ISSN 2347 - 3983
Volume 8, No. 1 January 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter29812020.pdf
https://doi.org/10.30534/ijeter/2020/29812020

Strengthening Authentication within OpenStack Cloud Computing System through Federation with ADDS System

M Trinath Basu[1], **JKR Sastry**[2]
[1]Koneru Lakshmaiah Education Foundation, Vaddeswaram, India, miriiyala68@kluniversity.in
[2]Koneru Lakshmaiah Education Foundation, Vaddeswaram, India, drsastry@kluniversity.in

ABSTRACT

Open source cloud computing systems are frequently being used for the development of private cloud so that the cloud computing system can be updated and customized to meet the needs of the business establishments. Many open-source cloud computing systems are in use, and most of them suffer from one vulnerability or the other. Out of all the OpenStack, the open-source system used by 80% of the customers. The analysis of security built into the Open Stack reveals that much vulnerability exists, which makes the cloud computing system in-secured. Security is the primary concern considering authentication, authorization, and access control and data security. The mechanisms built into Open Stack to ensure a secure environment are vulnerable to attack.

In OpenStack, the process of authentication implemented through the use of fernet tokens. The use of a fernet token for authentication reveals many weaknesses. Open Stack did not give much consideration to use the existing and proven authentication systems used, such as ADDC and IMS. Uses of proven authentication systems as a part of the implementation of authentication systems within the cloud computing system in conjunction with native Fernet tokens will help to improve the authentication system so that secured authentication services implemented within Open Stack. Using two authentications systems within the same cloud computing system leads to the requirement of Implementation of Multi-Factor Authentication. In this paper, the implementation of a Multifactor Authentication system that integrates the Native Fernet system and the most stabilized and worthwhile ADDS Authentication system so that a user can work with any of the Applications including the Open Stack system with the single sign-on. The proposed method **implemented within the Open** Stack through making changes to the source code, addition of independent components, and customization of the configuration files**.**

Key words: Open Source cloud computing, Open Stack, ADDS Authentication System, Federated authentication, Security enforcement within open Stack.

1. INTRODUCTION

1.1 The Authenticating System

Users register with a cloud computing system using the username name and password. The registration process sometimes involves the operation of a contract, which includes pricing, services required, response time, throughput, penalties, downtime provisions, limitations on the levels of assistance needed, etc.

Users start communicating with the identity service of a specific cloud computing system by keying in the user name and password. The identity service after validating and verifying the user will send a token, which is a formatted data string containing the details of the services availed, Token to be used for access to the services, Endpoints to start Accessing the services, projects to which the user belongs, etc. The user then uses the authentication token to start directly communicating with the services intended by the user. The general process used for authentication shown in Figure 1. The service component verifies the authenticity of the user after receiving the request from the user by contacting the authentication mechanism, and on getting confirmation, the service component authorizes the user to access the service.

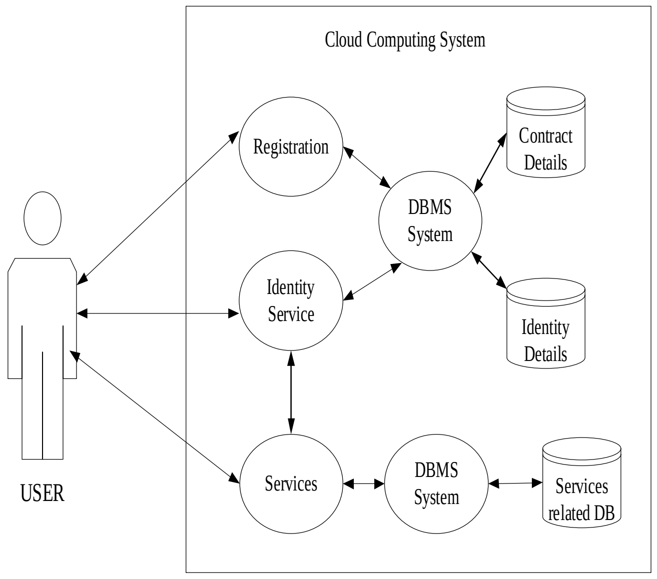

Figure 1: General Process Flow for authentication System

The user first registers into the cloud computing systems and then logs into the identity services using user name and password, which after checking the authenticates the user by passing token used for accessing the service.

1.2 Multi-factor Authentication

Multi-factor authentication means implementing the process of authentication by using multiple mechanisms. For example, an authentication system in addition to ensuring the implementing authentication within itself but also further check the credentials of the users by checking with other kinds of authentication systems such as ADDS, Red Hat IMS, Free IPA, etc.

The multi-factor authentication helps to mitigate various kinds of attacks that include brute force, social engineering, spear, and mass phishing attacks, which generally attack the user names and passwords. There is a need to deploy third-party tools and integrate the same with the underlying authentication system built to recognize the users.

Administrators or Automated preinstalled software programs or users with admin or superuser access facility configure an application such that both native and external authentication systems used considering two or more authentication factors for securing a cloud computing system. The factors considered include digital signatures, digital certificates, encryption and decryption methods, etc. There has been well established external authentication that includes ADDS, IMS, which are the third party applications that implement different protocols for effecting authentication. Multi-factor Authentication reduces the risk massively while ensuring high security to the cloud computing system.

The identity service usually is designed to work with backbends used as plugins, which may use a further extension to the process of authentication. The Identify service configured to use one or more plugins so that the authentication system implemented using multiple factors. The process called a federated authentication system. Figure 2 shows the way the multi-factor authentication system works

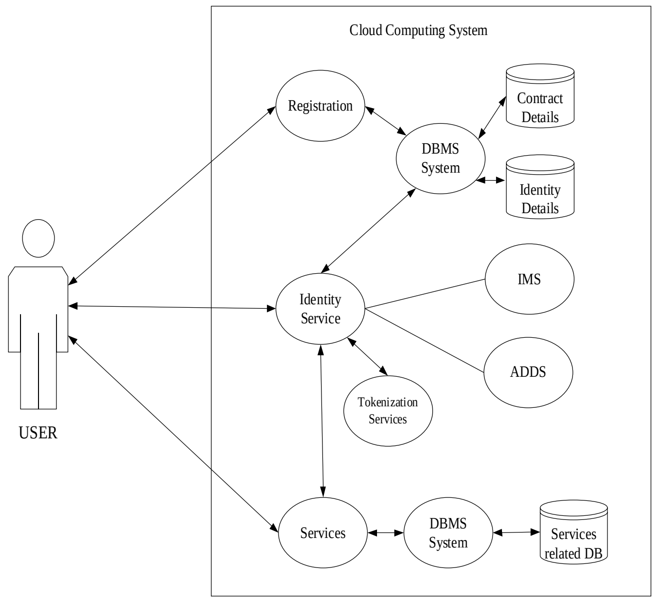

Figure 2 : Process Flow for Multi-Factor Authentication

1.3 Use of token

The token often passed as a specific structure containing the details of the services, resources that can be accessed, etc. The authentication token also provides a catalog of various services that a cloud computing system can offer — each service listed with its name, Access endpoints for internal, admin, and public access.

The token, once distributed, can be revoked by the system that made available the Token. Users can use API of the Identity service to revoke the tokens, get the list of revoked tokens, get the list of various services offered by the cloud computing system to the user who has access to the token, to remove the existing token — all queries related to the tokens initiated by the users or the services supported through API calls. The identify service provides API, which can be used for token management through operations such as token revocation, to list existing tokens, remove tokens, cache tokens, etc.

There are many types of taken management systems used in the literature, which include UUID, Fernet, PKI, PKIZ, JSON, etc. which differ from each other in many ways in terms of the content of those tokens and the way content in the token is secured. The token is the most venerable elements within the cloud computing system.

1.4 Authentication system as implemented in OpenStack

Open Stack is an open-source Infrastructure as a Service (IaaS) cloud computing software, where users can provision virtual machines by using its components such as storage (called “swift”), compute (called “nova”), etc. Figure 3 shows a high-level overview of Open Stack.

Open Stack deployed in standard hardware and its resources like computation, networking, and storage shared in the cloud.

These resources controlled using an Open Stack dashboard. Users can avail of these resources by using a client program such as an Internet browser. Open Stack has a modular architecture.

Open Stack is composed of a set of Modules that together deliver the functionality required by the user. Many modules are available, and each module provides a kind of service needed by the user. The functioning of the OPEN stack module individually attacked. To understand the extent to which an open stack is secured, each Module assessed to find the vulnerabilities and the level of security built into each of the Modules. The weaknesses of the OPEN STACK component that can be exploited by attackers must be known to make the Modules secured from attacking through the implementation of counter-attacking mechanisms.

The Security enforcement under open Stack recognized in terms of authorization, authentication access control, and data security.

The authorization and authentication service and the access control archived through KEYSTONE Module. KEYSTONE Module handles all the issues related to the identification of the users.

Virtual Images managed through GLANCE Module, but the security of virtual images handled through SWIFT, which stores all the VM images and security of which is dealt with by it.

Data in Open Stack managed through three distinct modules that include SWIFT (Object Storage), CINDER (Block Storage), Trove (Relational Databases), Regular applications use a database, and therefore, the use of Trove done extensively. Securing when TROVE used is the Major Issue. A certain level of enforcement of security done within these modules

One of the most important issues connected with the security is identifying the users, group of users, and their roles and privileges that the users have in availing the resources. The Module KEYSTONE provides the Identity services to all the other modules in Open Stack. The security issue is very much related to identifying the users and the rights that are granted to those users, such that the users provided with access to the resources for which permissions granted.

Keystone is a component of Open Stack that handles the issue of authenticating the users to have access to different services. Keystone implemented a standard authenticating system that uses a repository in which details required for authentication are stored. Users, user groups, user roles, group roles, access points, etc. are stored in a repository. Many functions implemented within keystone for effecting authentication of the users who log in with their user name and passwords.

Keystone verifies the authentication of the users by checking the association between the users, user groups, roles, and the entry points for the users to start accessing the services. Keystone maintains a catalog of the services and entry points, which are API endpoints using which the users access the services.

A user, group, service, or system is a digital representation of an Open Stack system. A tenant is an entry point form in which location the user starts accessing the service. The Keystone verifies the authenticity of the digital signature, assigns an entry point that is used by the user to access the resource.

Users, groups, tokens, tenants, roles, and endpoints are the elements used by Keystone for effecting authentication. A tenant combines resources that include processes, customers and users, etc. Roles are rights assigned to the users for being able to undertake a set of operations. Keystone issues a token to the users, which contain details related to the roles and the tenants that can be accessed by the user. Users access the services by providing the tokens to the services. A service verifies the roles using its internal policies and allows access to the service if the policies maintained by it enable access to the user having a specific role.

A token designed using a standard structure. The token intended to provide details related to the resources that can be accessed. A token is valid for some specific time, and the token revoked once issued after its expiry, in which case, the token deleted. An endpoint is a URL provided by keystone for the users to start accessing the resource

Figure 3 shows the way the keystone interacts with other components of the OpenStack. Security with the Open Stack implemented through 3 different processes relating to authentication, authorization, and data security.

The following steps followed when a user tries to access an OPEN STACK service.

