Integrating an AAA-based federation mechanism for OpenStack - The CLASSe view

Alejandro Pérez Méndez (alex@um.es)
Gabriel López Millán (gabilm@um.es)
Rafael Marín López (rafa@um.es)
Dept. of Information and Communications Engineering, University of Murcia, ES

David W. Chadwick (d.w.chadwick@kent.ac.uk)
School of Computing, University of Kent, UK

Ioram Schechtman Sette (iss@cin.ufpe.br)
Informatics Center (CIn), Federal University of Pernambuco (UFPE), BR
Recife Center for Advanced Studies and System (CESAR), BR

Abstract

Identity federations enable users, service providers and identity providers from different organizations to exchange authentication and authorization information in a secure way. In this paper we present a novel identity federation architecture for cloud services based on the integration of a cloud identity management service with an Authentication, Authorization and Accounting (AAA) infrastructure. Specifically we analyse how this type of AAA-based federation can be smoothly integrated into OpenStack, the leading open source cloud software solution, using the IETF Application Bridging for Federated Access Beyond web (ABFAB) specification for authentication and authorization. We provide details of the implementation undertaken in GÉANT's CLASSe project, and show its validation in a real testbed.

Keywords: Identity federation, ABFAB, OpenStack, Cloud, authentication, authorization, GÉANT, CLASSe

1 Introduction

The increasing amount of shared services and data across networks has also increased the security concerns about how to protect them, and how to enable authenticated and authorized users to gain access to them in a safe way. One of the most deployed solutions to protect these widely distributed services is the establishment of identity federations. Identity federations enable a user, who is affiliated with a particular organization (the Identity Provider - IdP), to access the services provided by a second organization in the federation (the Service Provider - SP) [COMST]. Typically the user is authenticated by the IdP, which provides an authentication assertion to the SP proving she has been successfully authenticated in her home organization. After that, the SP may run an authorization process to check whether the user possesses the required permissions to access the requested service. This authorization decision can be based on authorization data (or attributes) local to the SP or received from the IdP. The federation defines and regulates the trust relationships between the different IdPs and SPs to make this possible.

Traditionally, identity federations have been based on technologies such as SAML [SAML2], OAuth [OAUTH], OpenID [OPENID] or more recently OpenID Connect [OIDC], though the scope has been generally limited to web applications and services. One of the most successful examples of web identity federation infrastructures is eduGAIN [EDUGAIN]. eduGAIN is a service that interconnects about 38 federation members composed by more that 3,200 educational and research institutions around the globe.

Network based federations on the other hand have been traditionally based on Authentication, Authorization and Accounting (AAA) trust infrastructures [RFC2903], making use of the so-called AAA protocols, such as RADIUS [RADIUS] or DIAMETER [DIAMETER]. Examples of AAA identity federations can be found in eduroam [EDUROAM] and in 3GPP networks [3GPP]. In fact, eduroam is a world-wide network roaming access service deployed in 76 countries and available to thousands of institutions. Both eduGAIN and eduroam are initiatives being deployed under the umbrella of the European GÉANT consortium [GEANT].

More recently, emerging *non-web* networking scenarios, such as the Cloud, the Grid, and mobile devices, are demanding a federation technology that can cater for their need to support CLIs (Command Line Interfaces) or bespoke applications without involving excessive complexity or web browsers. Organizations such as the IETF, Cloud Security Alliance and OASIS have echoed this need [CSA] [OASIS]. For this reason, the IETF created the Application Bridging for Federated Access Beyond web (ABFAB) WG [ABFAB], in order to promote the development of a single unifying technology for extending the benefits of federated identity to a broad range of application services (cloud, e-mail, file storage, remote access, instant messaging, etc.).

In ABFAB, the end user, through a Client application, authenticates with its IdP in order to get access to a particular SP (also called the Relying Party (RP) in the ABFAB context). The Client requests access to the RP by means of the well-known GSS-API [GSSAPI] and the underlying application protocol. The RP then contacts the IdP in order to launch the authentication and authorization process. To achieve this, an AAA infrastructure interconnecting IdPs and RPs is deployed. The federated authentication is performed using the *Extensible Authentication Protocol* (EAP) [EAP] between the Client application and the IdP. EAP packets between the Client and the RP are transported using a new GSS-API mechanism called GSS-EAP [GSSEAP], whereas the AAA infrastructure is leveraged to deliver the EAP packets between RP and IdP. After the authentication succeeds, the IdP may deliver SAML assertions for authorization purposes to the RP including information (i.e. identity attributes) about the user.

This new trend for Identity Federations Beyond-Web is gaining devotees, as organizations such as JANET and RedIRIS, the British and Spanish NRENs (National Research and Education Networks) respectively, are now promoting the deployment of ABFAB technologies in their countries [USEC]. Moreover, the GÉANT project [GEANT], involving most European NRENs, is also establishing a pilot for these technologies, in anticipation of its future convergence with its already well-established eduroam service that has hundreds of thousands of users per day.

Aimed at the integration of Cloud services and non-web based identity federations, the European research project *CLASSe* (Cloud ABFAB Federation Services in eduroam) [CLASSE], scoped under the GÉANT GN3Plus Opencall umbrella, was launched to investigate how to provide federated access to cloud services using ABFAB. As a reference implementation of cloud services, CLASSe used OpenStack [OSTACK], since it is one of the most popular open source cloud computing packages.

This paper presents the results of the CLASSe project and analyzes how identity federations based on AAA infrastructures can be smoothly integrated in OpenStack. In particular, it describes how OpenStack can take advantage of the work being standardized in the ABFAB WG, given a powerful authentication and authorization mechanism for cloud users and services when cloud providers share an AAA infrastructure. Moreover, it also provides general guidelines describing how other Cloud solutions could integrate with the ABFAB technologies. This kind of support fills the existing gap of using CLI (Command Line Interface) applications when enabling federated access in Cloud scenarios.

In order to demonstrate the feasibility of this proposal we extended the OpenStack software to make use of the reference ABFAB implementation, Moonshot [MSHOT], and deployed a testbed scenario making use of this software. A performance analysis is also described showing the computational and time requirements for these kinds of scenarios. It is worth mentioning that this proposal also fits in those scenarios proposed by ETSI's 3GPP where AAA networks and application services, including cloud services, are combined, for example in [IMSCCA] [C3GPPIMS].

The rest of this paper is structured as follows: section 2 describes the background technologies. Section 3 summarizes some related work on integrating OpenStack with different authentication mechanisms. Section 4 describes how current versions of Openstack handle federated authentication and authorisation. Section 5 describes conceptually how OpenStack can be integrated with the ABFAB protocol suite to support AAA federation. Section 6 describes our implementation and demonstration service, using the Moonshot software [MSHOT]. Finally, section 7 concludes and describes where further work is still needed.

2 Background Technologies

This section introduces the reader to the two main technologies that are the basis of this work: OpenStack, one of the most popular open source implementations of cloud software, and ABFAB, the IETF's set of standards for identity federations for non-web based services.

2.1 OpenStack and Keystone

OpenStack is a cloud software composed of several modules providing different cloud services: *Nova* for virtual computing, *Neutron* for networking, *Swift* for file and object storage, etc. These modules communicate with each other and with external entities through HTTP RESTFUL [REST] APIs. One of these modules, *Keystone*, provides the identity services

(authentication and authorization) to the rest of the modules in OpenStack. As such Keystone is a critical component of the infrastructure.

Keystone supports multiple kinds of identity credentials (e.g. username/password or X.509 certificates) and authentication technologies (Kerberos, LDAP, etc.). Prior to the addition of federated access, OpenStack typically required a backend LDAP [LDAP] database to hold the users, groups, roles and project information. Since 2014, Keystone has supported federated access, although the first version (in the Icehouse release) only supported the SAMLv2 protocol [SAML2].

Prior research [FEDOS] added protocol independent federated identity management directly into the Keystone Grizzly and Havana releases, but this design supported the federation protocol handling via Keystone plugins. The Keystone core developers decided not to support any federation protocol handling in their core code, so opted instead to use an Apache [APACHE] front end to handle these, since many different applications already successfully use Apache for federated login. Since then, and as part of the CLASSe project, the authors worked closely with the Keystone core group in order to remove any SAML dependencies from the core code, so that alternative federation protocols, such as ABFAB, could be used, providing there was an appropriate Apache authentication module.

