Multiple Authentication in OpenStack

Nilesh Ghavate
Asst. Prof
Information Technology
Don Bosco Institute of Technology
Kurla, 400070
Email: nilesh.dbit@gmail.com

Kunal Dandekar
Information Technology
Don Bosco Institute of Technology
Kurla, 400070
Email: kunal.dandekar95@gmail.com

Apurva Deore
Information Technology
Don Bosco Institute of Technology
Kurla, 400070
Email: apurvadeore95@gmail.com

Yash Deorukhkar
Information Technology
Don Bosco Institute of Technology
Kurla, 400070
Email: deorukhkaryash@gmail.com

Tanaya Mulik
Information Technology
Don Bosco Institute of Technology
Kurla, 400070
Email: tanayamulik@gmail.com

***Abstract*—OpenStack is a cloud environment that controls large pools of compute, storage, and networking resources throughout a data center, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. There are various components in OpenStack and these components provide various services. Keystone is an OpenStack service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStacks Identity API. This project aims at adding a single sign on feature to access the resources provided by OpenStack cloud. This project explains the role of Keystone component in OpenStack that allows the user to authorize and authenticate himself to the cloud environment using his credentials. This project enables the end user(s) to login to a cloud environment using their existing Google credentials using single sign-on approach.**
OAuth is an open standard for authorization, commonly used as a way for Internet users to login to third party websites using their Google, Facebook, Microsoft, Twitter, One Network, etc. accounts without exposing their password. This project will be using OAuth 2.0 protocol. The project efficiently explains the working of OAuth 2.0 protocol and its implementation in Keystone. Finally the steps for including google authentication in keystone and the technologies required are presented.

Keywords*—*OpenStack, Cloud, Keystone, OAuth, Google

I. Introduction

The project aims at allowing end users to login to a cloud environment using their existing Google credentials.
The user sends an authentication request to the Keystone server. The Keystone server forwards this request to the external Google Authentication server which will perform the task of fetching identities and authenticating them, after which a token is passed to the Keystone server for authorization. The authentication of the user is done by using the OAuth 2.0 protocol that uses the OpenID Connect plugin. The project thus, adds an external authentication feature and also separates authentication and authorization procedure.

A. *Single Sign-on*

SSO (Single Sign On) occurs when a user logs in to one Client and is then signed in to other Clients automatically, regardless of the platform, technology, or domain the user is using.

Google's implementation of login for their products, such as Gmail, YouTube, Google Analytics, and so on, is an example of SSO. Any user that is logged in to one of Google's products are automatically logged in to their other products as well.

B. *OAuth 2.0*

OAuth[5] is an open standard for authorization, commonly used as a way for Internet users to login to third party websites using their Google, Facebook, Microsoft, Twitter, One Network, etc. accounts without exposing their password. This authorization standard uses the OpenID Connect authentication layer to get data from the external identity provider in the form of OIDC Claims.

C. *OpenStack components*

Keystone and Horizon are the major components for this project. Keystone is used for authentication and authorization purposes whereas horizon serves as the dashboard.

II. Existing mechanism

1. The user enters his credentials to the services client, along with the request he wishes to make. Currently the OpenStack open source code only provides command line clients for each of its services, although a web based administrative client called Horizon is also available.
2. The client sends an authentication request to the Keystone server using the users credentials.

3. If the credentials are correct, and the user specified the project he wishes to use, Keystone sends back a scoped token and a list of endpoints to the different services that are available to him/her. If the project was not specified, Keystone sends back an unscoped token. An unscoped token can only be used with Keystone to exchange it for a scoped token, by providing a project ID. The user needs a scoped token to access any of the OpenStack services.
4. The client chooses the endpoint of its service and sends the scoped token along with the users request to the chosen service.
5. Assuming this is an opaque token and not a PKI based token, the service sends the scoped token to the Keystone server asking for it to be validated. (If it is a PKI based token the service can validate it itself by using the public key of the Keystone server.)
6. The Keystone server checks if the token is valid. If it is, it sends back the user id, domain, project, and the roles the user has in the project. The user is then authenticated with the service.
7. The service now makes an authorisation decision based on either the role and the project (and domain) that the user has (i.e. RBAC), or user id (i.e. ACL). If the user is authorised, then the service processes the request and sends the response to the client, otherwise the users request is rejected.

