International Journal of Engineering Research and General Science Volume 3, Issue 5, September-October, 2015
ISSN 2091-2730

Implementing Dynamic Access Control in Openstack with Rbac using Token and Session Management

NITESH KUMAR

CH. Brahm Prakash Govt. Engineering College

Niteshks070@gmail.com

New delhi,India

Abstract— The emerging industry of Openstack brought us to an extent where security is of great importance .The model proposed in this paper implements dynamic access control in openstack using keystone ,this gives the finer granularity of authorization in openstack .The V model elaborates the flexible approach to attain the keystone identity service with the use of token and session management. The path followed by user to gain access to the system justify authenticity on every node .Therefore, providing a significant control mechanism.

Keywords—openstack ; Rbac ; token; session; V model ;session activation; token reallocation;

INTRODUCTION

As the cloud computing is new concept which enables the 2technologies such as web server ,network function virtualization and business models used to deliver IT capabilities(software ,platform ,hardware) as a service request, scalable and elastic[1].This is the new trend of computing where IT resources are dynamically scalable, virtualized and exposed as service on the internet[2].

This technology provide a well structured tool for the cloud management i.e Openstack, It is free and open source cloud software platform which can be used a infrastructure as a service[Iaas].All code is licensed under Apace 2 license. It is developed by NASA .

The main characteristics of openstack are as follows:

(i). Scalable: This solution is already developed worldwide in companies whose data volumes is measured in petabytes of distributed architecture and massively scalable upto 1 million physical machines, upto 60 million virtual machines and billion of stored objects[3].

(ii) Flexible: Openstack supports most virtualization solution of the market ESX. HYPER-V,KVM ,LXC,QEMU,UML,Xen andXenserver[2,4].

(iii). Opensource: It is the opensource technology so that it can be easily modified and changed but after going through a validation process. Many companies came forward to support the project and based on the code used by the NASA and Rackspace Cloud . It is written in python and currently implements two control APIs, the EC2 API and Rackspace.It uses different drivers to interface with a maximum no of hypervisors (Xen,KVM,HYPER-V,QEMU)[5].
Openstack components are as follows:

(1). Compute (Nova)

OpenStack Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC)configurations. KVM, VMware, and Xen area available choices for hypervisor technology, together with Hyper-V and Linux container technology such as LXC.[6][7]
It is written in Python and uses many external libraries such as Eventlet (for concurrent programming), Kombu (for AMQP communication), and SQLAlchemy (for database access).[8] Compute's architecture is designed to scale horizontally on

standard hardware with no proprietary hardware or software requirements and provide the ability to integrate with legacy systems and third-party technologies.

(2) Image Service (Glance)

OpenStack Image Service (Glance) provides discovery, registration, and delivery services for disk and server images. Stored images can be used as a template. It can also be used to store and catalog an unlimited number of backups. The Image Service can store disk and server images in a variety of back-ends, including OpenStack Object Storage. OpenStack's image is an operating system installed on a virtual machine (VM).[14]
Glance—OpenStack's image service module—is a compute module, as it does not store images, variations, or instances—but rather catalogs them and holds their metadata from Swift or a storage backend datastore.[14]

(3) Object Storage (Swift)

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data center,with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used. SwiftStack, an object storage software company, is currently the leading developer for Swift.[14]

(4) Dashboard (Horizon)

OpenStack Dashboard (Horizon) provides administrators and users a graphical interface to access, provision, and automate cloud-based resources. The design accommodates third party products and services, such as billing, monitoring, and additional management tools. The dashboard is also brandable for service providers and other commercial vendors who want to make use of it. The dashboard is one of several ways users can interact with OpenStack resources. Developers can automate access or build tools to manage resources using the native OpenStack API or the EC2 compatibility API.[14]

(5) Identity Service (Keystone)

OpenStack Identity (Keystone) provides a central directory of users mapped to the OpenStack services they can access. It acts as a common authentication system across the cloud operating system and can integrate with existing backend directory services like LDAP. It supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style (i.e. Amazon Web Services) logins. Additionally, the catalog provides a queryable list of all of the services deployed in an OpenStack cloud in a single registry. Users and third-party tools can programmatically determine which resources they can access.[14]

(6) ACCESS CONTROL

Access is the ability to enter into a computer resource. An access control system enables an authority to control access to areas and resources in a given physical quantity or computer based system. With the rapid growth of information technologies, it is obviously convenient and efficient to provide good security services. Access control is a term for security practice that is supported by security systems and provides a way for security management [9] delivered via the Internet. Security issues are the major issue in the enterprise and e-management