Briefly, OpenStack uses a type of *Role Based Access Control* (RBAC), although it is not pure NIST RBAC [NIST], since *<Users>* and *<Groups>* are assigned to *<Roles>* and *<Projects>* pairs, rather than simply to *<Roles>. <Roles>* are given permission to access project resources, usually through rules in policy files (separate policy files are used by each OpenStack service: Keystone, Nova, Glance, etc.) although Swift uses conventional access control lists. The RBAC objects are defined in Keystone as follows:

- *<User>:* Represents a registered user in a particular OpenStack instance.
- *<Group>:* Represents a collection of *<User>* objects.
- *<Project>:* Represents a set of resources that are available on the OpenStack instance (e.g. set of virtual machines, virtual containers, etc.).
- *<Role>.* Represents a set of rights and privileges that delimit which actions a *<User>* or a *<Group>* can perform over the different resources associated to a *<Project>.*

For instance, the *<User>* named *Alice* might have been assigned the *<Role> Admin* (giving full access to all the resources) for the *<Project> TestDeployment*, and the *<Role> Member* (giving read-only access) in the *<Project> ProductionDeployment*.

Keystone was modeled on Kerberos [KERBEROS], in that after the user has successfully authenticated (s)he is given a Keystone token, called an *unscoped token*, which is equivalent to the Kerberos Ticket Granting Ticket (TGT). The user now chooses which *<Project>* (s)he wishes to be associated with, and his unscoped token is swapped for an appropriate *scoped token* (equivalent to a Kerberos Service Ticket (ST)) that lists his *<Roles>* in the chosen

<Project>. The scoped token allows the user to talk to the various OpenStack services and be granted access according to its RBAC policy for the project.

Three types of tokens are currently supported by Keystone: a PKI based token that allows the OpenStack services to validate the user's privileges locally without contacting Keystone, a UUID opaque token that services have to pass back to Keystone for validation, and a Fernet token that likewise has to be passed back to Keystone for full validation. UUID tokens are randomly generated 32 character strings that have to be stored by Keystone, whereas Fernet tokens contain minimal identity and authorization information about the user and are symmetrically encrypted using AES-CBC and signed using a SHA256 HMAC. PKI tokens on the other hand are much larger, several kilobytes in fact, and contain all the identity and authorization information about the user. They are asymmetrically signed. Due to their size and poor performance they are gradually being phased out.

Unfortunately, all Keystone tokens are bearer tokens, which means that if any token is intercepted or copied from a user's machine by an attacker, the attacker will be able to masquerade as the user. It has been a long running debate inside the Keystone group [KS-GROUP] when or if to make the tokens non-bearer, by making the users prove possession of them, but it is not known when this enhancement might be added.

From the point of view of identity federation, one can see that Keystone acts very similarly to an Identity Provider in federated access, in that the user authenticates to Keystone and is given an assertion (the scoped token) that allows the user to talk to the various OpenStack services. We used this in our original federation design [FEDOS] to add support to Keystone to receive an assertion from a federated IdP as proof of user authentication. Keystone then swapped the IdP's assertion for an unscoped token, and then subsequently for a scoped token.

2.2 ABFAB

ABFAB promotes the development of a single unifying technology for federated identity that is valid for any kind of application service. The ABFAB WG has proposed several standards, which are being implemented in the Moonshot project [MSHOT]. ABFAB is based on a widely accepted set of security protocols such as AAA, EAP, GSS-API and SAML, and it is composed of three main entities:

- Client application, representing the user wishing to access a particular service or application,
- Relying Party (RP) that controls access to the service,
- Identity Provider (IdP) that verifies the user's credentials and provides assertions about her identity (e.g. affiliation, entitlement, location) to the RP.

As Figure 1 describes, the application protocol (e.g. FTP, SSH, HTTP, or SMTP) is executed between the Client and the RP. On top of this protocol, as part of an access control procedure, a GSS-API security context must be successfully established between the Client and RP, playing the roles of *GSS Initiator* and *GSS Acceptor* respectively. This GSS context mutually

authenticates both entities. Nowadays, many application services either provide direct support for GSS-API, or it can be integrated by means of the *GS2 Mechanism Family [GS2]* and the SASL (*Simple Authentication and Security Layer*) [SASL] protocols.

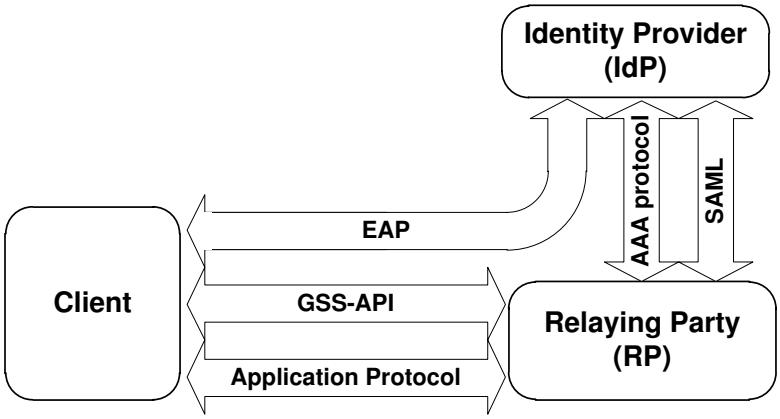

Figure 1. ABFAB architecture

Although the user identification process is performed using the GSS-API, the actual authentication process is carried out by means of EAP. ABFAB has defined the *GSS-API Mechanism for the Extensible Authentication Protocol* (GSS-EAP) [GSSEAP], which specifies how EAP packets can be transported in GSS tokens, and how the key material exported by EAP can be used to provide security services (e.g. confidentiality and integrity protection).

Finally, the AAA protocol is used between the RP and the IdP in order to provide the federation substrate, that is, it implements the trust relationships that form the federation. It has two purposes: the transport of EAP packets between them, and the delivery from the IdP to the RP of the user's authorization attributes represented as SAML assertions (attribute statements). Although any AAA protocol could potentially be used, ABFAB has focused on RADIUS, due to its widespread deployment and acceptance.

ABFAB can provide privacy protection by means of the pseudonymity mechanisms of its underlying technologies. In particular, using an EAP tunneling method (e.g. EAP-TTLS [EAPTTLS] or PEAP [PEAP]) can hide the end user's identifier from the RP, whereas SAML [SAML2] defines how assertions can make use of pseudonyms for the same purpose.

Once the end user is authenticated, the IdP generates (or obtains) the first SAML statement with information about the end user and sends it to the RP, as described in [SAML-AAA]. Moreover, this document also details how the RP and IdP can make use of the *SAML Assertion Query/Response profile* to issue further attribute requests in order to improve the authorization decision process.

3 Related Work

Beside the already described traditional OpenStack authentication methods provided by Keystone, additional authentication mechanisms can be added as plugins (which is why we originally added federated access to OpenStack in this way [FEDOS]).

The integration of Kerberos into OpenStack to provide advanced authentication mechanisms can be found here [OS-KRB]. Kerberos also supports identity federation, by means of the so-called Kerberos cross-realm mechanism. Therefore, it would have been possible to provide identity federation to OpenStack following the standard Kerberos cross-realm model. However, the deployment of Kerberos cross-realm infrastructures has been scarce, due to some recognized issues [RFC5868]. Thus, the usage of Kerberos has been limited to controlling the access of local users registered in the service's domain. Recently, some alternatives, such as FedKERB [FEDKRB], PanaKERB [PANAKRB] and EduKERB [EDUKRB], have been proposed to obtain the benefits of Kerberos without deploying an alternative cross-realm infrastructure, by using a more common and widely used federation substrate: the AAA-based federation. However, these solutions add or inflict changes in several entities, and require new amendments to standards and related implementations. In particular, FedKERB requires modifying the Kerberos's Key Distribution Center (KDC), which is the central entity in charge of distributing key material, in order to implement a pre-authentication mechanism for integration with the AAA infrastructure. PanaKERB requires adding a new protocol (PANA [PANA]) to bootstrap Kerberos credentials in the service's domain. Finally, EduKERB modifies the KDC to consume a non-standardized token obtained during network access authentication. Thus we discounted the use of Kerberos for federated access.