1. User logs into the Keystone using the user name password, which gets established when the user registers into the Open Stack system.
2. If the username and password found to be correct, the keystone develops a token containing the details of the tenants, which is a group including the details related to the users or resources. All the users placed in a group given access to the services contained in the Group specification. The Group, as such, is called as a Tenant.
3. The user sends a cross to the service the allocated token to the user.
4. The service verifies with the keystone the authentication details using its internal policies. The user is provided with the access if keystones confirm that the token is issued by it and also that the user request confirms to one of its policies.
5. If access is allowed, the service provides the endpoint to the user, which is nothing but the URL using which the resource required is accessed.

6. If access is not allowed, the user is notified of the rejection to permit the service, giving the reasons.

The architecture of the KEYSTONE module shown in Figure 4.

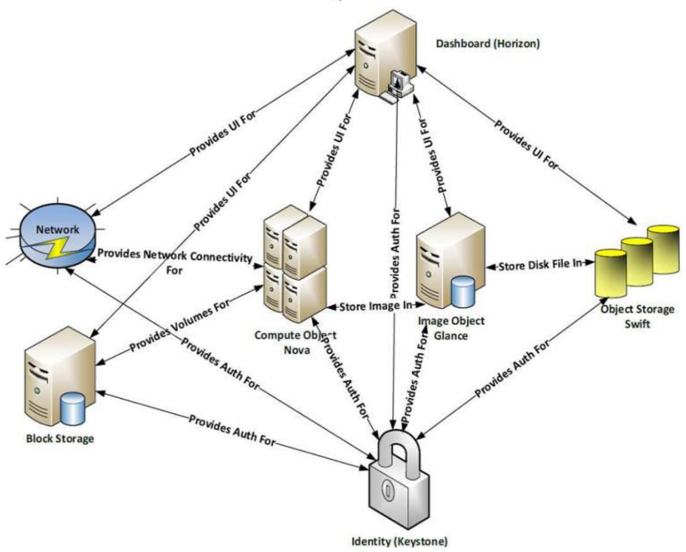

Figure 3: Overall Interactions within Key Modules

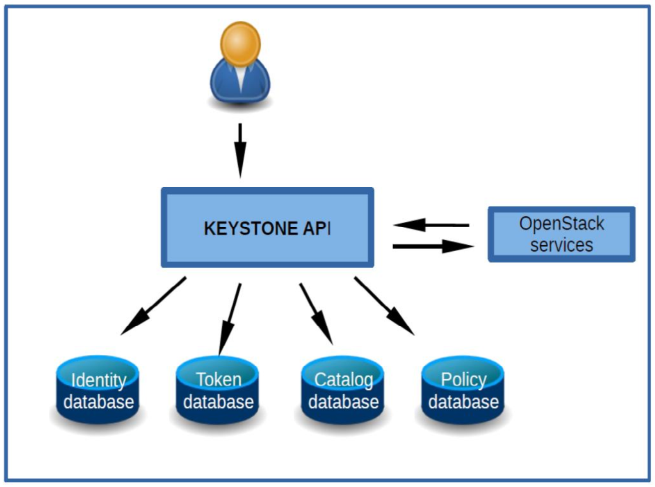

Figure 4: Keystone architecture

Within keystone architecture, four-component services include token service, Catalogue service, and Policy service, and database service included. The identity service provides authentication service that provides validation of credentials of the users, and the validation of the users concerning roles and Accounts and also validation regarding metadata data.

The "Keystone" component of Open Stack provides identity service for authenticating and high-level authorization. A token-based and service-based authorization system implemented by the Keystone component of Open Stack. Keystone is the centralized identity and access management component of Open Stack. The keystone module uses a pluggable data store (SQL, LDAP).

Keystone stores the user access details in a SQL based database or the details stored in a server that implements the LDAP protocol. The database maintained by the Keystone for storing the access details is independent, and therefore, no other data stored in the database in which access details stored.

Many external authentication systems are available, which can be opted by an organization for adaption provided that there is a compatible with its internal authentication system and when desired to implement a stronger authentication system and also that the risk involved significantly reduced. While the internal authentication system checks the correctness of the usernames and passwords, the external authentication system taken into account the digital certificates, digital signatures for authentications and also provides the endpoints based on the access control and its internal policies that it enforces.

A user can manage the OpenStack session using its dashboard (Horizon), accessed using a web browser. To be able to get the services from various services, the user has to authenticate to the Keystone server.

First, the user gives identity and credentials (e.g., password) to Keystone. Assuming the user is registered, Keystone authenticates the user, creates a tamper-evident digital token that contains information about the user, the endpoint information of each service (e.g., Nova, Neutron, etc.), and the operations the user is allowed to perform at each of those services.

Keystone authentication performed by using public-key cryptography. It uses a digital signature, and the usage of the digital signature in this system is unconventional. It is well-known that a significant drawback of the digital signature is that it takes a longer time to sign and decrypt the data. For this reason, in the real-world, a digital signature is used for small-sized data (typically hashed data). But the existing system of keystone signs large amounts of data, and this makes the keystone exists system a non- standard and inefficient for high-volume deployments.

The critical point is that a significantly more efficient and standards-based authentication protocol for Open Stack developed. It is feasible to re-designing and re-implements Open Stack's authentication protocol implemented in its Keystone component by employing different approaches. Either the authentication protocol is modified, or sometimes the multi-factor authentication system is implemented. Keystone is one of the Open Stack components used for providing identification, authentication, and authorization service. This service categorized into two primary functions, which include user Management and Service Catalogue. User Management keeps track of user's necessary data, such as what roles the user has, which project the user belongs to, etc. Service Catalogue keeps track of what services are available and provides the location of the services' endpoints.

Keystone provides identity, token, catalog, and policy services, a public key-based mechanism used in the keystone's authentication system. Public key cryptography allows users to communicate securely over public networks and verify the identity of a user using a digital signature. A digital signature is an electronic signature that can authenticate the user. In a digital signature, a sender typically uses her private key to sign the data, and the receiver uses the sender's public to the key to verify the signature. A Certificate Authority (CA) plays the trusted role to vouch for the identity of the user with a specific public key.

In Open Stack, the keystone can play the role of a CA using the keystone-manage utility or done by a third party. A Keystone PKI token used for authentication. A PKI token is nothing but a token signed by the keystone using its private key. Keystone uses cryptographic message syntax (CMS) within PKI. For this reason, the token referred to as CMS token.

Whenever the user authenticates with his/her user name and password, keystone gathers credential data (e.g., user's roles) of the user and creates a token and places them in a file called user metadata. The metadata contains all information of the user like token, service catalog, user role, etc. It also includes an issue and expiration date and the id of the token.

The project information follows next, after which the service catalog information placed. The service catalog has the information on the service(s) and related endpoints the authenticating user can avail. The endpoints are where the services should connect to obtain a specific service (e.g., compute vs. network service). After the endpoints, the information about the user listed. It shows the roles of the user, username, and id of the user. Again, this data called CMS data because the ID of the services here written in CMS format, and the signed CMS data is called CMS token.
When a user logs-in with username and password, the keystone gathers all of the information mentioned above and generates a CMS token and sends it to the user's workstation. The user's Open Stack client program in her workstation caches the token locally and uses it for later requests. When the user later requests a service, the client sends the token along with the service request to the Keystone endpoint. The Open Stack service verifies the user's signature and responds with the token.

When the client needs any of the services like nova, glance, cinder, etc., it sends a request along with the CMS token. The target service receives the CMS token and verifies the signature with keystone, and provides the requested service if the token is valid and the user is authorized.

Once a service receives the PKI token, it verifies the correctness of the same. The verification of the taken includes verification of the digital signature, token expiry date and time, and then proceed to handle revoked tokens — a keystone digital certificate required for confirmation of the digital certificate. The digital certificate can be obtained from KEYSTONE or through a third-party certificate authority. The CMS taken is verified using the certificate. If the signature is found valid, then the metadata is decrypted and then proceed to check the expiry of the token. An error is flagged if the digital certificate does not tally or the token expired.

Token verification carried using the token revocation list. The open stack services update the revocation list when the service requested is provided or when the token is either expired. The id of the token is the md5 hashed token, which is revoked by the service. Once a token is revoked, no more the token is valid. The Id, if present in the withdrawn list, then the token is found to be invalid, and if not, the token considered as accurate. Once the token found to be correct, the service is allowed, and a response provided to the end-user. If the request is for a VM, then the VM is created and the IP address of the VM along with the valid port number provided to the user, who in turn uses the IP address for further processing required by the user, such as executing a program.

The whole process of authentication based on the digital certificate, and the method has significant drawbacks.

1. A considerable amount of time taken for processing the token through digital certificate affecting the response time
2. Digital signature help securing the data integrity but not the data confidentiality as the token its self has essential information about the open stack system such as the details of the services, endpoints, etc.
3. Tokens once used are revoked, and are not re-used, leading to the signification token processing for revoking, removing, invalidating, etc.

TLS (transmission layer security) enabled within the Identify service for supporting the authentication system so that the tokens are secured while transmitted through the network. TLS provides an additional layer of security, leading to a higher level of protection. There was additional administrative overhead when TLS used. Issuing certificates to the end-user is a costly proposition.

Configuration files contain secret information, especially relating to the endpoints, and the locations access related data is stored. At any cost, the configuration files secured against any attack. It is worthwhile to uses access control frameworks such an SELinux so that the configuration files protected at any cost.

Implementation of TLS requires issues of digital certificates to the services and the users. A certificate authority required which issue certificates. The certificate authority can be situated either internally or externally. The Open Stack services check the validity of the certificates by referring the same to Certificate Authorities. However, Cloud computing Systems provided with an internal certificate server used for dealing with the certificate.

In OpenStack, Fernet tokens used as default tokens. Fernet uses a secure messaging system. The tokens designed to such that same used as Tokens accessed through specific API. There is no need to store Fernet tokens as the tokens are non-persistent. The tokens are lightweight as the size of the token is between 80 to 240 bytes, which leads to reducing the operational overheads. The authentication data contained within a packet (payload) is of the maximum size of 250 bytes. The data in the payload is encrypted and signed. There is no need to implement a token revoking system as the fernet token is non-persistence.

The fernet tokens are universal across the services leading to a reduction in overhead when services have to communicate with each other. The fernet tokens, as such, need not replicated as they transmitted across the network. The number of calls required for accessing the fernet token is few, which leads to the high proficient processing of the tokens.

However, the fernet format has problems that make it non-ideal. The fernet specification abandoned, making it hard to get changes into it and thereby into the cryptography implementation of it. Moreover, the fernet specification not recognized by any standards body and therefore not as carefully audited as an IETF standard, making it more susceptible to zero-day vulnerabilities. Addressing these vulnerabilities falls solely on the cloud computing service providers.

Some of the requirements of the users that are not supported by the fernet tokens

1. Need for a non-persistent token that does not depend on symmetric encryption or signing implementation. An implementation built on asymmetric signing or encryption allows the distribution of public keys from one node to another instead of synchronizing a repository of symmetric keys, which makes it easier for the cloud components with read-only capabilities strictly used for token validation. The asymmetric encryption method helps to deploy keys in read-only regions where the token validation undertook, while the tokens issued from a central identity management system in a separate area.
2. Need for a fall-back mechanism in the event of noticing a security vulnerability in the fernet-spec or the implementation of the cryptography.
3. As an operator, I want to be having a token provider to fall back on in the event there is a security vulnerability in the fernet spec or the cryptography implementation consumed by keystone.
4. The need for a token used not be used within a cloud computing system but also with other prices of the software, which is outside the scope of cloud computing systems.