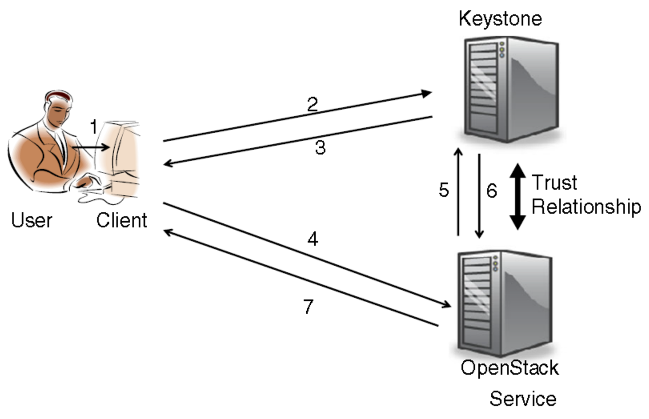

Fig. 1. Using Keystone to access an OpenStack service [1]

III. Work accomplished by project team

A. OAuth Mechanism

(A) The client requests authorization from the resource owner via the Authorization server.
(B) The client receives an authorization grant, which is a credential representing the resource owners authorization.
(C) The client requests an access token by authenticating with the authorization server and presenting an authorization grant.
(D) The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token and a refresh token.
(E) The client makes a protected resource request to the resource server by presenting the access token.
(F) The resource server validates the access token, and if valid, serves the request.
(G) Steps (E) and (F) repeat until the access token expires. If the client knows the access token expired, it skips to step (G); otherwise, it makes another protected resource request.
(H) Since the access token is invalid, the resource server returns an invalid token error.
(I) The client requests a new access token by authenticating with the authorization server and presenting the refresh token.
(J) The authorization server authenticates the client and validates the refresh token, and if valid, issues a new access token (and, optionally, a new refresh token).

```
+--------+                                           +---------------+
|        |--(A)------- Authorization Grant --------->|               |
|        |                                           |               |
|        |<-(B)----------- Access Token -------------|               |
|        |               & Refresh Token             |               |
|        |                                           |               |
|        |--(A)------ Authorization Request -------->|               |
|        |                                           |               |
|        |<-(B)------- Authorization Grant ----------|               |
|        |                                           |               |
|        |--(C)------- Authorization Grant --------->|               |
|        |                                           |               |
|        |<-(D)----------- Access Token -------------|               |
|        |               & Refresh Token             |               |
|        |                                           |               |
|        |                            +----------+   |               |
| Client |--(E)---- Access Token ---->|          |   | Authorization |
|        |                            |          |   |     Server    |
|        |<-(F)- Protected Resource --| Resource |   |               |
|        |                            |  Server  |   |               |
|        |--(G)---- Access Token ---->|          |   |               |
|        |                            |          |   |               |
|        |<-(H)- Invalid Token Error -|          |   |               |
|        |                            +----------+   |               |
|        |                                           |               |
|        |--(I)----------- Refresh Token ----------->|               |
|        |                                           |               |
|        |<-(J)----------- Access Token -------------|               |
+--------+                                           +---------------+
```

Fig. 2. OAuth authorization flow [4]

B. System's Working after addition of OAuth

(1) The client program or Horizon sends an authentication request to the Google server that includes user credentials of that particular server.
(2) The authorization server provides an access token to the client program.
NOTE: The mechanism of access tokens and refresh tokens is explained above in the working of OAuth section.
(3) This access token that the client program receives, is then sent to the Keystone server for authorization.
(4) Keystone then sends a scoped or unscoped token (concept explained in current scenario page).
(5) It is necessary for the user to get a scoped token, which then is passed on to the service provider (OpenStack).
(6) The service provider then provides the necessary services requested to the client program.