The IETF is also currently working on a SASL and GSS-API mechanism for SAML 2.0 that leverages the capabilities of an enhanced client to address federated authentication reusing existing SAML bindings and profiles [SASL-SAML-EC]. In this way, SAML can be used for arbitrary applications as long as they support either the SASL framework or GSS-API (as happens with Keystone, based on the HTTP protocol). The main advantages of this approach is that it is under standardization in the IETF WG, and it requires little modifications to existing SAML capable RPs and IdPs. Its main disadvantage is that the Client Application must first contact the SAML IdP in order to request the SAML authentication statement. This would require KeyStone to support SAML, which is not natively integrated.

4 Identity Federations in OpenStack

The latest versions of OpenStack (i.e. Juno, Kilo and Liberty) have included generic support for federated authentication via an Apache front end. After Apache receives the user's identity attributes from the IdP, it passes these to Keystone as environmental variables. In this way, Keystone does not need to parse or process the federation protocol messages, since no matter what the federation protocol is, the identity attributes are transferred by Apache in the same way. This gives OpenStack the ability to permit users not registered with Keystone, but with external IdPs, to obtain *unscoped/scoped tokens.*

In order to enable federated authentication, the Keystone administrator must first define which external IdPs are trusted to authenticate users. Keystone will then reject any user authenticated by an unlisted IdP. Next, since federated users do not exist in Keystone's database, the administrator must define how their federated identity attributes will be mapped into existing *<Users>* or *<Groups>*. In this way, federated users will inherit the same *<Roles>* on *<Projects>* as these existing objects. This is achieved by the creation of the following additional objects in Keystone:

- *<IdentityProvider>*: Represents a trusted IdP that is allowed to authenticate federated users in this Keystone. Each *<IdentityProvider>* has an identifier within Keystone (e.g. *acmeidp, kent, umeidprovider…)*.

- *<Mapping>*: Represents a list of *<Rules> objects*.

- *<Rules>*: Represents how a specific federated user's identity attributes received from the IdP via Apache are mapped into a *<Group>* or *<User>* known to Keystone. A sample *<Rule>* might say "if the federated user has the *orgPersonType* attribute with a value of *Contractor* or *Guest*, then map it to the *<Group>* named *External"*.

- *<Protocol>*: Represents a specific authentication protocol (e.g. *saml2*) supported by an *<IdentityProvider>*. It also indicates a *<Mapping>* object that should be applied to all the users that are authenticated using this *<IdentityProvider>* and *<Protocol>* pair.

Figure 2 depicts the general federated authentication process in Keystone for services such as Nova, Neutron, Glance, etc. When a federated user desires to start an authentication process, it must access the Keystone URL associated with a particular *<IdentityProvider>* and *<Protocol>*, and protected by Apache (**step 1**). Apache must be properly configured to protect this URL using the correct federation protocol module (e.g. *mod_shib* [MODSHB]).

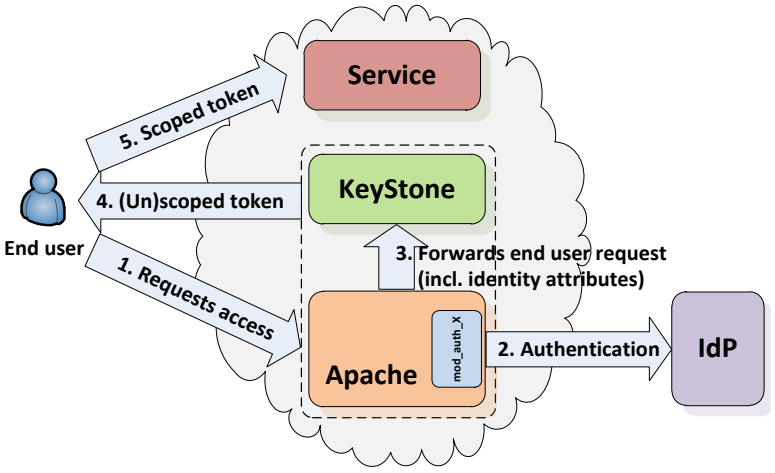

Figure 2. OpenStack federated authentication

This module will take care of the federated authentication process, interacting with the IdP (**step 2**) to obtain the authentication and identity information about the user. At the end of this process, the module establishes a set of environment variables that contain the identity information of the authenticated user. Apache forwards the initial user HTTP request to Keystone along with these variables (**step 3**). Keystone uses this information to execute the correct <Mapping> rules for the IdP and protocol combination. Once mapped into a <User> or <Group> the federated user is sent an unscoped token, he chooses his project, and is given a scoped token (**step 4**) for it. Finally, the user can present the scoped token to the OpenStack service, such as Swift, Nova, etc. (**step 5**).

It is worth noting that Horizon, the OpenStack web dashboard service, follows a slightly different flow. As it is web-based, the end user cannot directly present the scoped token as is done for the rest of the services. Instead, Horizon shows a selector for federated authentications where the end user can choose the <IdentityProvider> and <Protocol> to be used. Horizon then redirects the end user to the appropriate protected URL (following the same flow as described above), and the federated authentication is performed. At the end of the process, the end user is redirected back to Horizon, including the scoped token in the HTTP request using a POST method. Then, Horizon can consume the scoped token in the same way as any other service and grant access to the end user.

5 ABFAB and OpenStack integration

The objective of the integration is to allow federated users to access OpenStack services through any kind of user interface (command line, client desktop application, browser, etc.), using the same AAA infrastructure that is already used today for network access, but now for federated access to applications. Prior to the development of ABFAB, AAA infrastructures have not been used to provide application level access controls. Their re-utilization for this purpose will avoid the need to deploy parallel trust infrastructures for the network-layer and the application-layer. This is currently a burden for organizations, which need to ensure consistency between both infrastructures, and support the extra work of deploying and maintaining two different infrastructures for similar purposes.

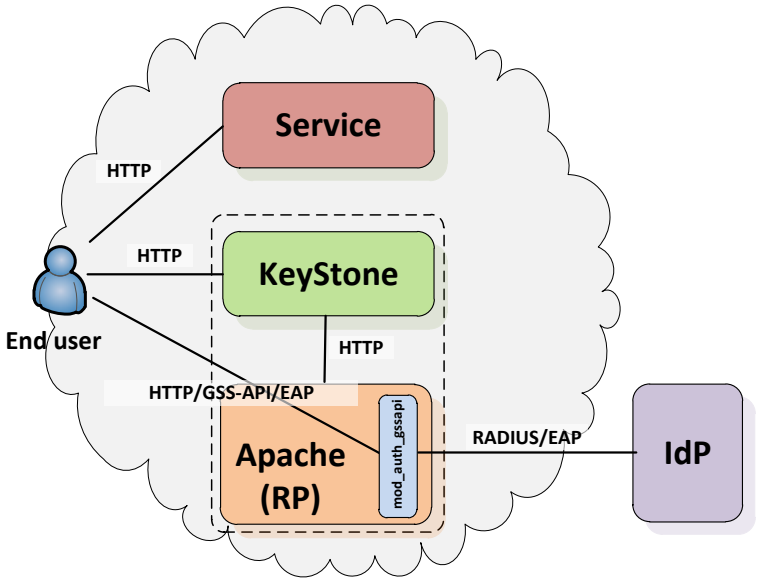

Figure 3. OpenStack and ABFAB integration

5.1 Architecture

The first step for understanding how we have integrated ABFAB into OpenStack is to describe how the ABFAB entities have been mapped into OpenStack components. In particular, the ABFAB Client can be directly mapped to the OpenStack user, whereas the ABFAB IdP has a correspondence with the OpenStack IdP component. Regarding the ABFAB RP, its functionality resides in the front end Apache server, since OpenStack's support for federation is based on it. However, unlike other Apache authentication modules, where the user is redirected to the IdP via HTTP, the ABFAB module uses the HTTP Negotiate [HTTPNG] protocol that specifies how to use GSS-API for web authentication. In this case, the RP acts as an intermediate entity, taking the EAP packets contained within the GSS-API tokens received from the user, and sending them to the IdP using a RADIUS like transport protocol. This process is depicted in Figure 3.

