Security Analysis of OpenStack Keystone

Darshan Tank[1], Akshai Aggarwal[2] and Nirbhay Chaubey[3]

[1]Department of Information Technology, L. E. College, Morbi, Gujarat, India
[2]Former Vice Chancellor, Gujarat Technological University, Ahmedabad, Gujarat, India
[3]Associate Professor, S.S. Agrawal Institute of Computer Science, Navsari, Gujarat, India

***Abstract* – OpenStack is one of the most used cloud management software today. OpenStack is a free and open-source software platform for cloud computing, mostly deployed as an infrastructure-as-a-service (IaaS). We have selected OpenStack as one of the underlying infrastructure service layer tool because OpenStack allows us to modify our cloud to fit into custom infrastructures that may be necessary for scientific research. Keystone is the key component of OpenStack responsible for authentication and authorization.**

As the distributed nature of OpenStack services, Keystone plays a major role in binding all of the projects together. Not only do we have to be wary of the services that connect to Keystone but also have to be cautious of the kinds of input and data we give to Keystone from the external sources. The security and protection of the identity and token repository for OpenStack needs to be the most protected component within cloud infrastructure.

As the threat surface in cloud changes constantly, security is one of the biggest concerns for any cloud solutions. Deploying the open source cloud raises additional challenges since the intruders have access to the cloud source code and can assess its vulnerabilities. In this paper, we systematically analyze the security aspects of the OpenStack keystone and explore the threat model against, and security requirements of, OpenStack keystone. We then propose a new authentication model using the RESTful API to satisfy the security needs of OpenStack Keystone. The proposed authentication model can accommodate a diverse set of security services.

***Index Terms* – Authentication, Cloud Computing, Keystone, OpenStack, Security, Vulnerabilities**

I. INTRODUCTION

Cloud computing is getting widely deployed and is changing the landscape how Information Technology (IT) will serve the needs of government, enterprises, society and home users. Organizations around the world are choosing cloud technologies for an optimal mix of reliability, flexibility and value. There are many cloud service providers (CSPs) offering various cloud solutions, such as Amazon Web Services, Microsoft Azure, Google Cloud Platform, IBM Cloud, Rackspace, VMware's vCloud, Red Hat, Oracle Cloud, Verizon Cloud, Navisite, Salesforce's Sales Cloud and many others [1].

A key question for modern IT is “how are security, compliance and privacy affected by cloud?” According to the 2016 State of the Cloud Survey, ongoing security risks are major concerns for organizations evaluating cloud computing and are seen as the primary barrier for cloud migration [2].

In recent years there has been increased the use of open source cloud framework. OpenStack, Eucalyptus, CloudStack, and OpenNebula are the most common open source cloud frameworks that allow the customers to build their own private IaaS (Infrastructure as a Service) cloud. OpenStack has became a preferred open source cloud framework solution for building private and public clouds within a large number of companies and the research community [7].

OpenStack has increased in popularity in the short amount of time it has been in existence. OpenStack as a platform is made up of multiple projects, which are all adopted at different rates. The core projects, including the Nova compute, Keystone identity and Horizon dashboard projects, are almost universally installed as part OpenStack platform deployments.

According to a 451 Group study, commissioned by the OpenStack Foundation, organizations of all sizes are deploying OpenStack. Many organizations use OpenStack in production. IBM Blue Box, for example, offers Private Cloud as a Service (PCaaS) using OpenStack, hosting thousands of customers across 17 datacenters covering North America, South America, Europe, the Middle East and Asia.

Identity service (keystone) provides identity, token, catalog, and policy services for use specifically by services in the OpenStack family. Identity service is organized as a group of internal services exposed on one or many endpoints. Many of these services are used in a combined fashion by the frontend, for example an authenticate call will validate user/project credentials with the identity service and, upon success, create and return a token with the token service [3].

The rest of the paper is organized as follows: - Section 2 briefly reviews related prior work. Section 3 describes OpenStack - open source cloud software. Section 4 presents and elaborates exploits on OpenStack. Section 5 explores key points of OpenStack identity service (keystone). Section 6 outlines keystone architecture. Section 7 analyzed security vulnerabilities of keystone. Section 8 illustrates the experimental setup. Section 9 discusses our plan of work. Finally, Section 10 concludes the paper followed by future scope.

II. RELATED WORK

This section reviews previous contributions on security issues in OpenStack.

Author in [4], analyzed security issues in open-source cloud computing project - OpenStack Object Storage. The author highlights which security issues should be taken care of when using cloud services. He reviewed the documents that were created by a Cloud Security Alliance, an organization consisting of industry representatives, and two governmental institutions: European Network and Information Security Agency, and National Institute of Standards and Technology. The study compiles a list of security issues to be used when evaluating security of OpenStack cloud solution.

Ishan GidwaniIshan et. al [5] discuss issues that arise with the deployment model of cloud computing; in particular, this study focuses on OpenStack security issues and threats. The authors argue that OpenStack does not support minimum password complexity requirements and passwords are stored in plain text format. They performed penetration test against OpenStack with all well-known and industry leading security auditing tools. The authors come up with the conclusion that OpenStack is a secure cloud platform, however keeping OpenStack secure for everyone is a challenge in itself, as new vulnerabilities are discovered with time.

B. Cui and T. Xi in [6] proposed an enhanced secure mechanism of Keystone authentication system for OpenStack. The authors discuss various security vulnerabilities and suggested security services based on the security features of Keystone. The study provides the details of the design and implementation of the new authentication model, and analyzes the security feature of the model with the testing of its ability of defending against several major security threats.

