Australian Journal of Basic and Applied Sciences, 9(11) May 2015, Pages: 795-799

ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Preserving Data Security in Cloud Storage

N. Jayaprakash and D. Dhanasekaran

Department Of CSE, Saveetha University, Saveetha Nagar, Thandalam, Chennai-602105, India

ARTICLE INFO

Article history:
Received 25 March 2015
Accepted 28 May 2015
Available online 14 June 2015

Keywords:
Cloud Computing, keystone, Swift, Python, Security.

ABSTRACT

To provide a secure data access and data storage in the cloud service, Data hosted in the cloud requires better security. In the present service of the cloud service providers, end users have to rely on the cloud service providers for the security of their stored data. Usually data in the cloud is encrypted, but there is a possibility of compromising the cloud or even the cloud service providers can avail the data stored in the cloud. In order to avoid entirely dependence on the cloud service providers for security, end users have to find a technique to secure their data on their own. An additional security is implemented along with the security provided by the cloud service providers. The data that is being uploaded in the cloud, it is encrypted at the user side, so that even someone compromise the cloud cannot access the data. At least it can protect the privacy of the data as well as raw data can be accessed only on the user end.

© 2015 AENSI Publisher All rights reserved.

To Cite This Article: N. Jayaprakash and D. Dhanasekaran, Preserving Data Security in Cloud Storage. ***Aust. J. Basic & Appl. Sci.,*** *9(11): 795-799, 2015*

INTRODUCTION

Cloud is a place where everyone places their data equally despite of its importance. Every user have their own credentials believing that would protect their data in the cloud. But the thing is cloud security might be breakable. Big companies can lend their money for building their own cloud. But the small companies and end user have no choice to have their own cloud due its cost and maintenance. Hence they are lending the services from the Cloud vendors.

As the security of the data is highly considered, the users has to locate a way to get rid of security difficulties. As of now, the data security is entirely depending on the cloud service providers (in most cases). Security is still a loophole in cloud services that faces enormous attacks over the year. To overcome it, new technologies should come in place.

Review of the related work:

Constructing cloud becomes easy even for the end users as some of the architecture is open source. And one of them is Rackspace Open Stack cloud service, using their packages one construct their own cloud easily. In this paper, there are mainly two services are used to construct the cloud. They are as follows:

1. Keystone service
2. Swift Object Storage Service

A. Keystone:

Once Keystone is installed in the machine, other services requires to be shaped to correlate with it. Usually keystone provides the database to store the username and respective passwords. And also, it is the service that authenticates the user who requiring the cloud service (in here-Swift Storage Service).

Generally, user provides his credentials to acquire the cloud service. Before reaching the cloud service it should have to be authenticated by the keystone service. During authentication, authentication token of the user is compared with the values in the database. It will also fetch other information from the token such as user identity, user name, project details, project id, and roles and so on.

a) Setting up Credentials:

Before installing keystone, rabbit server and NTP server is installed to keep the synchronization between services. All the configuration details to install keystone is stored in the file "Keystone.Conf".

For installing the keystone, one should provide the below details.

 # Public IP:

```
IP=$(cat details.txt | grep -i 'IP' | cut -d'=' -f2)
```

 # Default password of mysql-server:

```
MYSQL_PASS=$(cat details.txt | grep -i 'MYSQL_PASS' | cut -d'=' -f2)
```

Corresponding Author: N. Jayaprakash, M.E, Department of CSE, Saveetha University, Saveetha Nagar, Thandalam, Chennai-602105, India
E-mail: javidprakash@gmail.com

#Cloud admin of Open stack:

CLOUD_ADMIN=$(cat details.txt | grep -i 'CLOUD_ADMIN=' | cut -d'=' -f2)

Password will use to login into Dashboard CLOUD_ADMIN_PASS=$(cat details.txt | grep -i 'CLOUD_ADMIN_PASS' | cut -d'=' -f2)