To perform the ABFAB Client functionality, things are more straightforward, as current versions of most web browsers, as well as many command line utilities (such as cURL [CURL]), already support the client-side (*Initiator)* functionality of the HTTP Negotiate protocol.

5.2 Authentication Workflow

To detail how Keystone can use an ABFAB-based authentication process using the proposed architecture, let us assume the user (running a Client) wants to access a particular OpenStack service. She first needs to get a scoped token from Keystone, after authenticating using an ABFAB IdP (e.g. HomeIdP). The process is depicted in Figure 4.

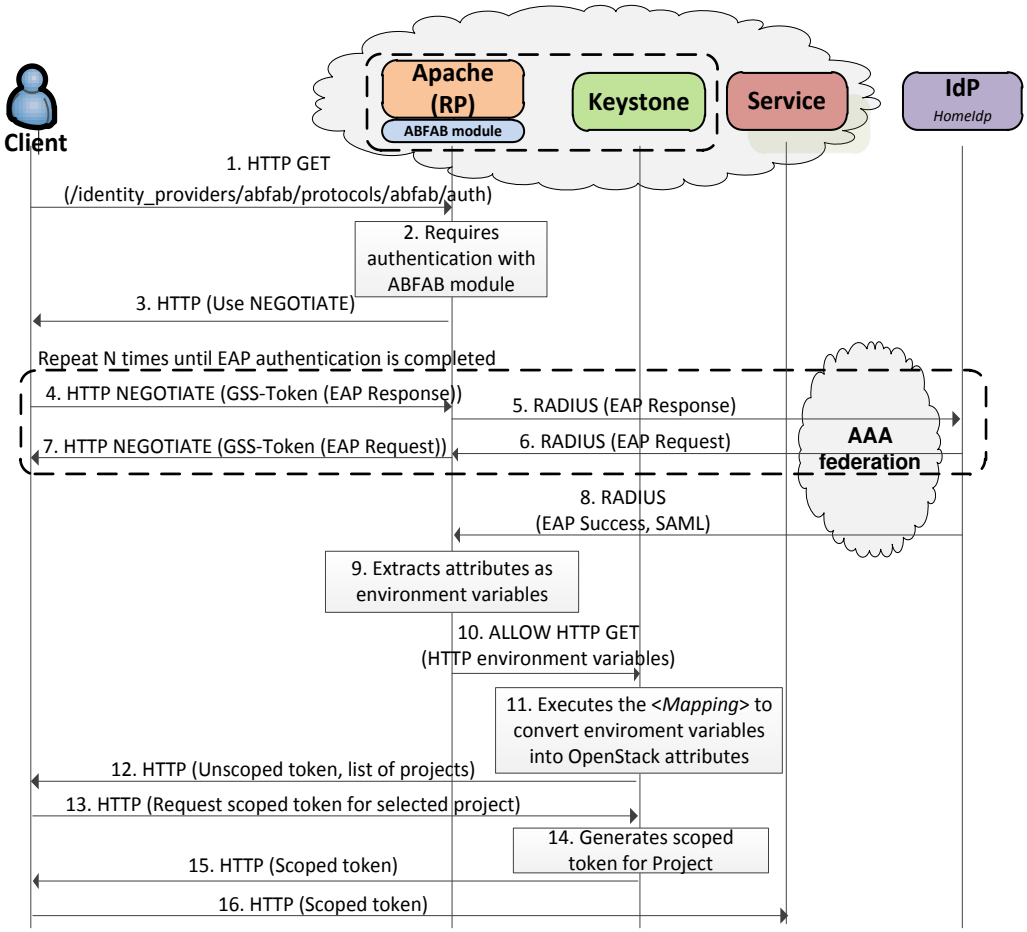

Figure 4. OpenStack/ABFAB workflow

Step 1. The authentication process with Keystone starts when the Client tries to access the protected URL on the Apache server. This URL has the following format:

/v3/OS-FEDERATION/identity_providers/{idp}/protocols/{protocol}/auth

where *{idp}* is the identifier of the trusted <*IdentityProvider*> ("***abfab***") that will authenticate the user, and *{protocol}* is the identifier of the federation <*Protocol*> that will be used ("*abfab*").

Hence, the actual URL would be the following:

/v3/OS-FEDERATION/identity_providers/abfab/protocols/abfab/auth

Here we encountered the first issue. Neither the Client nor Keystone know which IdP will be used by the user. It all depends upon the user's ID (formulated as *user@idpname*). Thus the protected URL must cater for all the IdPs in the AAA network. This necessitated a change in Keystone. Instead of assigning one identifier to a trusted IdP, Keystone was modified to accept a set of identifiers, where each identifier is the name of one actual IdP, and the

<*IdentityProvider*> in the URL now represents a trusted federation of IdPs. Consequently we used '*abfab*' to represent both the protocol and the trusted federation in our protected URL.

Step 2. The Apache server captures the request to this URL, but it will not allow it to reach Keystone until the Client has completed the authentication process. In this case, Apache has configured the ABFAB GSS-API module as the authentication module.

Step 3. The ABFAB module replies to the Client indicating that authentication using HTTP Negotiate is required to get access to the requested resource.

Step 4. Upon receiving this message, the Client calls its GSS-API module to get the first GSS-Token to be sent to the Apache server. Since the Client is using the GSS-EAP mechanism, this GSS-Token will transport an EAP response packet, which contains the user's anonymous identifier (e.g. *anonymous@homeidp*). It is worth noting that the anonymous identifier hides the end user's real identity, but it has to include the end user's real domain name ("*homeidp*") which will be used to identify the end user's IdP.

The HTTP Negotiate protocol includes an *Authorization* HTTP header with the value *"Negotiate [base64 encoded GSS-Token]".*

Step 5. The ABFAB module processes the received GSS-Token using the GSS-EAP mechanism, which in turn extracts the EAP packet and sends it to the IdP that is indicated in the anonymous identifier (i.e. @*homeidp*), by using the RADIUS infrastructure.

Step 6. The IdP receives and processes the EAP packet and creates a new EAP request for the Client, which is sent back to the ABFAB module using the RADIUS infrastructure.

Step 7. The RP extracts the EAP request packet and sends it to the Client using the HTTP Negotiate protocol. That is, a base64 encoded GSS-Token is included as the value of a *www-authenticate* HTTP header.

This process (steps 4-7) is repeated until the execution of the EAP method has been completed. Typical EAP methods used in ABFAB are EAP-TTLS or EAP-PEAP, both of which involve the establishment of a TLS tunnel between the Client and the IdP, so that the user's credentials are not visible to the ABFAB module. The user's actual identifier (say *alice@homidp*) is sent through an internal EAP method (e.g. EAP-MD5) protected by the TLS tunnel.

Step 8. Once the EAP authentication is complete, the IdP generates an *EAP Success* packet and generates a SAML assertion containing the user's identity attributes. This information is sent to the ABFAB module using the AAA infrastructure [SAML-AAA]. The *NameID* field in the SAML assertion can contain a pseudonym or transient ID to protect the user's identity over the RADIUS network. An example of a SAML assertion is shown in Figure 7.

Attribute Release Policies [ATRP] can be applied in the IdP in order to decide which attributes can or can not be revealed to RPs, depending on end user or IdP preferences, the application service being requested, etc. For doing this the end user might, for instance, use a

web portal to enumerate the trusted RPs. The lifetime of the assertion is also set up by the IdP according to its preferences, although it should not exceed the lifetime of the EAP keying material, since the user should be considered unauthenticated beyond that moment.

Step 9. The ABFAB module extracts the SAML and RADIUS attributes, using the GSS-API Naming Extensions [GSSNAM], and provides them to Apache, in the form of HTTP environment variables (e.g. IdP identifier, user entitlement, role…).

Step 10. As the user has now been successfully authenticated, Apache lets the initial HTTP request from the Client reach Keystone, along with the environment variables obtained from the ABFAB module.

Step 11. Keystone executes the <*Mapping*> rules that correspond to the <*IdentityProvider*> and <*Protocol*> indicated in the request URL (in this case, *abfab* and *abfab*). The execution of these mapping rules associates the federated user with an existing <*User*> or one or more <*Groups*>. For example, the <*Mapping*> might establish that if the federated user has the *eduPersonAffiliation* SAML attribute with a value of *student*, then she will be assigned to the <*Group*> called *students.* Separately the Keystone administrator will have assigned on or more <*Roles*> in <*Projects*> to the *students* <*Group*>. An example of a mapping rule is shown in Figure 6.

