyses. The result of this integration is that Galaxy users can securely and seamlessly access their protected cloud-hosted (and potentially sensitive) genomic data and use that data in Galaxy to run complex, integrated analyses.

Our work enabling Galaxy to leverage RBAC model and use OIDC protocols to securely access biomedical datasets across cloud computing platforms is a first step toward developing a user-friendly yet functional platform for analysis of distributed biomedical data [2]. Galaxy is currently implemented as a monolithic application, which we are changing toward a collection of micro-services; in this regard, our next step is to implement a *common identity* across Galaxy servers and services. A common identity paves the path toward a coherent user experience across different Galaxy instances, where their data, workflows and analysis histories are unified. A common identity can be realized by out-sourcing user authentication and authorization as an independent and "globally-accessible" AuthNZ service.

Funding

This work was supported by the National Institutes of Health [HG006620 and CA231877], the National Science Foundation [DBI 1661497]; and Oregon Health and Science University.

References

[1] Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn A Grüning, et al. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46(W1):W537–W544, 2018.

[2] Enis Afgan, Vahid Jalili, Nuwan Goonasekera, James Taylor, and Jeremy Goecks. Federated galaxy: Biomedical computing at the frontier. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 871–874. IEEE, 2018.

[3] Enis Afgan, Andrew Lonie, James Taylor, and Nuwan Goonasekera. Cloudlaunch: Discover and deploy cloud applications. *Future Generation Computer Systems*, 2018.

[4] Moran N Cabili, Knox Carey, Stephanie OM Dyke, Anthony J Brookes, Marc Fiume, Francis Jeanson, Giselle Kerry, Alex Lash, Heidi Sofia, Dylan Spalding, et al. Simplifying research access to genomics and health data with library cards. *Scientific data*, 5, 2018.

[5] Timothy Caulfield and Blake Murdoch. Genes, cells, and biobanks: Yes, there's still a consent problem. *PLoS biology*, 15(7):e2002654, 2017.

[6] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[7] Tabula Muris Consortium et al. Single-cell transcriptomics of 20 mouse organs creates a tabula muris. *Nature*, 562(7727):367, 2018.

[8] Somalee Datta, Keith Bettinger, and Michael Snyder. Secure cloud computing for genomic data. *Nature biotechnology*, 34(6):588, 2016.

[9] Stephanie OM Dyke, Emily Kirby, Mahsa Shabani, Adrian Thorogood, Kazuto Kato, and Bartha M Knoppers. Registered access: a 'triple-a'approach. *European Journal of Human Genetics*, 24(12):1676, 2016.

[10] Stephanie OM Dyke, Mikael Linden, Ilkka Lappalainen, Jordi Rambla De Argila, Knox Carey, David Lloyd, J Dylan Spalding, Moran N Cabili, Giselle Kerry, Julia Foreman, et al. Registered access: authorizing data access. *European Journal of Human Genetics*, page 1, 2018.

[11] Stephanie OM Dyke, Anthony A Philippakis, Jordi Rambla De Argila, Dina N Paltoo, Erin S Luetkemeier, Bartha M Knoppers, Anthony J Brookes, J Dylan Spalding, Mark Thompson, Marco Roos, et al. Consent codes: upholding standard data use conditions. *PLoS genetics*, 12(1):e1005772, 2016.

[12] Thomas Erl, Stephen G Bennett, Benjamin Carlyle, Clive Gee, Robert Laird, Anne Thomas Manes, Robert Moores, and Andre Tost. *SOA governance: governing shared services on-premise and in the cloud*. Pearson Education, 2011.

[13] Ian Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP international conference on network and parallel computing*, pages 2–13. Springer, 2005.

[14] Jason Hogg. *Web service security: Scenarios, patterns, and implementation guidance for Web Services Enhancements (WSE) 3.0*. " O'Reilly Media, Inc.", 2005.

[15] Ben Langmead and Abhinav Nellore. Cloud computing for genomic data analysis and collaboration. *Nature Reviews Genetics*, 19(4):208, 2018.

[16] Daniel J McGrail, Curtis Chun-Jen Lin, Jeannine Garnett, Qingxin Liu, Wei Mo, Hui Dai, Yiling Lu, Qinghua Yu, Zhenlin Ju, Jun Yin, et al. Improved prediction of parp inhibitor response and identification of synergizing agents through use of a novel gene expression signature generation algorithm. *NPJ systems biology and applications*, 3(1):8, 2017.

[17] Anthony Nadalin, Marc Goodner, Martin Gudgin, Abbie Barbir, and Hans Granqvist. Oasis ws-trust 1.4. *Specification Version*, 1:41–45, 2008.

[18] Radia Perlman, Charlie Kaufman, and Mike Speciner. *Network security: private communication in a public world*. Pearson Education India, 2016.

[19] Kun Qu, Sara Garamszegi, Felix Wu, Helga Thorvaldsdottir, Ted Liefeld, Marco Ocana, Diego Borges-Rivera, Nathalie Pochet, James T Robinson, Barry Demchak, et al. Integrative genomic analysis by interoperation of bioinformatics tools in genomespace. *nature methods*, 13(3):245, 2016.

[20] Nat Sakimura, John Bradley, Michael B Jones, Breno de Medeiros, and Chuck Mortimore. Openid connect core 1.0 incorporating errata set 1. `http://openid.net/specs/openid-connect-core-1_0.html`, 2014.

[21] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[22] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. Big data: astronomical or genomical? *PLoS biology*, 13(7):e1002195, 2015.

[23] Yue Wang, Kenneth MK Mark, Matthew H Ung, Arminja Kettenbach, Todd Miller, Wei Xu, Wenqing Cheng, Tian Xia, and Chao Cheng. Application of rnai-induced gene expression profiles for prognostic prediction in breast cancer. *Genome medicine*, 8(1):114, 2016.

[24] Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M Hoffman. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion*, 50:71–91, 2019.