Thus there is a need for implementing proper authentication system within cloud computing systems to make it more secured from the point of Authenticating the users

1.5 Security issues relating to Keystone Module

The critical study of the keystone module carried to find security lapses contained within the keystone module. A detailed presentation on the vulnerabilities existing in the keystone module presented in the following sections

1.5.1 Invalid Login Attempts

The keystone component implements the identity service. The user has to login using the user ID and password to begin. A user can keep on trying several times to get the user name, and the password entered is accepted. There is no check or the limit on the number of times the user can try. This kind of attack counter-attacked by restricting the number of times a user can try by setting proper firewall rules. The access log periodically checked to find the amount of time that the user has logged-in and the resources that the user is continuously accessing. The user account can be barred when any of the attacking features sensed.

1.5.2 User authentication Issue

Two functions are used by the “Keystone” Module (tempAuth () or “swath ()) for authenticating the users based on the user name and password. An authentication token is sent to the user when the user name and password found to be correct. It is the essay to attack the username and password as the same is either stored in plain text or in an encrypted mode, which can be brutal attacked. The keystone module does not use any delegation system. Use of a standard delegation system or making the password and username based authentication system much more complex such that it is complicated to attack such a system.

1.5.3 Password strength issue

User IDs and Passwords must be made secure and complicated by using NIST rules that include checking dictionary, uses of combinations of alphabets, special characters, and digits, minimum length enforcement, use of lower and uppercase letters, so that login information is secured.

There should be protection from phishing when the web interface used. The dashboard component of Open Stack allows user access through the use of the WEB interface. Neither the dashboard nor the Keystone components of Open Stack use NIST standards to protect at the time of login.

1.5.4 Password storage Issue

Keystone Module stores the user IDs and passwords in a file, either in plain text or encrypted text. The data stored in a configuration file. There is no access limit to this file. Even the superuser ID and password also saved in the same folder — the read access to the file assigned to Every user. There should be limited access to this file and mainly limited to the processes contained in the keystone module.

An insider attack can access this file and get hold of the Admin ID and password or superuser ID and password. The password stored configuration file protected from the insider attack. Saving the user IDs and passwords either in an encrypted manner or in hashed mode would lessen the risk of attacking the passwords.

Both “tempAuth” and “swAuth” lack the appropriate protection of passwords. A recommendation for both authentication systems taken from NIST’s "Electronic Authentication Guideline" would be to store passwords "concatenated and hashed with an approved algorithm, so that the computations used to conduct a dictionary or exhaustion attack on a stolen password file would not be useful to attack other similar password files."

1.5.5 Tokens of authentication

The Keystone Module sends back an authentication string over an open a channel once the user ID and passwords found to be correct. The string travel over the network before the user receives it. The token contains the information that includes user roles, service catalog, token number, entry points, which are tenants in plain text. The text, as such, attacked while traveling over the network. The authentication text secured by making it move over TLS (Transport Layer Security) channel. If the channel is not secured, a Man-in-the-Middle attack enforced.

1.5.6 DDOS Attack

The users of Open Stack can send repeated requests to access for availing the services. There is no frequency limit of the demands made by the user. The user can initiate a DDOS attack. Such an attack sensed, and the requests initiated by the users rejected when the frequency of the requests reaches beyond the limit.

2. PROBLEM DEFINITION

The Identity service provided within the keystone module leaves several vulnerabilities making any cloud computing system vulnerable for attacks. A sound security mechanism is needed to be built within the open stack cloud computing system so that the Open Stack system can be used effectively for creating either public or private clouds. A secure authentication system implemented for establishing the user credentials so that the desired access to the services provided — the authentication system secured in addition to making the authentication system faster. The authentication system must be implemented in an open platform so that the operation easily integrated with other internationally proven and available authentication systems.

3. RELATED WORKS

Platform Specific digital signatures and tokens used for authentication within Open Stack. An independent, decentralized, and flexible Mechanism that serve the purpose of authentication presented by Razib Hassan Khan et al., [1]. They have used OpenID, which is an open-source for the development and implementation of authentication within Open Stack. They have developed and offered the authentication system as a service. The platforms proposed by then are built web services and have implemented a single-sign-on for accessing multiple services. The users signing into an operating system is also used as login into Open Stack. They have shown how the users who registered into OpenID can log into Dashboard/Django GUI.

Jamie Bodley-Scott [2] presented that identity management is now being made user-centric from organization-centric approaches. Access to multiple service points implemented through user-centric approaches that are scalable and flexible. The user-centric methods based on single sign-on for making available different kinds of services. Through a single sign-on, a federation of login systems used, which improves the usability of service-oriented systems extensively.

API generally used within Open Stack EC2API [6] or OSAPI [3]) for implementing front-end GUI services. Through API, the processes that are related to Access control, authentication and cryptographic algorithms, and generation systems handled within Open Stack. Many weaknesses found when the authentication systems implemented using API calls. The user names and passwords when used within a different framework on which the service provider has no control for authenticating the user. Once the user verified, administrator credentials used for retrieving the credentials of the user. The server that provides the service in the open stack does not participate in the authentication process but rather depends on the credentials provided by the administrator.

Administering the policies and taking policy-related decisions are situated at specific policy decision points (PDP), and there can be many policy enforcement points that communicate with PDP for effecting the authentication and access control to the resources. Policy enforcement points situated within a cloud computing system that delivers with PDP for providing access to resources [4][5]. In Open Stack, the front-end GUI server within the client is a separate area within Open Stack. The user credentials have to be stored within the GUI server as Open Stack has no support for federation with the different authentication servers. The implementation of Multiple PAPs and PDPs is not possible when a centralized authentication server was not in place. There should be trust between the Client (Front-end GUI) and the cloud computing system, which is possible when a tightly coupled GUI and Cloud computing system implemented. The WEB server tightly coupled to the Back-end cloud computing servers which provide the services required by the users.

Open Stack is a cloud computing software that is available as open-source. Open Stack initially designed to of Infrastructure as a service (IaaS). Users can ask for the kind of machine in terms of CPU power, extent memory and storage required, and the nature of the operating system that must run on the Machine. Users install an IDE and develop their application. Users can also connect their System software like database management software etc. Users can deploy their claims on the machines and also run the application.

Open Stack has not used any standard for implementing security enforcement within the cloud. The security implemented within the OpenStack system is nonstandard. Kerberos is an authentication standard. SazzadMasudet al. [6] have studied the Kerberos system and have shown what the system implemented within the open stack as an independent component that federates with a keystone.

C. Kaufman et al. [7] have presented the way the authentication protocol system used by Open Stack implemented within the Kerberos protocol. They have also introduced a prototype authentication system based on Kerberos standard.

The Keystone service developed using many interlinked and structured internal services [8]. The services offered by keystone can be services provided by keystone include policy services, catalog services, token services, and Identity services. The authentication system implemented within the keystone based on critical public infrastructure that helps the users to communicate securely over public networks — the identity of the users established through a digital signature. A user uses a private key to sign the message digitally, and the receiver uses the public key of the user to decrypt the message and find the identity of the user who sent the message. A certificate authority, either internal or external, will verify the trustworthiness of the public key used by the user.

Marek Denis et al., [9] have explored the implementation of identity federation within the Open Stack system through the use of local identity called "Domain Accounts."

Most of the organizations around the world are shifting to cloud computing infrastructure for supporting their IT requirements due to cost, reliability, and availability of the required resources as and when needed. Many cloud service providers have already come into the market, playing a significant role. Some of the service providers that have come into play include SalesForce, Amazon, Google, Microsoft, Rackspace, Oracle, Verizon, etc. [10].
Several frameworks developed in the past related to cloud computing systems. Some of the frameworks are open source based. The customers use the frameworks for building private clouds that offer different types of services. Some of the notable frameworks include Cloud Stack, Eucalyptus, and Open Nebula. Off late Open stack has become the most sought out Open sources based framework for developing users their private networks [11].

A study conducted and an analysis of security issues relating to open stack carried especially considering object storage service. Security requirements, as stated in two different standards released by NIST(National institute of standards and technology) and ENISA (European Network and Information security agency) and came out with a set of security requirements implemented within Open Stack [12].

A new Enhanced authentication system that works in conjunction with the original authentication system implemented within the keystone presented by B. Cui and T. Xi in [13]. They have shown details and the way the new model performed. They have compared the features of the new model with the features supported by Keystone and also have introduced the way the new model provides a high level of security by subjecting the open stack system with the attacks that cannot be mitigated by the keystone system.

Series of versions of the Open Stack released, leading to the improvement of security enforcement that mitigates many of the vulnerabilities existing in the open stack. The openstack being open source exposes many threats and vulnerabilities. Open source-based Testbeds used for testing cloud computing systems, and these testbeds used to test new methods included in the Open Stack system for testing resource provisioning and management of services deployed under Multi-data centers [14].

Authenticating the users for providing secured storage and access to the information is required, when it comes to service-oriented information exchanges — identity management systems needed for ensuring confidentiality and security considering both sides of the client and provider. Many drawbacks exist within many of the cloud computing systems, including open stack, which causes data violations, unauthorized access. A single point of failure happens due to the use of centralized access. Security of cloud computing systems enhanced through Federated Identity management, which leads to the structured, adaptable, and systematic implementation of the security systems within cloud computing systems [15].

A cloud computing adaption framework is proposed by Vicor Chang et al., [16], which can be customized by the user for implementing the organization-specific security requirements. They have presented that security enforcements applied in real-time through a Multi-Layer approach. They have used three layers for implementing security through a firewall, intrusion, and access control, implemented in three layers.

Benjamin Ertl [17] presented that Authentication for every kind of service has to be rendered based on cross-domain identification. They have introduced a protocol that considers the issue of linking different accounts associated with a client. The protocol proposed by them supports verification of authentication for each of the services requested by the clients. They have put their recommendation in terms of the existing federated infrastructures.

Security and privacy are the two major concerns of the users who store their data in the clouds. Several security concerns arise, which include access control, Data integrity control, access logging, access auditing, and managing the identity of the users when data transmitted between the user and the cloud. Many complex issues lead to multiple open problems requiring in-depth research carried Bhale Pradeep Kumar [18].

Policy-based methods and mechanisms used for effecting access control have been used by Georgios Katsikogiannis [19] for implementing multilevel identity integration, authentication, and authorization for providing secured access

to cloud computing resources. They have used SOA for implementing a policy-based security system. They have analyzed Identity integration, user roles, authentication, authorized access control, and used for validating the rules.
A cryptographic primitive, which is key-homomorphic embedded into RDIC (identity-based remote data integrity checking) protocol, which considers the user identity for reducing the complexity of a system. The modified RDIC helps in minimizing the cost of implementing PKI within RDIC, Yong Yu [20].

A cloud computing platform suffers from both external and internal attacks. Nit many Mechanisms/methods proposed to deal with internal attacks. Carlos Eduardo et al. [21] described self-adoption schemes to handle insider threats. The authors have presented the way the self-adaption systems introduced into the Open Stack cloud computing platform

The self-adaption mechanism found to deal with the uncertainty that exists in a wide range of applications. The component "Keystone" contained in open stack can be included with self-adaption mechanisms so that internal threats counter-attacked

Carlos Eduardo et al., have presented that adds self-adaption components to Open Stack architecture to handle insider threats. They have analyzed several threats occurrences that can happen within the open stack and have evaluated the impact of the treats that occur in several scenarios. The self-adaption system explained considering several threat scenarios.