Step 12. Once Keystone knows which <*Roles*> and <*Projects*> are available to the federated user, it generates and delivers the unscoped token to the Client, along with the list of projects the user is able to access.

Step 13. This allows the user to select the desired project, and the Client to request a scoped token for it.

Step 14 and 15. Keystone generates and provides the Client with a scoped token.

Step 16. Finally, the Client uses this to access the desired Cloud service. Steps 12-16 are legacy OpenStack's messages where tokens are exchanged among the Client, Keystone and Service. Examples of these messages can be found here [TOKENS]

5.3 Security considerations

The solution proposed in this paper is grounded on well-known security protocols and technologies that have been proven to be reliable and secure over the years (e.g. EAP, GSS-API, RADIUS, HTTP Negotiate…). The only exception to this is the GSS-EAP mechanism, which is a relatively new standard, although it has been deeply analyzed during its standardization process in the IETF. Moreover, this mechanism makes extensive use of GSS and EAP channel bindings [CHBIND] as a means of assuring that the different security layers involved in the process share sufficient information to assure they are establishing security associations between the same set of entities (i.e. Client, RP, and IdP). In this paper we just provide a means of using this mechanism for OpenStack federated access control, therefore it does not introduce any new security protocols or vulnerabilities, as it is a standard use of the aforementioned technologies.

6 Implementation

This section describes the most relevant implementation details along with our deployed testbed, in which we have validated the prototype.

6.1 Software components

For our testbed we have made use of *Moonshot*, the ABFAB implementation developed under the Moonshot project [MSHOT], on its version 0.9.3. It provides *mech_eap,* an implementation of GSS-EAP. It also implements an Apache authentication module that performs the server-side (*Acceptor)* functionality of the HTTP Negotiate protocol. This module is based on the existing *mod_auth_krb* module, and is called *mod_auth_gssapi* [MODGSS]. Finally, Moonshot also provides Client side functionality for Windows OS and various flavours of Linux. This includes an *Identity Selector* software (as seen in section 6.5) used by the user to manage her identities and select the one that should be used to perform the authentication process.

The other component that is needed for the ABFAB AAA infrastructure is the IdP. It is based on FreeRadius 3 [FREERAD], and it consists of a RADIUS server that is able to authenticate end users, and has the required SAML components to issue the SAML assertions.

Regarding OpenStack, we chose the *Liberty* release, as using *Juno or Kilo* would have required some fixes to make them work, since they do not implement complete support for federation.

Another issue we encountered, was that the list of trusted <*IdentityProvider*> in Keystone is not publicly available to unauthenticated users. Our initial fix was to remove all access controls from the API call to GET them. Several discussions of how to fix this problem in the core release were held in the Keystone group, but no agreement on the best solution was reached. At the time of writing the OpenStack administrator has to configure this list separately in Horizon. Once the list of trusted IdPs is made available, we still needed to modify the Horizon login page so that it could include a way for the user to select his chosen <*IdentityProvider*> and <*Protocol*> to perform the federated authentication (see Figure 4 step 1). This is now part of the core Horizon release.

6.2 Testbed infrastructure for functional validation

With the software components described in section 6.1 we deployed a testbed for a functional validation, involving three different locations: University of Murcia, University of Kent, and GÉANT's QALab[1]. In this test scenario, a user from the University of Kent (*alice@cs.kent.ac.uk*), working in the QALab, wants to access the OpenStack Swift service deployed at QALab (*classe1.qalab.geant.net*), which is connected to the AAA infrastructure through a RP Proxy/IdP located at the University of Murcia.

[1] https://issues.geant.net/jira/browse/QATB

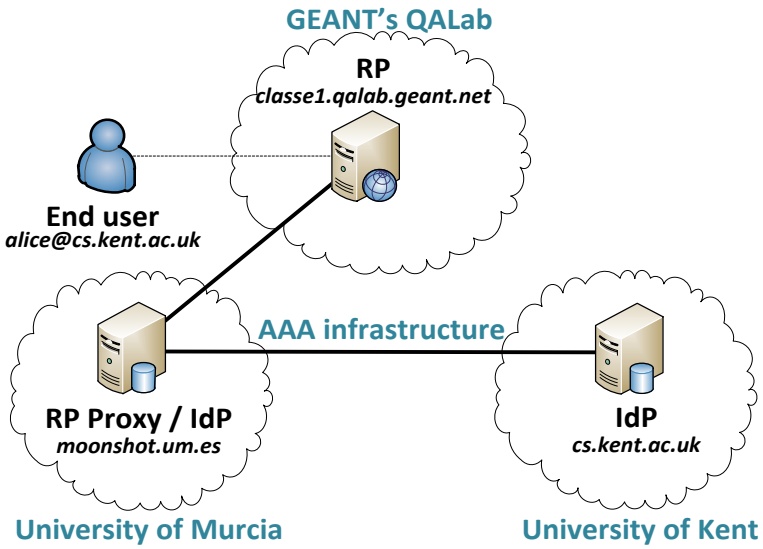

Figure 5. Functional testbed infrastructure

In the context of the GÉANT project, we could have made use of the currently deployed RADIUS infrastructure (used for the eduroam service), in order to test the federated OpenStack authentication. The reason for not having done so is that eduroam is a federated network access service defined by GÉANT and governed by a set of service policies [EDUPOL], agreed and signed by all the federation members. These policies implicitly disallow ABFAB data traffic, at it is not specific for the network access service. Nevertheless, for the purpose of our performance tests, this reduced infrastructure behaves in the same way as eduroam, except that it is without the network delays.

Using this testbed we validated our proposal from a functional point of view, demonstrating how AAA federated infrastructures can be used for access control in OpenStack thanks to the ABFAB technologies. It is worth mentioning that end user experience is identical, regardless of a) the location of the user on the AAA infrastructure, and b) the complexity of, and number of nodes in, the AAA infrastructure.

6.3 Testbed infrastructure for performance analysis and public demonstrator

We also deployed an additional simplified testbed (depicted in Figure 6) with a twofold objective: 1) to use it to analyze the performance of the proposal, and 2) to serve as a public demonstrator (see section 6.5). This testbed involves a single location: the University of Murcia. In this test scenario, a student from the University of Murcia (*alice@um.es*) wants to access an OpenStack Swift service deployed at the University of Murcia (*abfab-openstack.inf.um.es*), which is connected to the AAA infrastructure through an IdP located at the University of Murcia (*moonshot.inf.um.es*).

The reasons for building the simplified testbed were mainly based on the difficulties we found in granting access to a public demo in the QALab deployment using our actual IdP at Murcia, but it also made it easier to take the performance measurements.

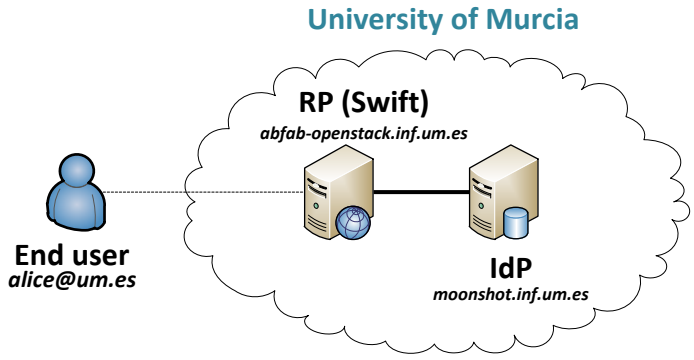

Figure 6. Testbed infrastructure

In the following we provide more details about the different components of the testbed.