 # Name of tenant (project):

TENANT=$(cat details.txt | grep -i TENANT | cut -d'=' -f2)

 # Service tenant:

SERVICE_TENANT=$(cat details.txt | grep -i 'SERVICE_TENANT' | cut -d'=' -f2)

And should configure the Keystone.Conf file and have to install the keystone service using terminal. Once installed keystone, service should be restarted. After installing the keystone, endpoints should have to be created and configured (details such as Service Endpoint, Public URL, Admin URL, Internal URL). In this paper, MySQL database is used to store the keys rather than keystone database, choosing database is up to the user concern.

b) Setting up projects, users, and roles:

Define a project, user, and role to link the project and user as the most basic set of details to get other services authenticating and authorizing with Keystone. And also should create the service, users and roles.

 # Define Admin, Member Role and Admin Tenant:

TENANT_ID=$(Keystone Tenant-Create --Name=$TENANT --Description="Admin Tenant"| Grep Id | Awk '{Print $4}')

Echo "TENANT_ID=$TENANT_ID" >> /Tmp/Service-Id-List

ADMIN_ROLE=$(Keystone Role-Create --Name=Admin | Grep Id | Awk '{Print $4}')

Echo "ADMIN_ROLE=$ADMIN_ROLE" >> /Tmp/Service-Id-List

MEMBER_ROLE=$(Keystone Role-Create --Name=Member | Grep Id | Awk '{Print $4}')

Echo "MEMBER_ROLE=$MEMBER_ROLE" >> /Tmp/Service-Id-List

 # Create User Admin:

ADMIN_USER=$(Keystone User-Create --Name=$CLOUD_ADMIN --Pass=$CLOUD_ADMIN_PASS --Email=Root@Localhost --Enabled=True | Grep Id | Awk '{Print $4}')

Echo "ADMIN_USER=$ADMIN_USER" >> /Tmp/Service-Id-List

 # Grant Admin role to the admin user in the admin tenant:

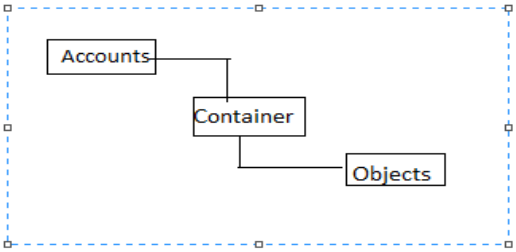
Keystone user-role-add --user=$CLOUD_ADMIN --tenant=$TENANT --role=admin

Once Keystone service is created, Swift storage should have to be installed.

B. Swift Storage Service:

Open Stack Object Storage (Swift) service provides redundant, scalable object storage service using the replicas of data stored in the different containers.

In order to access an object one should have to access three contents. Object is the data that is uploaded by the user. Accounts and container represents the user and location of the file stored in the cloud. Swift service consists of following components such as account server,



Container server, Proxy server and Object Server which is used for handling requests and storing and retrieving files from the storage service.

Rsyncd.conf holds the configuration of the container, accounts and objects.

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = $IP
[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock

[Container]
Max connections = 2
Path = /srv/node/
Read only = false
Lock file = /var/lock/container.lock