Various models presented in the literature relating to authentication, authorization, and access control helps to implement different security measures that help to ensure integrity, confidentiality, and availability of the data as per the user requirements. Many models presented in the literature include RBAC (Role-based Access control) [22], ABAC (Attribute-Based Access Control)[23]. These models, based on the assignment of attributes to roles either through relationships or rules, assign permissions to access the resources, find Rules that express relationships between the users and roles. An identity federation management system includes identity providers, assigned attributes to the users, and uses the Authentication related infrastructure provided by the service providers for effecting the security enforcement [24].

The system that implements ABAC/RBAC relies on the software components that protect access to several resources. The self-adaptive systems use the inputs provided by the elements meant for controlling the access for changing their behavior [25]. MAPE-K framework implements the self-adaptive model combined with the components that protect access to the resources [26].

The authentication systems implemented within the cloud computing systems must be flexible such that the authentication system implemented varies based on the kind of resource requested by the user. Many existing cloud computing systems implement proprietary authentication systems through uses of signatures and tokens. Khan [27] has designed and implemented a model based on the OpenID framework so that limitations existing in proprietary protocols removed.

A new authentication framework is proposed by Anisetti [28], which is deployed on a single open stack node and proved the effectiveness of the framework in implementing security within Open Stack. Cui [29] et al. have analyzed the security implementation within an Open Stack, considering each component separately. They have proposed a new model based on symmetric and asymmetric encryption the feasibility verified by deploying the same within Open Stack.

R. T. Fielding [30], in his thesis, has presented REST architecture for designing web applications. REST is stateless and works on the principle of cloud computing. The API uses a secured https protocol for proving communication between the users and a server. The users through API call logs into the server and get connected

EC2API client [31] and the python-nova OSAPI client [32], which are API tools used for authentication of the users into cloud computing systems. A GUI hides the use of API and makes accessing a service much simpler for the users. WEB-GUI became the widely acceptable front end for making available the cloud services to the end-user and also the administrators. The Horizon component of Open Stack provides the GUI through which the user can access the services without the need for accessing the system through the use of API. There are, however, many shortcomings in the way the dashboard provides authentication of the end-user.

Open Stack does not support federated identity management. Many federation based authentication systems such as OpenID [33][34[, SAML [35] [36], Shibboleth[37] used within the open Stack for implementing the authentication system

OpenID is an authentication system available as open sources extended to implement user preferences. The system provides a centralized Identity system that is user-centric, which means the users can opt for the kind of system that needs performing for enforcing the authentication to access the cloud computing resources.

In Open Stack, Security is built through the process of authentication and access control and also providing security infrastructure that can be used by the users to enforce security on their own. The Keystone Module within open stack takes care of security enforcement through primarily utilizing a process based on Tokenization. The Keystone module provides tokens to the users, and the users access the services offered by Open Stack with the help of tokens

Open Stack [38] is an open-source system that offers Infrastructure as a Service (IaaS)[39]. Open stack allows the users to provision the virtual machine with storage, computing resources, which are provisioned by the Open Stack components such as SWIFT, NOVA, etc. RACKSPACE and NASA together have developed Open Stack in python language [40].

Keystone uses Digital signatures for implementing the authentication system. Authenticating a massive amount of text would be cumbersome [41] when massive traffic between the clients and the cloud computing system expected. The authors have proposed to hash the data so that the size of the text reduced and then they have applied a digital signature

Darshan et al. [42] have presented that keystone plays a significant role in binding all cloud computing projects together, each project implementing a service. There is a need to protect the resources used by the keystone such token repository, the identity of the users and resources, the endpoints, etc. The security of the open stack system is enormous as all the possible attackers have the source code of the Open Stack in their hands. The authors have analyzed the security requirements openstack and formed a threat model. A Restful API based authentication system that offers various security services needs implementation within the Open Stack system.

Security concerns are many when one wants to use cloud computing systems. Security is a real barrier to using cloud computing systems [43]. A survey conducted in 2016 revealed that security risks are the primary concerns/obstacles in using cloud computing systems.

Open Stack is vulnerable to attack. Experimentation using a prototype specification revealed that the dashboard component of Open Stack is sensitive for attacking. Most of the researchers have offered different kinds of solutions used for making the Open Stack framework secured. The keystone module of the OpenStack provides another type of identity services that include identification, cataloging, management of policies, and dealing with tokens for authentication. The identity services offered by the keystone module provided as a set of services situated at more than one endpoint. The services initiated through frontend calls. An authenticate call initiated from the user frontend will validate either project or user credentials, and on finding the eligibility, an authentication token issued using which the users access the service [44].

GidwaniIshan et al. made another study that focuses on the security issues and threats existing in Open Stack system., [45]. The authors argued that Open Stack did not implement any complexity within the password system and also that the passwords stored in plain text. They have conducted a penetration test using some of the existing tools and have come out that Open Stack is prone to future attacks as many vulnerabilities still existing in the system

A study of the features supported in the keystone module [46] has found many weaknesses and a lack of support for access control, authentication based on attributed provisioning, audit mechanisms, and policy-based security enforcement. They have carried a threat and identified threats that exist concerning interfaces, components, internal processes of the elements within Open Stack.

The architecture of most of the open sources is similar [47]. Most of the cloud computing architectures consider including a cloud controller and a set of nodes on which several services implemented. The controller controls the instances, network, administrative interfaces, and schedules the interfaces. The nodes run instances of VMs through the use of available resources.

Performance analysis of a two factor authenticated system carried by J. M. Alve, T. G. Rodrigues [48] using the different hypervisors, which include VMware, Xen, KVM, etc. They have used the user name password in the first instance and then followed by OTP based authorization. They have used OpenID protocol, so that single sign-on access to the services provided. They have shown that KVM hypervisor performance extensively well using a two-factor approach [48].

KrysztBenedyczak et al. have presented the use of middleware for implementing federated computing [49]. They have proposed a method that does not require either certificates or delegation mechanisms. They have used a component called “Unity” for serving the identity management services. The method proposed by them allows many federation integration approaches that include integrating with OpenID and SAML.

Controlling access to different resources considering Fine-grained access control, Scalability, data utilization, privacy preservation, and revocation of privilege is most complicated. A scheme covering these aspects was proposed by Rohit Ahuja [50] based on encryption carried using a set of attributes. Encryption of data undertaken through consideration of a set of attributes. They have considered that the users hierarchically organized, with each user carrying specific attributes. The characteristics are selected based on the path to be used for moving data from one user location to another, keeping because of the scalability. The authors have presented the method of hybridization of re-encryption and attribute-based encryption to realize the flexible revocation of system privileges.

A single sign-on is sufficient to access the services proposed by multiple service providers recommended by JaweherZouari [51]. They have proposed identity as a service framework in which an Identity Finder system incorporated. The identity finder system, associates service providers with identity providers systems after taking the consent of the users. They have proposed additional functionality that helps to transform between different standards and mapping semantics relating to varying attributes so that the same identity context preserved over the entire system

X Darth protocol has been used by QuratulainAlam [52] for implementing identity management that spread across several domains. The follow of information that takes place when XDAuth used is modeled using Petri nets at a high level. The authors have used the Z language for analyzing the rules of information flow. The model verified by using a Z3 solver.

A scheme that helps to implement RIBS (Revocable Identity-based signature) proposed by Xiaoping Jia [53] uses an external cloud revocation server used for carrying all critical updates. Xiaoping has proved that a new framework that incorporates RIBS is highly resistant to foraged messages and different identity attacks. They have convinced that RIBS is highly efficient when compared IBS scheme.

Mell PM et al. [54] explained that the distribution of data among different servers is primarily dependent on the type of cloud (private, public, and Hybrid). The data distribution is also dependent on the users who are either internal, external, or both. The data distribution aspects considered while making available infrastructure as service through an open stack cloud computing system.

The self-adoption techniques did not lead to complete protection considering security and privacy, especially when insider threats are involved. Many contributions made to deal with securing the cloud computing systems [55, 56, 57], but none of these methods could solve the issues of insider attacks.
Cappelli DM et al. [58] have explained that an insider threat is either a user or process that has authorized access to the internal resources and can attack the integrity, availability, and confidentiality of the data.

Cole DE et al. [59] have explained that insider threats are not the same as those connected with the cloud computing components, which are either hypervisors or brokers. Insider threats can be at catastrophic resulting in considerable losses to the organizations as explained by Duncan A et al., [60]

Self-adaption systems proved to be effective in dealing with insider threats as the mechanism deal with un-certainty considering a wide range of application and especially with the apps that are related to effecting access control mechanisms [61, 62, 63, 64]. De Lemos R et al. [65] have explained that the self-adaption systems could modify/update their behavior or data structure at run-time, thus dealing with the dynamic management of insider attacks.

An insider threat generally caused by authorized users of the system [66]. The internal user regarded as the inside attacker. Cert et al. [67] defined an employee, business partner, and contractor who has access to the internal information resources as the inside attackers. The users have intensions to take advantage of the company's data for unlawful activity affecting availability, integrity, and confidentiality regarded as inside attackers [68] [69]. Cappelli DM et al. [70] have considered the issue of security from the point of likely abuse of the data by the users, which can lead to some threat 71]. Insider risks classified as unintentional and intentional. Only intentional threats are classified into insider attacks [72]

An enhanced scheme presented by Yapping Chi et al. [73] strengthen the authentication system implemented within the keystone module of the Open Stack System. The project uses FreeIPA for including a sentinel that performs authentication, service management, and access control. The effectiveness of the sentinel tested by exposing the open stack to the external users.

Clouds provide resources that are shared by several users/tenants through availing of different services that are made available by cloud computing providers — the accessibility to the resources controlled so that one user does not get into another user jurisdiction. Several security mechanisms must put in place, which includes authentication and access control to provide non-conflicting access to the resources [74][75].

Chi Yaping et al. [76] have used the FreeIPA framework and developed a sentinel who has been introduced into an open stack and proved the effectiveness of the same. Several papers published relating to securing various aspects within cloud computing systems some of which have not been included in Open Stack [77][78][79][80][81][82][83][84] [85][86][87].

4. THE GAP

None of the authentication systems proposed for implementation in Open Stack considered the use of a well-proven authentication system already in existence and used so that a single sign-on is sufficient to access multiple applications, including the Open Stack System. A multi-factor authentications system is required to implement a full proof security system within the Open Stack.

5.0 INVESTIGATIONS AND FINDINGS

The Vulnerabilities were existing in the Keystone module investigated from different perspectives, especially the issue of tokenization and the use of multi-factor authentication. Several mechanisms can be introduced into the Open Stack so that the identity of the users can be made more secure. The measures added into Open Stack include the introduction of more secured Tokens, implementation of multi-factor authentication through federation approaches, etc. Every path leads to some complexity. The more security built into the cloud computing system, the more will be the cost, especially in terms of loss of response time, which also requirement of sophisticated security models to be added into the system. The security models chosen must match the risk involved in providing a specific service required by the customers. The risk mitigation based security model is the most ideal.