6.3.1 RP

The **RP** (*abfab-openstack.inf.um.es*) consists of a dedicated virtual machine (VMWare ESXi 5.5), provisioned with an 4-core Intel Xeon CPU (2.53 GHz) and 2 GB of RAM, running Ubuntu 14.04 and OpenStack *Liberty*. It has a Swift service and an Apache server with the *mod_auth_gssapi* authentication module. Keystone has been configured with an <*IdentityProvider*> named *abfab* and a <*Protocol*> named *abfab.* Thus, Apache has been configured to use *mod_auth_gssapi* to protect the following URL:

```
/v3/OS-FEDERATION/identity_providers/abfab/protocols/abfab/auth
```

Keystone has been configured with a <*Mapping*> for the protected URL and its corresponding <*Rules*> that say that any user from any organization within the *abfab* federation with an *eduPersonAffiliation* attribute of *Faculty* will be assigned to the OpenStack <*Group*> called *faculty,* and any user from any organization within the *abfab* federation with an *eduPersonAffiliation* attribute of *Student* will be assigned to the OpenStack <*Group*> called *students.* Members of the *faculty* group have been granted access to the <*Project*> *privatefiles* whereas members of the *student* group have been granted access to the <*Project*> *publicfiles.* This <*Mapping*> is represented in its JSON format as follows:

```
[
   {
      "remote":[
        { "type": "eduPersonAffiliation", "any_one_of": ["Faculty"] }
      ],
      "local":[
         { "group":{"name":"Faculty"} }
      ]
   },
   {
      "remote":[
        { "type": "eduPersonAffiliation", "any_one_of": ["Student"] }
      ],
      "local":[
         { "group":{ "name":"Student" } }
      ]
   }
]
```

Figure 7. Mapping rules

The Moonshot RP (i.e. the Apache module) has been configured to deliver the entire RADIUS traffic through to the IDP located at the University of Murcia.

6.3.2 IdP

The IDP (*moonshot.inf.um.*es) consists of a dedicated virtual machine (VMWare ESXi 5.5), provisioned with an 4-core Intel Xeon CPU (2.53 GHz) and 2 GB of RAM, running Ubuntu 14.04 and FreeRadius 3.0.7. It is configured to use EAP TTLS/MD5 as the authentication method. It handles the *um.es* test RADIUS realm. It is configured to generate and distribute a SAML assertion with basic information about the user. The delivered SAML assertion is depicted in Figure 8. Note the use of a pseudonym for the *Subject's Name ID* element (first highlighted text), in order to protect her privacy. Note also that the value of the *eduPersonAffiliation* attribute for Alice is set to *Student* (second highlighted text), so she will be mapped to the OpenStack <Group> *students.*