[Object]
Max connections = 2
Path = /srv/node/
Read only = false
```

Lock file = /var/lock/object. Lock

Account server has the list of containers for a specific user. Container service holds the details of the objects that stored in the container. Object server that just responds with the data stored.

a) Rings:

The data stored in the storage is distributed over the container and it is important to track the information where they are located. Swift service does this task using a data structure called "Rings". This stores the metadata information about the objects.

To configure the ring:

Swift-ring-builder account.builder create 18 3 1
Swift-ring-builder container.builder create 18 3 1
Swift-ring-builder object.builder create 18 3 1
Swift-ring-builder account.builder add r1z1-10.196.9.139:6002/sdb1 100
Swift-ring-builder container.builder add r1z1-10.196.9.139:6001/sdb1 100
Swift-ring-builder object.builder add r1z1-10.196.9.139:6000/sdb1 100
Swift-ring-builder account.builder
Swift-ring-builder container.builder
Swift-ring-builder object.builder
Swift-ring-builder account.builder rebalance
Swift-ring-builder container.builder rebalance
Swift-ring-builder object.builder rebalance

Separate rings are used for account databases, container databases, and individual objects and every ring does the same work.

b) Proxy Server:

Apache server is configured as proxy server to react. Every proxy server handles the request that is sent by any user, which redirects the request to exact ring or container, so that swift service will provide the data. Once it has the location of the request, it will serve the requested user. Objects always have to pass through the proxy server.

Proxy server details are stored in the Proxy-Server.conf file so that the server is configured as per requirement.

bind_port = 8080
User = swift
[Pipeline: main]

Pipeline= catch_errors gatekeeper healthcheck proxy-logging cache authtoken keystoneauth proxy-logging proxy-server
[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

c) Object server:

Object server can support multimedia files to be store, retrieve, and delete objects. It is always stored in the binary format with metadata of the file is stored in the same filename with extension 'Xattrs'. Every Object is found using the hashed name of the object and the timestamp when it's written in the cloud. As we synchronized the service, it will exactly retrieve or delete the exact data.

d) Container server:

Container service holds the details of the objects that stored in the container. Container server doesn't hold information about the location where the object is stored. It just holds the list of the data that is stored in the respective containers.

e) Python:

Python is the language that used to deploy the cloud services as well as for encryption in the above scenario. Pycrypto is a package that deals with encryption that is to be imported while uploading the file. Using Pycrypto files are encrypted using AES encryption. The keys are stored in the local system (tenant system) that will be used for encryption of files to be stored.

Problem statement:

Rackspace Open Stack is used for providing cloud storage service. By configuring cloud using OpenStack, cloud data storage service is deployed in Linux Operating System. Data is hosted in the cloud by an owner. It is then serviced to users. Every user can access only after authentication by the keystone service. It will redirect to swift storage service. Data stored is encrypted for increasing the security. Usually Open Stack mirrors the data in different containers, so that it will help in data integrity. User has to provide their login so that it is verified by the authentication service and if the credentials are passed. The request will be redirected to the cloud service.

And keystone will manage another important thing, it will redirect the user to the apt container where user is a member. Further the webpage will load the files that are stored in the container repository. User can download the files from the container or Upload the files into the container.

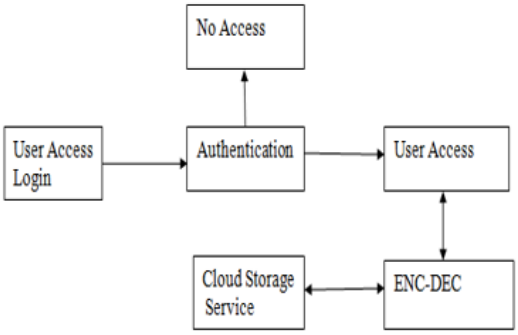