Microsoft introduced Active Directory (AD) in which information about the domains and its related IP addresses, valid users and their passwords, details of the devices such as printers, disks, files, telephone numbers, etc. The directory queried using a standard API. The directory can be moved into different clients so that checks/decoding required carried in client location. The directory, as such moves over a network. AD is a shared infrastructure for managing and administering various network resources. Sometimes the directory is stored in a server, and the server queried for want of information especially the domain names and decoding of which to the IP addresses. AD considers every resource as an object and maintains different attributes of those objects.

A set of rules used to name the resources the details stored in AD. The user names used by AD are unique, and there will not be a name collision. The ADDS (Active Directory Domain system) implements the access control mechanisms so that only eligible users and the processes could access/Query the AD. ADDS is called the domain controller as it provides unique resource IDs given the name of the resources. ADDS system controls access to the information related to the resources - the details of which stored in AD. It implements the authentication and authorization system that regulates access by the users .to the resources contained in AD based on access policies. ADDS system provides that include directory services, federation services, certificate service, rights management services, and Lightweight Directory Services using LDAP (Lightweight directory access protocol). Users can access the services through a standard API supported ADDS

ADDS server can be stand-alone or installed as a cluster. AD is distributed among several servers when clustered architecture used primarily when high availability is required. To implement a fault-free environment, ADDS performed as a Primary domain (PD), and a secondary domain (SD) with SD replicated using PD from time to time. In the case failure of PD, SD accessed until the time PD repaired and replicated again.

Each time a user makes a request for a resource, ADDS logs in the request, accesses a network resource, or runs an application, and the AD domain controller either authenticates the request are rejects the request if permissions for the resource access does not exist. Corruption in the ADDS database or the failure of the domain controller server can devastate an enterprise, so administrators often set up ADDS on a server cluster for automatic replication and synchronization for resiliency and added performance.
Smaller organizations can use Active Directory Lightweight Directory Services, which functions almost identically to ADDS but does not need domains or separate domain controllers.

Active Directory Certificate Services creates, validates, and revokes public key certificates used to encrypt files, emails, virtual private network traffic, and Transport Layer Security/IPsec network traffic. Active Directory Federation Services provides a single sign-on service to give users access to resources or services -- typically outside of the enterprise -- using one set of credentials. Active Directory Rights Management Services controls encryption and access control for email, documents, and web content.

ADDS system used for authenticating and authorizing the users to have access to different resources contained in the computer system. Every user needs to be authenticated by the operating system before the user is allowed to have access to the application. The Application intern depended on the access provided by the operating system or enforces an additional security system built within the application. The app uses the ADDS services for verifying the access rights of the user. The access mechanism as supported by the ADDS systems, shown in Figure 5.

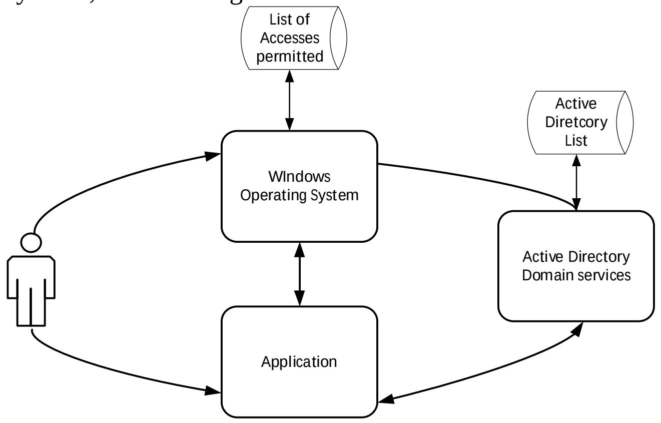

Figure 5: Authentication and authorizations ADDS system

As explained earlier, Keystone uses fernet services for authenticating and authorizing to have access to different resources, and the bottlenecks of using such a tokenization system need consideration. Active services proved to be a versatile system of enforcing security, and ADDS system provides extensive services used to enforce security. Federation of ADDS system with fernet gives high-security provisions, and also existing users need not have any additional registrations with Open Stack. The way the ADDS system federated with Fernet shown in Figure 6.

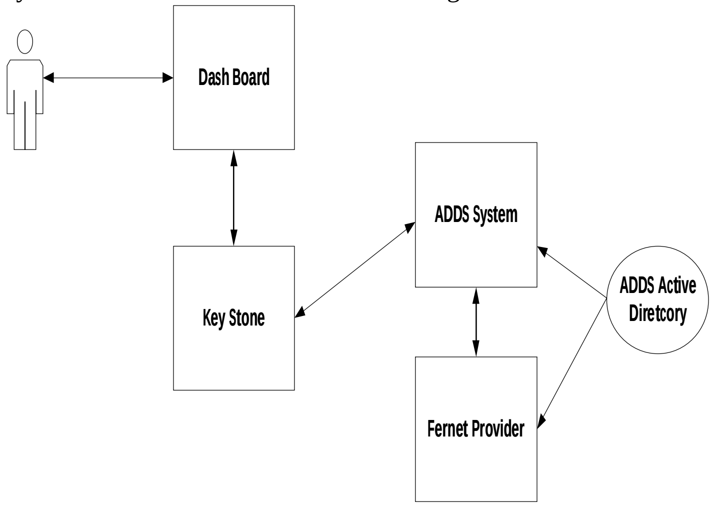

Figure 6: ADDS federation with Fernet Tokenisation system

It is necessary to complete prior tasks that include installation, configuration, and operationalization of ADDS, Open Stack, and DNS Systems and also that ADDS configured to use LADAP using port number 636, so that ADDS properly integrated with Keystone services All components of Open Stack that include NOVA, COMPUTE, KEYSTONE, HORIZON restarted for effecting the ADDS system for authentication purposes. Users account created in the ADDS system for making the users interact through DASHBOARD.

If a firewall installed to filter the communication between Open Stack and other services, then a configuration of the firewall is needed to allow the traffic between Keystone and ADDS systems. An Open Stack controller node should be able to communicate with ADDS using LDAP and port number TCP636.ADDS configured after the installation is complete. The List of configuration tasks carried shown in Table 1 and the configuration steps with its related commands shown in Table 2.

Table 1: ADDS Configuration Tasks

TASK Serial	Task	Details
1	Create a service account.	This can be named according to your naming convention for service accounts, for example, svc-LDAP. This can be a regular domain user account or Administrator privileges attached accounts
2	Create a user group.	Every user given access to the Open Stack must be a member of a Group. A group name selected such that the name conveys the meaning and follows the naming convention. Access to the projects included in the dashboard granted to the Group, provided the users also configured members of the Project Groups.
3	Create Project groups.	There must be an AD group for every Open Stack project. An ADDS group created for every Open Stack Project.
4	Configure the service account.	There must be a service account such as “svc-LDAP” configured as a member of an Open Stack Group
5	Export the LDAPS public key.	The public key converted to a certificate file in the x509 format file DER-encoded x509 .cer file.
6	Send the key to the Open Stack administrators.	The certificate file must be stored at a location as required by the Open Stack — the certificate used for encrypting the communication that happens between the ADDS and Keystone.
7	Retrieve the NetBIOS name of your ADDS domain.	The ADDS domain name must be retrieved to set the Open Stack domain name for adapting the consistency between the environments

Table 2: ADDS Configuration Steps and Commands

Config. Serial	Configuration step	Configuration Command Strings
1	An LDAP lookup account needs to be created. The keystone will use this account for querying the ADDS through LDAP	PS C:\> New-ADUser -SamAccountName svc-ldap -Name "svc-ldap" -GivenName LDAP -Surname Lookups -UserPrincipalName svc- ldap@lab.local -Enabled $false -PasswordNeverExpires $true -Path 'OU=labUsers,DC=lab,DC=local'
2	A password set for the LDAP lookup account. The password must be enabled. The password entered which must comply with AD naming convention	PS C:\> Set-ADAccountPassword svc-LDAP -PassThru | Enable-ADAccount
3	A group must be created for all the OpemStack users such as grp-openstack	PS C:\> NEW-ADGroup -name "grp-openstack" -groupscope Global -path "OU=labUsers,DC=lab,DC=local"
4	Create Project Groups	PS C:\> NEW-ADGroup -name "grp-openstack-demo" -groupscope Global - path "OU=labUsers,DC=lab,DC=local"PS C:\> NEW-ADGroup -name "grp-openstack-admin" -groupscope Global - path "OU=labUsers,DC=lab,DC=local"
5	The svc-LDAP user added to the grp-OpenStack	PS C:\> ADD-ADGroupMember "grp-openstack" -members "svc-ldap
6	The public key of LDAP encoded into DER-encodedx509.cer file and copy the same into an administrator-defined location, The must be done from a Server where AD is stored	
7	The NetBIOS name of the ADDS system should be retrieved	PS C:\> Get-ADDomain | select NetBIOS Name NetBIOSName

The commands along with arguments can be programmed into Batch File or through execution of either Python program or a C++ program through system calls can be initiated.

For the keystone to access the ADDS services for authentication and access control purposes, the Multi-factor authentications systems parameter set, and the system used to set as ADDS. Keystone from then calls the ADDS system using API for implementing different security processes. If ADDS and Keystone are resident on different servers, then there is network flow which must be secured requiring the use of a digital certificate. The location of the Digital certificate stored in the Keystone configuration file. Keystone makes an inquiry to ADDS for a public key converted in a certificate stored at the location indicated in the keystone configuration file. The certificate file configured to copy into it the Private and other certificate details shown in Table 4.

A standard protocol required for communication with the ADDS system to avail of the directory services. LDAP (Lightweight Directory Access Protocol) is one of the protocols used as it proved to be extensive and stable. The protocol used for locating any user, resource files, and devices situated on a network. LDAP provides the integration required in-between Keystone and ADDS system. Keystone implements the authentication and authorization through the following steps indicated in the LDAP protocol.

Requests for authentication from the user received by Keystone delegated to the ADDS system through the use of LDAP protocol. On receiving the authentication confirmation from ADDS, the Keystone then creates a token and forwards the same to be stored in AD and then passes on the token to the user

The LDAP deals with a set of attributes for proper functioning as required by the user. One of the characteristics related to identifying the user as admin, superuser, or user groups such as “Finance,” “HR,” etc. The roles played defined in the keystone configuration file mapped to the users which are then used by various services to determine whether a requested service extended to the end-user, LDAP requires some attributes, and Keystone requires some characteristics and the mapping between these attributes are needed. The mapping between the keystone attributes and LDAP attributes stored in the keystone configuration file.

No other program should be allowed to call LDAP for authentication outside the Open Stack for ensuring that no changes to the configuration files carried outside the Scope of Open Stack.

Having configured the ADDS system and the digital certificate that it has to communicate with an identity server, the next step is to set the KEYSTONE identity service itself to integrate with the ADDS system. Several actions to be undertaken for enabling keystone command line access which includes, configuring controller and compute function, configure block storage, for using Keystone 3 component, set to allow ADDP group users to get access to Open Stack Projects, and configure to provide ADMIN TAB access to the administrator. Once these steps completed, the ADDS system gets fully integrated with the Keystone service. Table 3 shows the steps involved in configuring the Keystone module to incorporate with ADDS system.