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                ID="48217432493" IssueInstant="2015-03-19T08:30:00Z" Version="2.0">
  <saml:Conditions NotOnOrAfter="2015-03-19T08:30:00Z" />
  <saml:Issuer>um.es</saml:Issuer>
    <saml:Subject>
      <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
        2137423432412387981231@um.es
      </saml:NameID>
    </saml:Subject>
  <saml:AttributeStatement>
    <saml:Attribute Name="eduPersonAffiliation"
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
      <saml:AttributeValue>Student</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatatement>
</saml:Assertion>
```

Figure 8. SAML assertion

The Moonshot software comes with a very basic SAML template that only allows the NameID to be substituted at runtime with the value of the RADIUS User-Name attribute. This is insufficient for most practical purposes. We provided a temporary fix for our testing purposes, by hardcoding two different assertions into FreeRAdius. If the user was Alice, then the "student" assertion was sent. If the user was Carol, the "faculty" one was sent.

6.3.3 End User / Client

The client is deployed on a dedicated virtual machine (VBox 5.0.14), provisioned with a 4-core Intel I3 CPU (1.90GHz) and 2 GB of RAM, running Ubuntu 14.04. It has a Firefox browser with support for HTTP Negotiate enabled. It also has the Moonshot plugin installed, including the *Identity Selector* software. The user has the two identities *alice@um.es* and *carol@um.es* configured (see section 6.5). Finally, the client is connected to the network using a wireless interface, which makes the RTT (Round-trip time) to the RP significant (see the following section).

6.4 Performance analysis

Using this testbed, we carried out a performance analysis of the solution for the purpose of verifying that the required time to complete a federated authentication, ignoring the network delays of the AAA infrastructure, is within acceptable values and, thus, the proposed solution does not result in too long waits that might lead end users to give up before completing the authentication process.

For the analysis we performed 200 executions of an ABFAB-based federated keystone authentication, measuring the sample mean value and the confidence interval at 95% confidence (computed as *Sample Mean ± 2 * standard deviation)* of the following indicators, where *X* and *Y* represent the core entities (*Client, RP, or IDP*):

- COMP (X). The amount of time spent by entity X to perform the required functionality. This indicator specifically excludes network delays. We have calculated the value of this indicator as the sum of the time elapsed in component X between the reception of a message (either a request or a response) and the emission of the resulting message.

- WAIT (X). The amount of time spent by entity X waiting for the response of the request sent. We have calculated this value as the time elapsed in component X between the emission of a request message and the receipt of the corresponding response message.

- NETW (X, Y). The amount of time spent delivering request and response messages between entities X and Y. NETW(X, Y) = WAIT(X) - COMP(Y) - WAIT(Y). This is the general equation for when X talks to Y and Y talks to Z. If only X and Y communicate then WAIT(Y) is zero.

- TOTAL_TIME. The amount of time that is required by the Client to complete the access process. TOTAL_TIME = COMP(Client) + NETW(Client, RP) + COMP(RP) + NETW(RP, IDP) + COMP(IDP). This can also be measured directly as the time between the first message sent and the last message received.

In order to measure these indicators we made use of Wireshark [WSHARK]. With this software, we captured the network traffic on each one of these components, and used it to calculate the values of the indicators with the help of a set of scripts. The results we obtained from this analysis are provided in Table 1, whereas Figure 9 presents a stacked graph with these values.

Table 1. Performance results (in milliseconds)

	ABFAB federated authentication *Sample mean time ± 2 * (std. dev.) ms*
COMP(EU)	96.21 ± 01.88
NETW(EU, RP)	543.76 ± 10.30
COMP(RP)	453.35 ± 10.50
NETW(RP, IDP)	1.55 ± 00.05
COMP(IDP)	5.10 ± 00.32
TOTAL_TIME	**1099.97 ± 23.05**

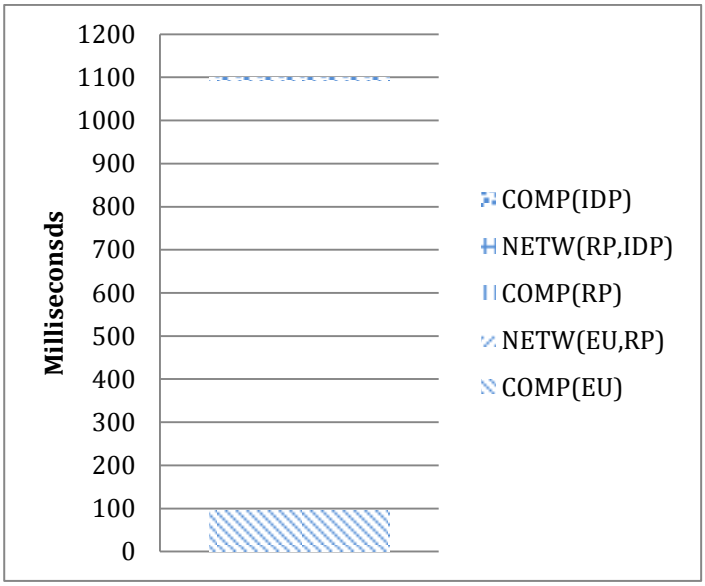

Figure 9. Performance results graph.

As can be observed, the average total time required to perform the federated authentication is ≈1099 ms., spending ≈554 ms. on computations (COMP(EU)+COMP(RP)+COMP(IDP)), and having ≈545 ms. attributable to network delays. We confirmed that this calculation is correct by measuring the time between the first message sent by the client and the last message received, and the results were broadly comparable. We should note that in this demo, the IDP and the RP are deployed on the same local network and thus, the network time between both elements is negligible (as shown in the value of *NETW(RP, IDP)*). However, in production environments, where the RP must contact a remote IDP, we can expect an average network time between both entities of ≈550 ms., as described in [GSSERP]. Hence, in production environments the total time would likely be increased up to ≈1650 ms. Given the expected

benefits of integrating OpenStack with ABFAB-based federations in terms of security, usability, and user management simplification, we consider this is a reasonable amount of time as it will hardly be noticed by end users. This is especially true if we consider that Keystone authentication is performed only at the moment of obtaining an unscoped token. After that, the token itself is used to obtain further scoped tokens to be used to access to the different services. Nonetheless, the techniques described in [GSSERP] also could be applicable. In such a case, both the network and computation times would be significantly reduced.

6.5 Public demo of the federated authentication process

The rest of this section describes, with snapshots of the process (Figure 10), how a real federated authentication process can be performed using this testbed.

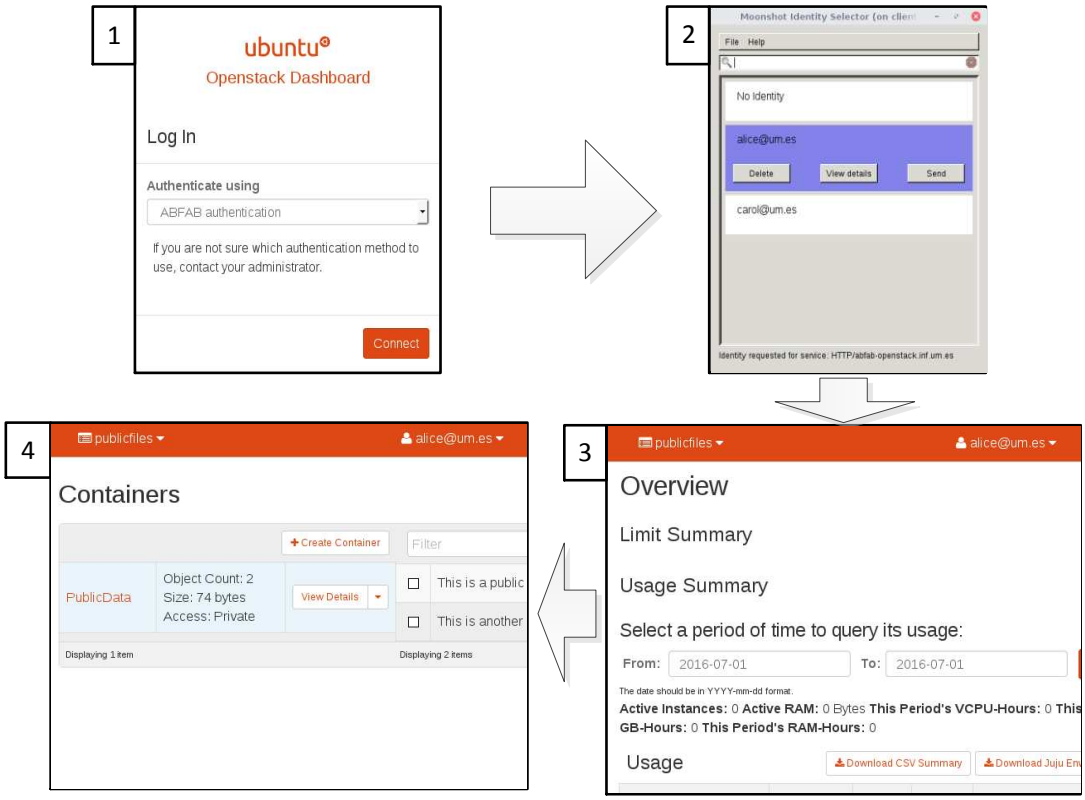

Figure 10. Federated access to Horizon

First, user Alice (not a Faculty member) uses her browser to connect to the Horizon login page (**step 1**) and selects the *ABFAB authentication* mechanism. At this point, the Moonshot Identity manager pops up, prompting Alice to select the identity she wants to use (**step 2**). She selects *alice@um.es* and starts the authentication process.

Authentication is performed between Alice's browser, Apache's *mod_auth_gssapi* on the RP, and the RADIUS IdP. When it finalizes, Keystone receives the SAML assertion attributes, executes the *<Mapping>* to determine the *<Groups>* that must be assigned to Alice, then determines the projects and roles assigned to these groups. As Alice is only a member of one

group (*students*) and this only has access to one project (*publicfiles*), this is sent to Horizon in the redirect. The Liberty release of OpenStack automatically chooses the default project for the user and returns this to Keystone. As can be observed in figure 9 (**step 3**), Alice is then granted access to the *publicfiles <Project>* by Keystone. Having access to the *publicfiles <Project>*, Alice can modify the *PublicData* container within the *Swift* component (**step 4**).

It should be noted that ABFAB provides zero sign on, because the next time Alice accesses OpenStack, her identity and password will be automatically selected and sent to the Murcia IdP without her having to enter anything.

On the other hand, the user *Carol* (carol@um.es) has an *eduPersonAffiliation* value of *Faculty*. Hence, if the end user switches the selected identity to *Carol,* she will be assigned to the *privatefiles <Project>*, and therefore has access to the *PrivateData* container.

The demo is publicly available at http://abfab-openstack.inf.um.es/, where there are further instructions to set up a suitable client, and to run all the possible tests.

6.6 Integrating other Cloud solutions with ABFAB

After having performed the work described in this paper, we have gathered enough knowledge to outline how a similar approach could be followed to integrate ABFAB support into other Cloud solutions. First, the Cloud solution must provide an identifiable URL that lets anyone be considered as authenticated. Second, that particular URL must be protected with Apache using the *mod_auth_gssapi* module. The module must be properly configured specifying the location of the ABFAB IdP. Finally, the Cloud solution must use the *REMOTE_USER* Apache environment variable set up by the *mod_auth_gssapi* module as the name of the federated end user. Optionally, the Cloud solution may also use the additional Apache environment variables conveying the information from the SAML Assertion, to perform an authorization process before granting access to the requested resources.

7 Conclusions and future work

Cloud technologies provide a convenient way to distribute the computing load in medium and large deployments, making them easily scalable and providing better resource utilization. Introducing federation support into them offers a new opportunity for extending their collaboration possibilities. However, most federation technologies focus on web-based services, while CLI-based ones (such as the Cloud) have been largely forgotten. To solve this gap, the IETF ABFAB WG has defined a set of technologies to integrate any kind of service with AAA-based federations, which are nowadays successfully deployed for providing access control to the network service (e.g. eduroam).

In this paper we have shown that a cloud service, such as OpenStack can be smoothly integrated in a federation built with an AAA infrastructure, by using ABFAB technologies. This is achieved by integrating the functionality of the ABFAB RP into Keystone's Apache front-end, as required by OpenStack's federation architecture. A detailed description of the authentication and authorization workflow has been provided demonstrating how information is exchanged between the Client, OpenStack and the AAA-based IdP.