Role Based Access Control (RBAC):

RBAC is considered a much more generalized model in compare to both the MAC & DAC, encompassing both the models as special cases providing a policy neutral framework which allows RBAC to be customized on a per-application basis. As the blend of the MAC & DAC models and integrity, RBAC is partially founded on principles showcased by Biba's .
RBAC dynamically assigns the roles to the users based on criteria defined by the manager or system administrator. Role-Based Access Control models are a set of fairly new models [10] first introduced in the ninety's. The RBAC92 model [11] introduces the concept of roles, and RBAC96 [12] refines RBAC92 thanks to the addition of the users notion (different from the subjects one) and a roles hierarchy defined as a partial order. RBAC also stands apart from the more traditional MAC and DAC by granted rights on transactions, not on underlying subjects. These rights are granted to roles, which at first glance appear to be a synonym for DAC groups. The difference lies in that groups consist of a collection of users while roles are a bridge between a collection of users and a collection of the Clark-Wilson model of transaction rights. While RBAC supports data abstraction through transactions, it cannot be

used to ensure permissions on sequences of operations need to be controlled .To do this, a less general and more sophisticated access control model must be used.

(B). Limitations in existing System:

(i) No centralized control: The OpenStack platform is divided into different services, like Nova for computing, Swift for storage, etc. Each service uses a configuration file with the corresponding security policy. For SDN controllers, there exists another control system. For the OPNFV platform, it lacks a synchronization mechanism between these configuration and control systems in order to build a consistent and End-to-End protection system[13].

(ii) No dynamic control: Currently, the authentication and authorization in OpenStack are achieved through the token mechanism, but there isn't any token revocation mechanism. Once a user gets an authorization token, we will not have any control over the user. It lacks dynamic control at runtime in OpenStack and/or SDN controllers[13].

(iii) No customization and flexibility: Each user of OpenStack and/or SDN controllers consumes their resource pool in their own manner, but it lacks customization for security management system integration. For example, in both OpenStack and OpenDaylight, user cannot define their own security policy for each VNF resource pool[13].

(iv) No fine-granularity: Finally, the granularity of authorization in OpenStack and/or SDN controllers is not enough fine. Currently, it's at the API-level. This means that we can authorize or deny a user from using an API like launch VM in OpenStack. But we need the granularity to be pushed to the resource-level, authorize or deny a user from using a specific resource through the API, e.g. allowing a user to launch a dedicated VNF VM. [13]

***(*C). METHODOLOGY PROPOSED**

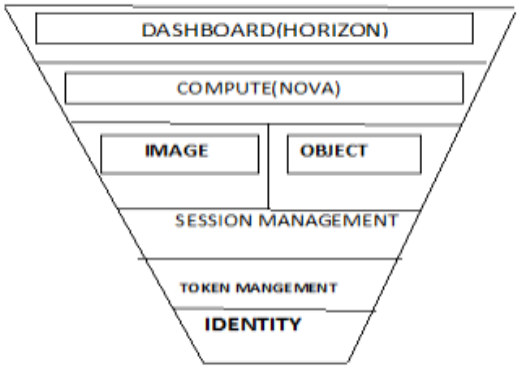

Fig.(i) V model

As per the given model V .this is more enhanced model for the openstack security system in which identity plays a major role in authentication.

The working of the model is as follows :-

Dashboard which is the major controlling center is at the top level and have the full view of the V components ,suppose a user wants to login then it enters through the UI of the dashboard and with the compute controller it register its image in the object storage and then it is logged in the session management and now it is binded with the time so that it verifies itself within the time allocated ,getting inside the token management, it mention its token and next in the identity it mention its role and access type and now the upward procedure follow again from the bottom i.e it gets its tokenid and then session Id means for which the token is valid,and then it is registered in the image service so that it have its backup and then get back to the compute. In this given model we actually overcome one of the flaws in the earlier system that is no dynamic control over the user after getting token from the keystone ,this is now

overcome by session management that is after getting the token from the keystone ,session will give a limited time to the user and in the image part we can apply token renewal system or in the session so that it can get again the session back.

(D).WORKING.