Table 3: steps for setting the LDAP certificate

Config Serial	Configuration step	Configuration Command Strings
1	The LDAOS public key is copied to the server running keystone and converted to a certificate file.addc.lab.local.cer: is the certificate file The RHEL Certificate store if it is needed to run the Diagnostic commands such as LDAP search	# openssl x509 -outform der -in addc.lab.local.pem -out addc.lab.local.crt # cp addc.lab.local.crt /etc/ssl/certs/
2	The certificate file ddc.lab.local.cer is converted ot .pem file	# openssl x509 -inform der -in addc.lab.local.cer -out addc.lab.local.pem
3	The .PEM program is installed on the server when Open Stack controller is installed	# cpaddc.lab.local.pem /etc/pki/ca-trust/source/anchors/ # update-ca-trust

6. CONCLUSION

Open Stack is a prominent open-source software for providing the infrastructure as a service to the users. It is necessary to investigate the sufficiency of the security built into the Open stack as OPEN STACK contemplated to be used by many users.

Management of tokens issues to the users is the most crucial issue. The system must be such that it is difficult to attack the initial logins or the tokens exchanged for providing authentication to the user.

The fernet tokens used within the keystone is weak as it exposes much vulnerability that can be exploited by the attackers. Fernet token formatting is also nonstandard.

ADDS system is versatile and well-proven and used for many applications for implementing security enforcement. Many users created into this system for accessing applications implemented within windows operating systems. The use of the ADDS system not only will help system integration across the applications but also help to achieve the accesses using a single sign-on.

Multifactor authentication using ADDS and LDAP within the OpenStack system helps in securing the Open Stack in a secure manner, which will satisfy the end-user enormously.

REFERENCES

1. Razib Hassan Khan, JukkaYlitalot and Abu Shohel Ahmed, OpenID Authentication As A Service in OpenStack, 7th International Conference on Information Assurance and Security (IAS), PP. 372-377, 2017
2. Jamie Bodley-Scott, "IDM09, Access or Identity, http: //www. open group. org/Jericho/idm2009_jbs .pdf."
3. "Amazon AWS EC2 API Reference, https://docs.amazonwebservices. Com /awsec2 /latest/ apireference/, 2011."
4. Rackspace US, Inc., "Openstack compute developer guide api 1.0, 2011."

5. S. Almulla and C. Y. Yeun, "Cloud computing security management," in Engineering Systems Management and Its Applications (ICESMA), 2010 Second International Conference on, 30 2010-April 1 2010, pp. 1-7.
6. D. Gollmann, "Computer security," Wiley Interdisciplinary Reviews: Computational Statistics, vol. 2, no. 5, pp. 544-554, 2010. [Online]. Available: http://dx.d0i.0rg/l 0.1002/wics. 106
7. SazzadMasud and Ram Krishnan, Kerberos-Based Authentication for OpenStack Cloud Infrastructure as a Service, IT CoNvergencePRActice (INPRA), volume: 3, number: 2 (June), pp. 1-24.
8. C. Kaufman, R. Perlman, and M. Speciner. Network Security: Private Communication in a Public World. Prentice-Hall,2002.
9. R. Xu. Keystone authentication. http:// OpenStackoz.blogspot.com/2014/08/keystone-authentication.
10. Marek Denis, Jose Castro Leon, Emmanuel Ormancey, Paolo Tedesco, Identity federation in OpenStack - an introduction to hybrid clouds, 21st International Conference on Computing in High Energy and Nuclear Physics, DOI:10.1088/1742-6596/664/2/022015
11. Ristov S, Gusev M, Kostoska M. Security assessment of OpenStack open-source cloud solution, Proceedings of the 7th southeast European Doctoral Student Conference (DSC2012). 2012: 577-587.
12. S. Ristov, M. Gusev and A. Donevski, "Security Vulnerability Assessment of OpenStack Cloud," 2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks, Tetova, 2014, pp. 95-100
13. Slipetskyy R, Security issues in OpenStack, Master's thesis, Norwegian University of Science and Technology, 2011.
14. B. Cui and T. Xi, "Security Analysis of OpenStack Keystone," 2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Blumenau, 2015, pp. 283-288. DOI: 10.1109/IMIS.2015.44
15. Cirrus, O.: Open cirrus - open cloud computing research-tested (Apr 2012), https://opencirrus.org/
16. RohitShere, Sonika Srivastava and R.K. Pateriya, A Review of Federated Identity Management of Open Stack Cloud, International Conference on Recent Innovations in Signal Processing and Embedded Systems (RISE), 2017
17. Vicor Chang and Muthu Ramacharandra, "Towards Achieving Data Security with the Cloud Computing adoption framework," IEEE Transactions on services computing, Vol.9, No.1, pp. 138-151, Feb. 2016
18. Benjamin E, Identity Harmonization for Federated HPC Grid and Cloud Services, IEEE proceedings, Pp. 621-627, 2016.
19. Bhale Pradeep Kumar, "Achieving Cloud Security using Third-Party Auditor, MD5 and Identity based Encryption", International Conference on Computing, Communication, and Automation, pp. 1304-1309, 2016.
20. Georgios Katsikogiannis, "An Identity and Access Management approach for SOA," IEEE International Symposium on Signal Processing and Information Technology, pp. 1-6,2016.
21. Yong Yu, "Identity-based Remote Data Integrity is hacking with perfect data privacy-preserving for cloud storage," IEEE, pp. 1-11, 2016.
22. Carlos Eduardo Da Silva, ThomásDiniz, NelioCacho, and Rogério de Lemos, Self-adaptive authorization in OpenStack cloud platform, JournalofInternetServices and applications, Journal of Internet Services and Applications (2018) 9:19, https://doi.org/10.1186/s13174-018-0090-7
23. De Lemos R, Giese H, Müller H, Shaw M, J. Software engineering for self-adaptive systems: A second research roadmap.
24. De Lemos R, Giese H, Müller H, Shaw M, editors. Software Engineering for Self-Adaptive Systems II, Lecture Notes in Computer Science, vol 7475. Berlin: Springer; 2013. p. 1–32. https://doi. org/10.1007/978-3-642-35813-5_1.
25. Clercq JD. Single Sign-On Architectures. London: Springer-Verlag; 2002. p. 40–58. http://dl.acm.org/citation.cfm?id=647333.722879
26. Chadwick DW, et al. PERMIS: A Modular Authorization Infrastructure. ConcurrComput: PractExper. 2008;20(11):1341–57. https://doi.org/10. 1002/cpe.v20:11. http://dx.doi.org/10.1002/cpe.v20:11
27. Sandhu RS, et al. Role-Based Access Control Models. Computer. 1996; 29(2):38–47. https://doi.org/10.1109/2.485845. http://dx.doi.org/10. 1109/2.485845.
28. Hu VC, et al. SP 800-162. Guide to Attribute-Based Access Control (ABAC) Definitions and Considerations. Tech. Rep., National Institute of Standards and Technology. VA: McLean and Clifton; 2014.
29. Chadwick DW. Federated Identity Management. In: Foundations of Security Analysis and Design V, Lecture Notes in Computer Science, vol 5705. Berlin: Springer; 2009. p. 96–120. https://doi.org/10.1007/978-3- 642-03829-7_3.
30. Hu VC, et al. SP 800-162. Guide to Attribute-Based Access Control (ABAC) Definitions and Considerations. Tech. Rep., National Institute of Standards and Technology. VA: McLean and Clifton; 2014.
31. De Lemos R, Giese H, Müller H, Software Engineering for Self-Adaptive Systems II, Lecture Notes in Computer Science, vol 7475. Berlin: Springer; 2013. p. 1–32. https://doi. org/10.1007/978-3-642-35813-5_1.
32. Kephart JO, Chess DM. The Vision of Autonomic Computing. IEEE Comput. 2003;

36(1):41–50.http://dx.doi.org/10.1109/MC.2003.1160055.

33. KHAN R H, YLITALO J, AHMED A S. OpenID Authentication as a Service in OpenStack[C]//IEEE. 7th International Conference on Information Assurance and Security, December 5-8, 2011, Malacca, Malaysia. New Jersey: IEEE, 2011: 372-377.
34. ANISETTI M, ARDAGNA C A, DAMIANI E, et al. Toward Security and Performance Certification of Open Stack[C]//IEEE. 2015 IEEE International Conference on Cloud Computing, June 27-July 2, 2015, New York, USA. New Jersey: IEEE, 2015: 564-571.
35. CUI Baojiang, XI Tao. Security Analysis of OpenStack Keystone[C]//IEEE. 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, July 8-10, 2015, Santa Catarina, Brazil. New Jersey: IEEE, 2015: 283-288.
36. R. T. Fielding, "Architectural Styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000
37. "Euca-Tools, Eucalyptus Community, http://open.eucalyprus.com/wiki-/toolsecosystem, last accessed 10th May 2011."
38. "Python-OpenID 2.2.5, http:// pypi.python.org/pypi/python-openid, last accessed 5th May 2011
39. OpenID Foundation, http://openid.net, last accessed 14th June 2011."
40. D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," in Proceedings of the second ACM workshop on Digital identity management, ser. DIM '06. New York, NY, USA: ACM, 2006, pp. 11-16. [Online]. Available: http://doi.acm.Org/10.1145/1179529.1179532
41. M. Erdos and S. Cantor, "Shibboleth architecture protocols and profiles, Http://shibboleth. internet2. edu/shibboleth-documents .html."
42. R. Philpott, E. Maler, N. Ragouzis, J. Hughes, P. Madsen, and T. Scavo, "OASIS Open 2008, Security Assertion Markup Language (SAML) V2.0 Technical Overview, Committee Draft 02, http://docs.oasisopen. org/security/saml/ post2.0/sstc-saml-tech-overview-2.0 .html," March 2008.
43. J. Rosenberg and D. Remy, Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption. Pearson Higher Education, 2004.
44. S. Mandy. Drawbacks of the digital signature. http://computerfun4u.blogspot.com/2009/02/drawbacks-of-using-digital-signature.html.
45. P. Mell and T. Grance. The NIST Definition of Cloud Computing. Technical Report 800-145, National Institute of Standards and Technology, 2011.
46. Rackspace. OpenStack: The Open Source Cloud Operating System. http://www.openstack.org/software/
47. Darshan Tank, Akshai Aggarwal, and NirbhayChaubey, Security Analysis of OpenStack Keystone, International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS) Volume VI, Issue VI, June 2017 | ISSN 2278-2540
48. http://www.hytrust.com/cloud-sddc-study/
49. https://docs.openstack.org/security-guide/identity.html
50. Ishan GidwaniIshan, Dasrath Mane. Security Issues In OpenStack, International Journal of Computer Science and Information Technology Research, Vol. 3, Issue 2, pp: (1147-1158), Month: April - June 2015
51. Ericsson, Keystone Security GAP and Threat Identification (Quick Study), OpenStack Folsom Release, 2014
52. Ng, C.H., Ma, M., Wong, T.Y., Lee, P.P.C., Lui, J.C.S.: Live deduplication storage of virtual machine images in an open-source cloud. Proceedings of 2011, 12th ACM/IFIP/USENIX International Conference on Middleware. Pp. 81–100.
53. J.M. Alve, T.G. Rodrigues, "Multi-Factor Authentication with OpenID in Virtualized Environments," IEEE Latin America Transactions, Vol. 15, No. 3, pp. 528-533, March 2017
54. Rohit Ahuja, An identity is preserving access control scheme with flexible system privilege revocation in cloud computing, IEEE- 11th Asia Joint Conference on Information Security, pp. 39-47, 2016.
55. JaweherZouari, An Identity as a service framework for the cloud, IEEE Proceedings, Pp. 1-5, 2016.
56. QuratulainAlam, "Formal Verification of the X Darth Protocol," IEEE, pp.1-14, 2016
57. XiaoingJia, Efficient Revocable ID-based signature with cloud revocation server, IEEE Proceedings, Pp. 1-9, 2017
58. Mell PM, Grance T. SP 800-145. The NIST Definition of Cloud Computing. Tech. Rep., National Institute of Standards and Technology. MD: Gaithersburg; 2011.
59. Duncan A, et al. Cloud Computing: Insider Attacks on Virtual Machines during Migration. In: Trust, Security, and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference
60. Garkoti G, Peddoju S, Balasubramanian R. Detection of Insider Attacks in Cloud-Based e-Healthcare Environment. In: Information Technology (ICIT), 2014 International Conference on; 2014. p. 195–200. https://doi.org/10.1109/ICIT.2014.43.
61. Stolfo S, Salem M, Keromytis A. Fog Computing: Mitigating Insider Data Theft Attacks in the Cloud. In: Security and Privacy Workshops (SPW), 2012

IEEE Symposium on; 2012. p. 125–128. https://doi.org/10.1109/SPW.2012.19.