Beyond the theoretical description of the process, we have deployed a real scenario where this integration has been successfully tested, making use of Moonshot, the ABFAB reference implementation. The details of this deployment, along with snapshots of an authentication process using the Horizon cloud service, can be used by other organizations as a guide to creating similar testbeds and evaluating this integration. Moreover, this experience has leaded us to conclude that a similar process could be used for other Cloud solutions that were interested on adding support for ABFAB-based authentication. Finally, the results of the performance analysis show the feasibility of this solution in terms of average authentication time, which in production environments would be below 2 seconds.

As future work, we envisage that increasing the range of supported client devices will be needed for wider acceptability. Currently, Moonshot is supported in GNU/Linux and Microsoft Windows environments, but is still unavailable for Apple MAC OSX and mobile devices (e.g. Android, Mac iOS, etc.).

Acknowledgments

This work was carried out under the CLASSe (Cloud ABFAB Federation Services in eduroam) project, which was partially supported by the GÉANT GN3Plus Opencall grant, reference number 605243. We also acknowledge the help of Kristy Siu for her development work.

References

[3GPP] Overview of 3GPP Release 6, Summary of all Release 6 Features, Version TSG #33, ETSI Mobile Competence Centre 2006

[ABFAB] Application Bridging for Federated Access Beyond web (abfab) IETF Working Group. http://datatracker.ietf.org/wg/abfab/charter/.

[APACHE] http://www.apache.org

[C3GPPIMS] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6814010&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6814010

[CHBIND] N. Williams. On the Use of Channel Bindings to Secure Channels. IETF RFC 5056, November 2007.

[CLASSE] Cloud-ABFAB Federation Services in eduroam. http://www.um.es/classe

[COMST] Alejandro Pérez-Méndez, Fernando Pereñiguez-García, Rafael Marín-López, Gabriel López-Millán and Josh Howlett. Identity Federations Beyond the Web: A survey. Communications Surveys & Tutorials, IEEE (Volume:16 , Issue: 4)

[CSA] https://cloudsecurityalliance.org/guidance/csaguide-dom12-v2.10.pdf

[DIAMETER] P. Calhoun and J. Loughney. Diameter Base Protocol. IETF RFC 6733, October

2012.

[EAP] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC3748, June 2004.

[EDUGAIN] http://services.geant.net/edugain/About_eduGAIN/Pages/Home.aspx

[EDUKERB] A. Pérez-Méndez, F. Pereñíguez-García, R. Marín-López, and G. López-Millán. A cross-layer SSO solution for federating access to kerberized services in the eduroam/DAMe network. Springer International Journal of Information Security, 11(6):365–388, 2012.

[EDUPOL] https://www.eduroam.org/?p=docs

[EDUROAM] https://www.eduroam.org/

[EAPTTLS] P. Funk, S. Blake-Wilson, EAP tunneled TLS authentication protocol, EAP-TTLS, in: IETF Internet Draft, July 2004. draft-ietf-pppext-eap-ttls-05.

[FEDKERB] R. Marín-López, F. Pereñíguez, G. López, and A. Pérez-Méndez. Providing EAP-based Kerberos pre-authentication and advanced authorization for network federations. Elsevier Computer Standard & Interfaces, 33(5):494-504, 2011.

[FEDOS] David W. Chadwick, Kristy Siu, Craig Lee, Yann Fouillat, Damien Germonville. "Adding Federated Identity Management to OpenStack". Journal of Grid Computing: ISSN: 1570-7873 (Print) 1572-9184 (Online). Volume 12, Issue 1 (2014), Page 3-27.

[FREERAD] http://freeradius.org/

[GEANT] http://www.geant.net/Pages/default.aspx

[GS2] S. Josefsson and N. Williams. Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family. IETF RFC 5801, July 2010.

[GSSAPI] J. Linn. Generic Security Service Application Program Interface Version 2. IETF RFC 2743, January 2000.

[GSSEAP] S. Hartman and J. Howlett. A GSS-API Mechanism for the Extensible Authentication Protocol. RFC7055. December 2013.

[GSSERP] Alejandro Pérez Méndez, Rafael Marín López, Gabriel López Millán, Providing efficient SSO to cloud service access in AAA-based identity federations, Future Generation Computer Systems, Volume 58, May 2016, Pages 13-28, ISSN 0167-739X, http://dx.doi.org/10.1016/j.future.2015.12.002.

[GSSNAM] S. Hartman and J. Howlett. Name Attributes for the GSS-API Extensible

Authentication Protocol (EAP) Mechanism. RFC 7056, December 2013.

[HTTPNG] K. Jaganathan and L. Zhu. SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows. IETF RFC 4559, June 2006.

[IMSCCA] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5982754&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5982754

[KERBEROS] C. Neuman, T. Yu, S. Hartman, K. Raeburn, The Kerberos network authentication service (V5), in: IETF RFC 4120, July 2005.

[MODGSS] GSSAPI Negotiate module for Apache. https://github.com/modauthgssapi/mod_auth_gssapi

[MODSHB] Internet2 - Shiboleth. http://shibboleth.internet2.edu.

[MSHOT] Howlett and S. Hartman. Project Moonshot. https://community.ja.net/groups/moonshot.

[NIST] http://www.nist.gov/

[OASIS] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=id-cloud

[OAUTH] E. Hammer-Lahav, D. Recordon, and D. Hardt. The OAuth 2.0 Authorization Protocol. IETF RFC 6749, October 2012.

[OIDC] N. Sakimura et al. “OpenID Connect Core 1.0 incorporating errata set 1”. 8 Nov 2104. Available from http://openid.net/specs/openid-connect-core-1_0.html

[OPENID] OpenID Web Site. Available from: http://openid.net/.

[OS-KRB] Step-by-Step: Kerberized Keystone. http://www.jamielennox.net/blog/2015/02/12/step-by-step-kerberized-keystone/

[OSTACK] OpenStack Open Source Cloud Computing Software. https://www.openstack.org

[PANA] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. Protocol for Carrying Authentication for Network Access (PANA). IETF RFC 5191, May 2008.

[PANAKERB] A. Pérez-Méndez, F. Pereñíguez-García, R. Marín-López, and G. López-Millán. Out-of-band federated authentication for Kerberos based on PANA. Elsevier Computer Communications, 36(14):1527 – 1538, 2013.

[PEAP] A. Palekar, D. Simon, G. Zorn, S. Josefsson, Protected EAP protocol (PEAP), in: IETF Internet Draft, March 2003. draft-josefsson-pppext-eap-tls-eap-06.

[RADIUS] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). IETF RFC 2865, June 2000.

[REST] Leonard Richardson and Sam Ruby. Restful Web Services. O'Reilly, first edition, 2007.

[RFC2903] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence. Generic AAA Architecture. IETF RFC 2903. August 2000.

[RFC5868] S. Sakane, K. Kamada, S. Zrelli, M. Ishiyama, Problem Statement on the Cross-Realm Operation of Kerberos. in: IETF RFC 5868, May 2010.

[SAML2] Assertions and protocol for the OASIS Security Assertion Markup Language (SAML) v2.0, March. 2005. OASIS standard.

[SAML-AAA] J. Howlett, S. Hartman, A. Perez-Mendez. A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for SAML. January 2016. IETF draft-ietf-abfab-aaa-saml-14.

[SASL] A. Melnikov and K. Zeilenga. Simple Authentication and Security Layer (SASL). RFC 4422, June 2006.

[TOKENS] http://docs.openstack.org/developer/keystone/api_curl_examples.html

[USEC] Assent (Moonshot) Use Cases promoted by JISC. https://www.jisc.ac.uk/assent#tab-4-3

[WSHARK] Wireshark. https://www.wireshark.org/

Biographies

Alejandro Pérez Méndez is a full time researcher within the Department of Information and Communications Engineering at the University of Murcia. He received B.E., M.E. and Ph.D. degrees in Computer Sciences from University of Murcia in 2005, 2010 and 2015, respectively. He has participated in several projects funded by the Spanish government and the European Union. His research interests are focused on network security, access control and identity management.

Rafael Marín López is a full time assistant lecturer in the Department Information and Communications Engineering at the University of Murcia (Spain). Additionally he is collaborating actively in IEEE 802.21a and IETF above all PANA and HOKEY Working Groups. His main research interests include network access authentication, key distribution and security in mobile networks.

Gabriel López Millán is a full time lecturer in the Department of Information and Communications Engineering of the University of Murcia. His research interests include network security, PKI, identity management, authentication and authorization. He received his MS and PhD in computer science from the University of Murcia.

David W. Chadwick is Professor of Information Systems Security at the University of Kent. His group is the creator of PERMIS (www.openpermis.org), an open source X.509 and SAML supported attribute based authorisation infrastructure. David has published widely, with over 150 research publications, which include books, book chapters, journal papers and refereed conferences. He specialises in distributed policy based systems, Public Key Infrastructures, Privilege Management Infrastructures, Trust Management, Federated Identity Management, Privacy Management and Autonomic Authorisation. He actively participates in standardisation activities, is the UK BSI representative to X.509 standards meetings, and is the co-author/editor of 2 Internet RFCs, 6OGF specifications, 1 ISO standard and many Internet Drafts. He has managed over 30 research projects during his career.

Ioram Sette is a PhD candidate in Computer Sciences at the Universidade Federal de Pernambuco - UFPE, researching Identity and Access Management in Cloud Computing (2012-2016). During the academic year 2014-15 he studied with David Chadwick at the University of Kent, and worked on the abfab implementation. He has an undergraduate degree (1998) and MSc (2002) in Computer Sciences conducted at UFPE. He has professional experience in system administration, network design, infrastructure management, and development of network services (BBS, Intranet and Internet), as co-founder of "NetPE BBS and Internet Provider". He also has

good knowledge of electronic transactions and credit cards, having worked for Itaú Unibanco/Hipercard. He is currently working on cloud computing and security projects as a specialist system engineer at the Advanced Studies and Research Center of Recife - CESAR.