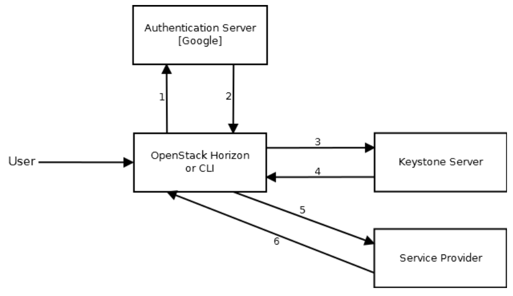

Fig. 3. Proposed system's working

IV. Erros encountered and measures taken to resolve them

(1) A domain name was required for the web server created. We had dbit-openstack.com as our domain name which points to the Keystone server.There was still an error of Site cannot be reached that was faced on clicking on the opened connect option. There was a solution of forwarding port 5000 since it was a non-standard port, that needed opening by adding a rule in the router configuration.

(2) After forwarding ports a **401 Authorization error Request requires authentication from** *ip address of the host machine (The computer from which browser requests are made)* was faced.

(3) Another error faced was in the /var/log/keystone/keystone wsgi public.log which says Missing entity ID from environment.We tried adding a [oidc] section in the keystone.conf file, yet the same 401 error persists.

(4) The whole setup was then installed on Amazon Web Service virtual instance to check for the issue of opening of ports, but the same server error was encountered again. Adding a security group rule for the particular port was tried, which did not work.

V. Conclusion

OpenStack is an open source software used to set up a private cloud in any organization or enterprise. Keystone is the primary component that we worked on. Our project is based on providing multiple ways for a user to authenticate themselves to use the cloud resources, and thus allowing Single Sign-On(SSO). This means that a user can sign in on OpenStack using their existing Google credentials. They will also be signed in on OpenStack if they are signed in on Google or any other Google service. This goes a long way in providing a satisfactory customer experience. Enabling Google sign-in in OpenStack will provide a much simpler and convenient authentication method without the need of additional login credentials.

Appendix A
Abbreviations

1. OpenStack - OpenStack is a cloud environment that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

2. SSO - Single Sign-On. It enables a user to log in with a single ID and password to gain access to a connected system or systems without using different usernames or passwords.

3. OAuth - OpenAuthorization protocol. It is used to authorize an external identity provider to authenticate users.

4. OIDC - OpenID Connect. It is an authentication layer above OAuth (authorization layer).

5. IdP - Identity provider.

Acknowledgment

This research was supported by Don Bosco Intitute Of Technology, Kurla. We are thankful to our institution who provided resources and expertise that greatly assisted the research. We thank our project guide Asst. Prof. Nilesh Ghavate for sharing his knowledge with us during the course of this research. We are also grateful to Asst. Prof. Tayyabali Sayyad who moderated this project and in that line improved it significantly. We would like to show our gratitude to Asst. Prof. Prasad Padalkar who constantly guided us throughout the semester. We are immensely grateful to them for their comments on any working related to our project. Any errors are our responsibility and should not tarnish the reputations of these esteemed professionals.

References

[1] Chadwick, David W., et al. ”Adding federated identity management to OpenStack”. Journal of Grid Computing 12.1 (2014): 3-27.

[2] Steve Martinelli, Henry Nash, Brad Topol. Identity, Authentication, and Access Management in OpenStack. O'Reilly Media. 2015. [Book].

[3] Introduction to OpenStack. [Online]. Available:https://www.openstack.org

[4] The OAuth 2.0 Authorization Framework Author: Dick Hardt Available: https://tools.ietf.org/html/rfc6749

[5] OAuth. From Wikipedia. [Online]. Available:https://en.wikipedia.org/wiki/OAuth