62. Cappelli DM, Moore AP, Trzeciak RF. The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crime, 1st ed. Addison-Wesley Professional; 2012.
63. Duncan A, Creese S, Goldsmith M. Insider Attacks in Cloud Computing. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on; 2012.p. 857–862. https://doi.org/ 10.1109/ TrustCom.2012.188.
64. Cole DE. Insider threats and the need for fast and directed responded. Tech. Rep.: SANS Institute InfoSec Reading Room; 2015.
65. Bailey C, Chadwick DW, de Lemos R. Self-adaptive federated authorization infrastructures. J Compute System Sci. 2014; 80(5):935–52. https:// dx.doi.org/10.1016/j.jcss.2014.02.003. http://www.sciencedirect.com/ science/article/PII /S0022000014000154
66. Pasquale L, et al. Securitas: A Tool for Engineering Adaptive Security. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. FSE ’12. New York: ACM; 2012. p. 19:1–19:4. https://doi.org/10.1145/2393596.2393618. https:// doi.acm.org/ 10.1145/2393596.2393618
67. Schmerl B, et al. Architecture-based Self-protection: Composing and Reasoning About Denial-of-service Mitigations. In: Proceedings of the 2014 Symposium and Bootcamp on the Science of Security. HotSoS ’14.
68. Yuan E, Esfahani N, Malek S. A systematic survey of self-protecting software systems. ACM Trans Auton Adapt Syst. 2014; 8(4):17:1–https://doi.org/10.1145/2555611. http://doi.acm.org/10.1145/2555611.
69. Schultz E, A framework for understanding and predicting insider attacks. Computer Security. 2002;21(6):526–31. https://doi.org/10.1016/S0167-4048(02)01009-X.
70. Cappelli DM, Moore AP, Trzeciak RF. The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crime, 1st ed. Addison-Wesley Professional; 2012.
71. George SilowashAM, Cappelli D, et al. Common sense guide to mitigating insider threats. Tech. Rep. CERT Carnegie Mellon; 2012.
72. Colwill C. Human factors in information security: The insider threat - who can you trust these days?.InfSecur Tech Rep. 2009;14(4):186–96. https:// doi.org/10.1016/j.istr.2010.04.004
73. Yaping Chi; Gefei Li; Ying Chen, Xiaohong Fan, Design and Implementation of OpenStack Cloud Platform Identity Management Scheme, Published in 2018 International Conference on Computer, Information and Telecommunication Systems (CITS)
74. Feng Dengguo, Zhang Min, Zhang Yan, et al. Research on cloud computing security [J].Journal of Software, 2011,22 (1): 71-83.
75. Yu Nenghai, HaoZhuo, Xu Jiajia, et al. Review of the progress of cloud security research [J].Journal of Electronics, 2013,41 (2): 371-381.
76. The Chi Yaping, Wang Huili, Yuan Ze Bo, and another authentication mechanism OpenStack Research and Improvement [J] Jilin University (Information Science), 2015 (11): 700-706.
77. JKRSastry, M TrinathBasu, Securing SAAS service under cloud computing-based multi-tenancy systems, Indonesian Journal of Electrical Engineering and Computer Science, Volume 13, Issue 1, Page 65-71, 2019
78. JKRSastry, M TrinathBasu, Securing Multi-tenancy systems through multi DB instances and multiple databases on different physical servers, International Journal of Electrical and Computer Engineering (IJECE), Volume 9, Issue 2, Pages 1385-1392, 2019
79. M.Trinath Basu1, Dr.JKRSastry, A full security included Cloud Computing Architecture, International Journal of Engineering & Technology, Volume 7, Issue 2.7, Page 807-812, 2018
80. JKRSastry, M TrinathBasu, Securing Multi-tenancy systems through user spaces defined within the database level, Jour of Adv Research in Dynamical & Control Systems, Volume 10, issue 7, Page 405-412, 2018
81. J. K. R. Sastry, K. Sai Abhigna, R. Samuel and D. B. K. Kamesh, Architectural models for fault tolerance within clouds at the infrastructure level, ARPN Journal of Engineering and Applied Sciences, VOL. 12, NO. 11, 2017, Pages 3463-3469
82. DBK Kamesh, JKRSastry, Ch. Devi Anusha, P. Padmini, G. Siva Anjaneyulu, Building Fault Tolerance within Clouds at Network Level, International Journal of Electrical and Computer Engineering (IJECE), Vol. 6, No. 4, pp. 1560~1569, 2016
83. S. L. SUSHMITHA, Dr. D. B. K. J.K. R. SASTRY, V. V. N. SRI RAVALI, Y.SAI KRISHNA REDDY, building fault tolerance within clouds for providing uninterrupted software as service, Journal of Theoretical and Applied Information Technology, Vol.88. No.1, Pages 65-76, 2016
84. NVSPavan Kumar, Dr.JKRSastry, Dr. K Raja Sekhara Rao, Mining Distributed Databases for Negative Associations from Regular and Frequent Patterns, International Journal of Advanced Trends, Volume 8, Issue 4, Pages 1440-1463, 2019
85. NVSPavan Kumar, Dr.JKRSastry, Dr. K Raja Sekhara Rao, On Incremental mining Databases for Regular and Frequent Patterns, International Journal of Emerging Trends and engineering research, Volume 7, Issue 9, Pages 291-305, 2019 https://doi.org/10.30534/ijeter/2019/12792019
86. NVSPavan Kumar, Dr.JKRSastry, Dr. K Raja Sekhara Rao, Mining Negative Frequent regular

Itemsets from Data Streams, International Journal of Emerging Trends and engineering research, Volume 7, Issue 8, Pages 85-98, 2019 https://doi.org/10.30534/ijeter/2019/02782019

87. M. TrinathBasu, JKRSastry, Improving the Open Stack Authentication system through federation with JASON Tokens, International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, Issue 6, Pages 3596-3614,2019. https://doi.org/10.30534/ijatcse/2019/143862019

Table 4: Steps to configure Keystone steps

Config. Serial	Configuration step	Configuration Command Strings
Enable command line access to keystone v3		
1	Create an environment variable called "overcloudrc-v3."	$ cp overclouds overclouds-v3
2	Change the authentication URL to refer to Version 3 of the authentication system	Change OS_AUTH_URLfrom *v2.0* to *v3*. For example: export OS_AUTH_URL=https://controllerIP:5000/v3/
3	Export the Environment variables related to OS-Identity-API-Version, Project-domain-name, and OS-user-domain-name	export OS_IDENTITY_API_VERSION=3 export PROJECT_DOMAIN_NAME=Default export OS_USER_DOMAIN_NAME=Default
4	The overcloudrc-v3 is enabled to contain the command lines entered in the session.	$ source overcloudrc-v3
Configure the controller		Most of the services of the open stack are now available as containers which include keystone, cinder, nova etc.No changes to the configuration files be done which are situated on physical servers as the container services do not access these files No changes to the configuration files contained within the containers as the changes made ignored every time a restart takes place Any changes to the configuration files retained when changes made to configuration files used to generate the containers
If keystone is running on the controller node then		
5	Configure SELinux:	# setsebool -P authlogin_nsswitch_use_ldap=on
6	Create the domainsdirectory	# mkdir /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/ # chown 42425:42425 /var/lib/config-data/puppet- generated /keystone /etc /keystone/ domains/
7	Configure keystone to use multiple back ends:	# crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf identity domain_specific_drivers_enabled true # crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf identity domain_config_dir /etc/keystone/domains # crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf assignment driver sql

8	To add commands to /etc/openstack-dashboard/local_settings: to enable multiple domains in the dashboard	OPENSTACK_API_VERSIONS =
		{
		"identity": 3
		}
		OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
		OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'
		Restart httpdto apply the settings:-
		# systemctl restart httpd

Configure an additional back-end: called LAB, which is the NetBIOS name to use as the Identity Service domain.		
9	The ADDS system must recognize the domain name of the Keystone. To create A domain name called LAB within OpenStack which will be the NetBIOS name for OpenStack	# OpenStack domain create LAB

Config. Serial	Configuration step	Configuration Command Strings

10	Add LDAP settings to Keystone.LAB configuration file. This setting will create an ADDS backend.
	
	/var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/keystone.LAB.conf
	
	The settings shown on the right needs to be changed to meet the customer local ADDS environment
		url = ldaps://addc.lab.local:636
		user = CN=svc-ldap,OU=labUsers,DC=lab,DC=local password = RedactedComplexPassword
		suffix = DC=lab,DC=local
		user_tree_dn = OU=labUsers,DC=lab,DC=local
		user_objectclass = person
		user_filter = (|(memberOf=cn=grp-openstack,OU=labUsers,DC=lab,DC=local)(memberOf=cn=grp-openstack-admin,OU=labUsers,DC=lab,DC=local)(memberOf=memberOf=cn=grp-openstack-demo,OU=labUsers,DC=lab,DC=local))
		user_id_attribute = sAMAccountNameuser_name_attribute = sAMAccountNameuser_mail_attribute = mail user_pass_attribute = user_enabled_attribute = userAccountControluser_enabled_mask = 2
		user_enabled_default = 512
		user_attribute_ignore = password,tenant_id,tenantsuser_allow_create = False
		user_allow_update =False
		user_allow_delete =False
		group_objectclass =group
		group_tree_dn = OU=labUsers,DC=lab,DC=local
		group_filter = (CN=grp-openstack*)
		group_id_attribute = cngroup_name_attribute = name group_allow_create = False
		group_allow_update =False
		group_allow_delete =False
		use_tls =False
		tls_cacertfile =
		/etc/ssl/certs/addc.lab.local.crt query_scope = sub
		chase_referrals = false
		[identity] driver = idap