The working of the model is as follows:

According to the working diagram fig.(ii), When a user wants to login for the access it will interact with the UI based dashboard which give the Api based platform. Then user proceeds to the compute which is the controller part of the system, it guides through the all components of system. At the next step the login details ,time of activity and other details will be stored in the image which is for the backup data .Now it comes to the session management ,which will generate a session id and stored it in image ,now user is under the session i.e it have a limited time to enter its details ,in the next part it requests for the token according to its role and identity, keystone will authenticate it using its database ,now user will follow the reverse path that is gain a token from the token management and session id which contains the details about the session ,then registering through image and return to the dashboard with the access token and a session id .The administrator will have a look over to each part of system from the backend.

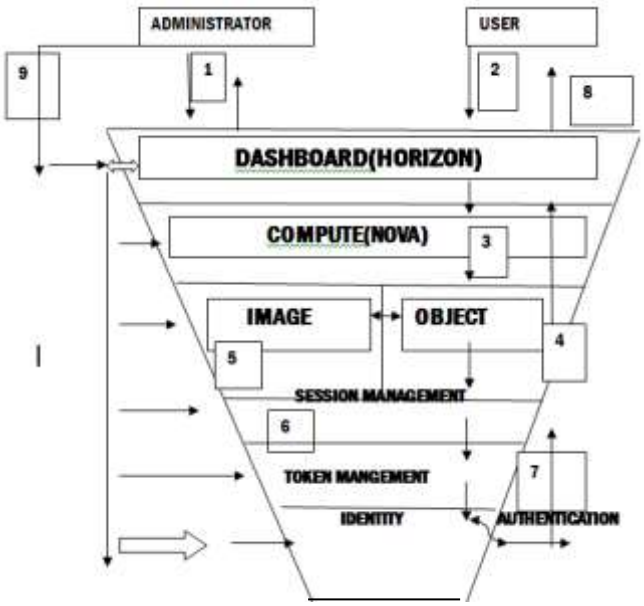

Fig.(ii) working of V model.

Working table of the diagram fig(ii).

S. NO. OPERATION

1. → admin operating at front end through user interface.
2 → user wants to gain access.
3 → manage the resources and user activity in the system.
4 → storing the temporary objects and files in multiple disk drives.
5 → containing backups about login credentials.
6 → provide session to the user before it takes the token and after taking the token.
7 → Manage the token granting ,token assessment, token reallocation and renewal.
8 → user comes out after authentication.
9 → Admin at the backend ,operates through dashboard , managing all components.

CONCLUSION

Finally ,customization of the dynamic access control has been made. Session and token management will keep track of user by token allocation, renewal and session activation[15]. Therefore the flaw that, no dynamic access control over the user ,has been fulfilled .Moreover the V model will provide finer granularity due to the deep authentication process which was at the api level earlier.

REFERENCES:

[1] Vaquero LM, Rodero-Merino L, Morn D (2011) "Locking the sky: a survey on IaaS cloud security. In Journal of computing Springer Verlag 91(1):93,1/2011 1-2.
[2] Keith J., Burkhard N.(2010).The future of cloud Computing.(expert group report).
[3] Ken pepple july 2011. Deploying OpenStack. OReilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol,CA 95472
[4] Takako,P., Estcio,G., Kelner,J., Sadok,D. (2010) . ASurveyonOpensourceCloudComputingSolutions.WCGA-8thWorkshop on Clouds, Grids and Applications.Gramado:28May, 3-16. Teyssier,S. (2010).
[5] Openstack Open source software for building private and public clouds.
[6] *"OpenStack Compute: An Overview"* (PDF). openstack.org. 2010.
[7] *"HypervisorSupportMatrix"*
[8] *OpenStack — more than just software"*.
[9] A. Stoughton, " Access flow: A protection model which integrates access control and information Flow". Proc. of the IEEE Symposium on Security and Privacy, pages 9–18, Oakland,CA, 1981
[10] L. Habib , M. Jaume and C. Morisset , " A formal comparison of the Bell &LaPadula and RBAC models",HTTS Funded by the Macao Science and Tech. Development Fund,2008
[11] D. Ferraiolo and R. Kuhn ," Role Based Access Control ", In Proc of 15th NIST- National computer security conference , Pages 554-563,Baltimore, October 1992
[12] R.S. Sandhu ,E.J. Coyne ,H.L. Feinstein ,C.E. Youman ," Role Based access control models ",IEEE Computer , Volume 29 Issue-2,1996
[13] wiki.opnfv.org/moon.
[14] wikipedia.org/wiki/OpenStack.
[15] Shashank , Nitesh kumar ,"token and session compatibility in role based access control with priveleges management" ijert.org. vol 3 issue 11 nov 2014.