Fig. 1: Workflow Diagram.

Ensuring data security:

The cloud is constructed using the Open stack service provided by the Rackspace. And the authentication and authorization service is carried out the by the keystone as like usual cloud service providers. Using Swift service, Different group are created on the cloud as tenant and each group has a

set of users (like an organization). Using Keystone services, every user including Owner is given password to access respected files/Group. Files/Folder is created and stored by the Owner of the group. Each user can access the cloud by providing his/her password and it is validated in the Keystone API. If the credentials proven right, the user is given access to specific group. Keystone API filters users accurately so that the user cannot cross interact with other group. This ensures the data integrity maintained. Once user got access to the service, user can have rights to access the content. User can upload or download files from the cloud. But in order to enhance security the files that are being uploaded are secured with encryption.

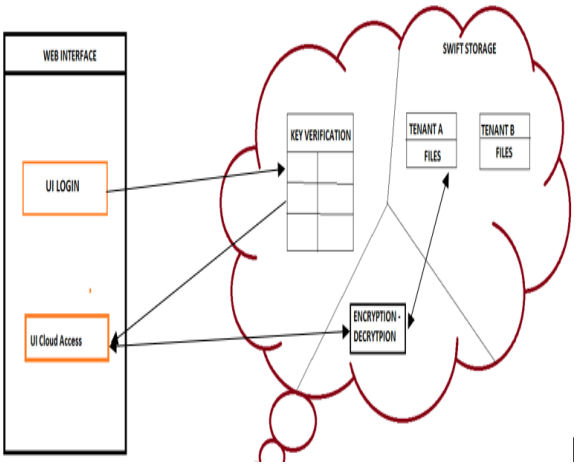


Fig. 2: Functional architecture Diagram.

But user unaware of the encryption only the group owner (tenant) know of information. Similarly while downloading files, the files are decrypted and provided to the user.

Conclusion:

Cloud service providers can introduce new way to enhance the security. But there always a possibility to compromise the security. Once cloud server is compromised by unknown sources, it may affect data integrity maintained. So users should come up with different techniques in order to secure the data from the hackers as well as from the cloud service providers.

REFERENCES

A New Approach to Pseudorandom Number Generation Ankur, 2014. Divyanjali; Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on DOI: 10.1109/ACCT.2014.26 Publication Year, pp: 290–295.

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, 2010. "A View of Cloud Computing," Communications of the ACM, 53(4): 50–58.

Ateniese, G., R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, 2007. "Provable Data Possession at Untrusted Stores," in the Proceedings of ACM CCS, pp: 598–610.

Certificateless Public Auditing for Data Integrity in the Cloud, Boyang Wang, Baochun Li, Hui Li and Fenghua Li "http://iqua.ece.toronto.edu/~bli/papers/boy ang - cns13.pdf"

Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds, Y.Zhu, H.Wang, Z.Hu, Gail-Joon, H.Hu and S.Yau "http://people.cs.clemson .edu/~hongxih/ papers/ acmsac2011.pdf"

http://github.com/openstack-dev/devstack" for open stack installation

http://mirantis.com/Cloud/Understanding%20OpenStack%20Authentication%20%20Keystone%20PKI%20%20Pure%20Play%20OpenStack.htm "for easy working with open stack.

Oruta, privacy-preserving public auditing for shared data in the cloud by Boyang Wang; State Key Lab. of Integrated Service Networks, Xidian Univ., Xi'an, China; Baochun Li; Hui Li. In Cloud Computing, IEEE Transactions on Jan.-March

Public Data Integrity Verification for Secure Cloud Storage, Hongwei Liu, Peng Zhang, Jun Li http://ojs.academy publisher.com/index.php/jnw/article/view/jnw08023 73380".

Secure Audit Service by Using TPA for Data Integrity in Cloud System, Shingare Vidya Marshal "http:// www.ijitee.org/attachments/File/v3i4/D1180093413. pdf"

Shacham, H. and B. Waters, 2008. "Compact Proofs of Retrievability," in the Proceedings of ASIACRYPT 2008. Springer-Verlag, pp: 90–107.

Wang, B., B. Li and H. Li, 2013. "Public Auditing for Shared Data with Efficient User Revocation in the Cloud," in the Proceedings of IEEE INFOCOM, pp: 2904–2912.

Wang, B., B. Li and H. Li, 2014. "Public Auditing for Shared Data with Efficient User Revocation in the Cloud," in the Proceedings of IEEE INFOCOM, pp: 2904–2912.

Wang, C., Q. Wang, K. Ren and W. Lou, 2009. "Ensuring Data Storage Security in Cloud Computing," in the Proceedings of ACM/IEEE IWQoS 2009.

Wang, Q., C. Wang, J. Li, K. Ren and W. Lou, 2009. "Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computing," in the Proceedings of ESORICS 2009. Springer-Verlag, pp: 355–370.