11	To make the keystone user the owner of the configuration file	# chown 42425:42425 /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/keystone.LAB.conf
12	To rerun the keystone component to effect the changes	# sudodocker exec -it keystone pkill -HUP -f keystone
To allocate the domain access to the admin user.		
13	To get the ID of LABdomain	# OpenStack domain show LAB
14	To get ID of *admin* user:	# openstack user list --domain default | grep admin | 3d75388d351846c6a880e53b2508172a | admin |
15	To get the ID of *admin* role	# OpenStack role list
16	To assign admin role of keystone "LABdomain" to the admin user using the Domain names and admin IDs returned in the earlier steps	# openstack role add --domain 6800b0496429431ab1c4efbb3fe810d4 --user 3d75388d351846c6a880e53b2508172a 785c70b150ee4c778fe4de088070b4cf
17	To view the list of users contained in domain LAB and default domains	# OpenStack user list --domain LAB
18	To view the account names contained in the database related to identity services	# OpenStack user list --domain default
Configure Compute to use keystone v3		

Config. Serial	Configuration step	Configuration Command Strings
19	Adjust the keystone_authtokenvalue on the controller and compute node	
20	Set the keystone_authtokenvalue on Compute nodes	# crudini --set /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf keystone_authtoken auth_version v3
21	Set the keystone_authtokenvalue on the controller:	# crudini --set /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf keystone_authtoken auth_version v3
22	Re-run the controller component to effect the changes	# systemctl restart openstack-nova-api.service openstack-nova-cert.service openstack-nova-conductor.service openstack-nova-consoleauth.service openstack-nova-novncproxy.service openstack-nova-scheduler.service # sudodocker exec -it keystone pkill -HUP -f keystone
23	Re-run the compute component to effect the changes	# systemctl restart OpenStacknova-compute.service
Configure Block Storage to use keystone v3		
24	In */etc/cinder/cinder.conf*:	[keystone_authtoken] auth_uri = https://controllerIP:5000/v3 auth_version = v3 auth_uri - replace controllerIPwith the IP address of the controller. If your deployment has more than one controller, you should use the keystone endpoint VIP instead of the controller IP.
25	Re-run the cinder-API-on on all the nodes	# systemctl restart OpenStack-cinder-API
26	Re-run cinder-scheduleron all nodes	# systemctl restart openstack-cinder-scheduler
27	Re-run cinder-volumeon the main controller	# pcs resource restart openstack-cinder-volume
28	To authorize the active director groups to have access to projects to eliminate the requirement of adding a role to each user to have access to the projects	
29	Administrator to Complete the steps in the right cell	Create the groups named grp-OpenStack-admin Active Directory, grp-OpenStack-demo Add Active Directory users to the above groups, Add Active Directory users to the grp-OpenStack group.
30	The following steps assign roles to ADDS groups so that the users will have access to OpenStack resources.	
	1. Get the list od ADDS groups	# OpenStack group list --domain LAB
	2. Get the list of ADDS roles	# OpenStack role list
	3. Assign the access to the projects to the Active Directory groups	# openstack role add --project demo --group d971bb3bd5e64a454cbd0cc7af4c0773e78d61b5f81321809f8323216938cae8 _member_

Allow Active Directory users to access Projects		
Config. Serial	Configuration step	Configuration Command Strings
31	To grant permissions to the grp-OpenStack group so that the Gropumemebers can log into a project through a dashboard.	
	Find the list of AD users:	# OpenStack user list --domain LAB
	Find the list of roles:	# OpenStack role list
	To assign access to the projects to the users by adding one more role to the users	# openstack role add --project demo --user 1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac41809 1e_member_ Or, if you want user1to be an administrative user of the demoproject, you add them to the adminrole: # openstack role add --project demo --user 1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac41809 1e admin
32	To assign access rights to the Domain Tab so that administrator can use the same by adding an admin role	
	1. Find the adminuser's UUID:	$ openstack user list | grep admin | a6a8adb6356f4a879f079485dad1321b | admin |
	2.Add the adminrole in the defaultdomain to the adminuser:	$ openstack role add --domain default --user a6a8adb6356f4a879f079485dad1321b admin
33	The user projects created with either the default domain name or the keystone domain name. The default domain name is an internal domain name used to manage the service accounts	
34	A domain field is created within the dashboard when multiple domains configured in the Identity service. User must enter an area that belongs to them using their login credentials	
35	Changes to the command line For specific commands, you might need to specify the applicable domain. For example, appending -- domain Labin this command returns users in the LAB domain (that are members of the *grp-OpenStack* group): # OpenStack user list --domain LAB. Appending --domain Defaultreturns the built-in keystone accounts: # OpenStack user list --domain Default	
36	The ADDS system integration with the keystone tested by using the dashboard features. The testing carried using the following steps Create a User in Active Directory Add the user to grp-OpenStack adds group Add user _member_role related to demotenant Login to Dashboard using new user credentials Click on the TABS Create a test instance through Dashboard. All the TABS shall have the details related to the users, their roles and the projects that they can access	

Config. Serial	Configuration step	Configuration Command Strings
37	Activate keystone v3 for achieving high availability through configuring the same to allow configuring multiple domain controllers. Version v1 and V2 are good enough with a single domain controller.	
	Add a second server to the urlentry.	url = ldaps://addc.lab.local,ldaps://addc2.lab.local Keystone will send all queries to the domain controller by changing the URL set in the keystone — Lbconfig file.
	Set the network timeout in /etc/OpenLDAP/ldap.conf	NETWORK_TIMEOUT 2
	Create an RC file for a non-admin user	$ cat overcloudrc-v3-user1 # Clear any old environment that may conflict. for key in $(set | awk '{FS="="} /^OS_/ {print $1}'); do unset $key ; done export OS_USERNAME=user1 export NOVA_VERSION=1.1 export OS_PROJECT_NAME=demo
	Create an RC file for a non-admin user	export OS_PASSWORD=RedactedComplexPassword export OS_NO_CACHE=True export COMPUTE_API_VERSION=1.1 export no_proxy=,10.0.0.5,192.168.2.11 export OS_CLOUDNAME=overcloud export OS_AUTH_URL=https://10.0.0.5:5000/v3 export OS_AUTH_TYPE=password export PYTHONWARNINGS="ignore:Certificate has no, ignore:A true SSLContext object is not available" export OS_IDENTITY_API_VERSION=3 export OS_PROJECT_DOMAIN_NAME=Default export OS_USER_DOMAIN_NAME=LAB

The keystone components installed in the controller nodes also need to be configured. The description of the parameters and the values to set within the configuration described in **Table 5**

Table 5 :Description of the parameters expressed in the configuration related to KEYSTONE installed on the Controller node

Parameter Serial	Parameter	Description
1	URL	The AD Domain Controller to use for authentication. Uses LDAPS port 636.
2	user	The *Distinguished Name* of an AD accounts to use for LDAP queries. For example, you can locate the *DistinguishedName* value of the *svc-LDAP* account in AD using Get-AD user svc-LDAP | select Distinguished Name
3	password	The plaintext password of the AD account used above.
4	suffix	The *Distinguished Name* of your AD domain. You can locate this value using Get- ADDomain | select Distinguished Name
5	user_tree_dn	The *Organizational Unit* (OU) that contains the OpenStack accounts.
6	user_objectclass	DefinesthetypeofLDAPuser.ForAD,use the persontype.
7	user_filter	Filters the users presented to Identity Service. As a result, only members of the grp- OpenStack group can have permissions defined in Identity Service. This value requires the full *Distinguished Name* of the group: Get-ADGroup grp-OpenStack | select DistinguishedName
8	user_id_attribute	Maps the AD value to use for user IDs.
9	user_name_attribute	Maps the AD value to use for *names*.
10	user_mail_attribute	MapstheADvaluetouseforuseremail addresses.
11	user_pass_attribute	Leave this value blank.
12	user_enabled_attribute	The AD setting that validates whether the account is enabled.
13	user_enabled_mask	Defines the value to check to determine whether an account is enabled. Used when Booleans returned.
14	user_enabled_default	TheADvaluethatindicatesthatanaccountis enabled.
15	user_attribute_ignore	Defines user attributes that Identity Service should disregard.
16	user_allow_create	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
17	user_allow_update	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
18	user_allow_delete	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
19	group_objectclass	Maps the AD value to use for *groups*.
20	group_tree_dn	The *Organizational Unit* (OU) that contains the user groups.
21	group_filter	Filters the groups presented to Identity Service.
22	group_id_attribute	Maps the AD value to use for group IDs.
23	group_name_attribute	Maps the AD value to use for group names.
24	group_allow_create	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
25	group_allow_update	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
26	group_allow_delete	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
27	use_tls	Defines whether TLS used. TLS needs to be disabled if you are encrypting with LDAPS rather thanSTARTTLS.
28	tls_cacertfile	Specifies the path to the *.crt*certificate file.

Parameter Serial	Parameter	Description
29	query_scope	Configures Identity Service to also search within nested child OUs when locating users that are members of the grp-OpenStack group.
30	chase_referrals	Set to false, this setting prevents python- LDAP from chasing all referrals with anonymous access.
31	URL	The AD Domain Controller to use for authentication using LDAPS port 636.
32	user	The*DistinguishedName*ofanADaccountsto useforLDAPqueries.Forexample,youcan locatethe*DistinguishedName*valueofthe *svc-ldap*accountinADusingGet-ADuser svc-ldap | select DistinguishedName
33	password	The plaintext password of the AD account used above.
34	suffix	The *Distinguished Name* of your AD domain. You can locate this value using Get- ADDomain | select DistinguishedName
35	user_tree_dn	The *Organizational Unit* (OU) that contains the OpenStack accounts.
36	user_objectclass	DefinesthetypeofLDAPuser.ForAD,use the persontype.
37	user_filter	Filters the users presented to Identity Service. As a result, only members of the grp- OpenStack group can have permissions defined in Identity Service. This value requires the full *Distinguished Name* of the group: Get-ADGroup grp-OpenStack | select DistinguishedName
38	user_id_attribute	Maps the AD value to use for user IDs.
39	user_name_attribute	Maps the AD value to use for *names.*
40	user_mail_attribute	MapstheADvaluetouseforuseremail addresses.
41	user_pass_attribute	Leave this value blank.
42	user_enabled_attribute	The AD setting that validates whether the account is enabled.
43	user_enabled_mask	Defines the value to check to determine whether an account is enabled. Used when Booleans not returned.
44	user_enabled_default	TheADvaluethatindicatesthatanaccountis enabled.
45	user_attribute_ignore	Defines user attributes that Identity Service should disregard.
46	user_allow_create	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
47	user_allow_update	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
48	user_allow_delete	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
49	group_objectclass	Maps the AD value to use for *groups.*
50	group_tree_dn	The *Organizational Unit* (OU) that contains the user groups.
51	group_filter	Filters the groups presented to Identity Service.
52	group_id_attribute	Maps the AD value to use for group IDs.
53	group_name_attribute	Maps the AD value to use for group names.
54	group_allow_create	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
55	group_allow_update	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.

Parameter Serial	Parameter	Description
56	group_allow_delete	SetthisvaluetoFalse,askeystoneonly requires read-onlyaccess.
57	use_tls	Defines whether TLS used. TLS needs to be disabled if you are encrypting with LDAPS rather thanSTARTTLS.
58	tls_cacertfile	Specifies the path to the .*crt*certificate file.
59	query_scope	Configures Identity Service to also search within nested child OUs when locating users that are members of the grp-OpenStack group.
60	chase_referrals	Set to false, this setting prevents python- LDAP from chasing all referrals with anonymous access.