According to S. Ristov, et al. [7], both the tenants and the OpenStack cloud provider are vulnerable. The study assesses the security vulnerabilities of the cloud service provider, tenants and the cloud management application. The experiments addressed the security vulnerabilities of OpenStack cloud server node, four virtual machine instances with different operating systems and the OpenStack Dashboard web management interface. The authors argue that the tenants' vulnerabilities can be secured, while OpenStack software framework must be secured with new patches.

In [8], the author state that Keystone, the centralized identity and access management system for OpenStack, provides basic functionalities for authentication and authorization. In this analysis, he found weaknesses / lack of feature support in many areas including access control mechanisms, authentication attribute provisioning, policy provisioning mechanism, and auditing mechanisms. The threat analysis has identified possible threats against interfaces, components and subcomponents of Keystone.

III. OVERVIEW OF OPENSTACK

OpenStack is a free and open-source cloud computing software platform. User primarily deploy it as an infrastructure as a service solution. Until now, OpenStack has released 10 versions from A to O; functions of the system have been continuously improved during the evolution of the version. As an open-source cloud platform, its safety is relatively high, but there are still bugs and vulnerabilities which have become one of the most important points need to be considered in the evolution of its development. Despite the commercial clouds and their services, there are many open source cloud solutions that provide the customers to develop their own private cloud, especially IaaS cloud service layer, such as well known OpenStack, Eucalyptus, OpenNebula, and CloudStack. There are open cloud-computing research testbed designed to support research into the design, provisioning, and management of services at a multi-datacenter scale, such as Opencyrrus [9].

Almost all open source cloud solutions have the same architecture. Eucalyptus has a similar architecture as OpenStack [10]. Each solution consists of several different parts and all have two main parts: The cloud controller and Nodes. The former controls the system, network, schedules the instances and is the administrator interface. The latter runs the instances of the virtual machines and uses the available hardware resources. Another important part of each solution is the network management, i.e. delimiting and managing the public and private network.

Almost all open source cloud solutions provide interfaces to commercial cloud services Amazon's EC2 and S3, and Google's App Engine. They also allow users to build not only their own private cloud, but also a hybrid cloud. The open source solutions give the freedom to customer to choose the hardware and software vendor. All main hypervisors and operating systems are supported.

In this paper we focus on OpenStack Cloud Software. It is a open source cloud software that builds both public and private clouds. OpenStack: The Ocata Release is the latest (the fifteenth) released version.

IV. EXPLOITS ON OPENSTACK

OpenStack is a diverse application made up of smaller components that provide individual functionality and serve a defined purpose. Therefore exploiting OpenStack in general means attacking these individual components with tools like MetaSploit, Hydra etc. Figure 1 below shows the possible exploits that can be performed on OpenStack.

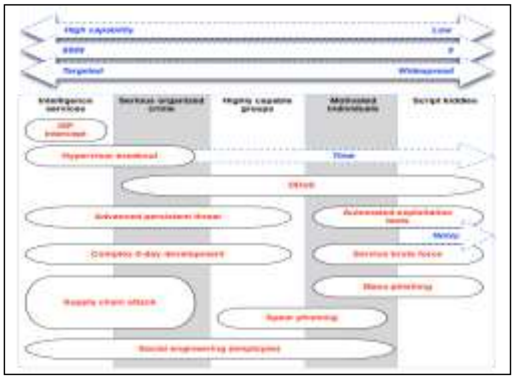


Figure 1: Exploits on OpenStack

- *ISP Interception*

ISP interception means intercepting traffic between the internet service provider and Open Stack, it's mostly done by intelligence agencies or by the law enforcement peoples in order to get personal information. This type of attack can be very fruitful if the data sent across the network is not encrypted. Sensitive information like security credentials can be easily compromised if the data is unencrypted.

- *Hypervisor Breakout*

Hypervisor breakout is one of the most intensely researched topics. This is due to the fact that, once a hypervisor is compromised then the whole system goes in the hands of the attacker.

- *DDOS Attack*

A Distributed Denial of Service attack can be executed by sending to much traffic on a server. This server however can handle only handle a set amount of requests, in results it either a complete failure or the slow down the services. This type of attack can be used by any activist group. DDOS is one of the most dangerous forms of attacks to the modern world as it cannot be traced easily.

- *Complex 0-Day Development*

Complex 0-Day development involves the process in which some highly capable group of hackers find the 0-day bug in the product and then develop the 0-day exploit to exploit the vulnerability in that product. It can be funded by the organizations or intelligence agencies to attack on certain targets.

- *Brute Force*

Brute force is an attack in which the attacker tries to get access to a vulnerable system with different combinations of a username and password. It can be done by individuals. Brute Force attacks have been mitigated with the latest developments in the security industry. There have been many cryptographic algorithms and captcha developments to stop it.

- *Phishing*

In phishing an attacker sends a fake page which looks real and when the victim enters the credentials the attacker easily get those.

- *Social engineering attack*

In a social engineering attack, an attacker uses human interaction (social skills) to obtain or compromise information about an organization or its computer systems.

V. OPENSTACK IDENTITY SERVICE (KEYSTONE)

Keystone is the identity service used by OpenStack for authentication and high-level authorization. It currently supports token-based authentication and user-service authorization [11]. Keystone is the centralized identity and access management component of OpenStack. This is designed as a shim layer on top of pluggable data store (SQL, LDAP). It consists of several services like Identity, Token, Catalog and Policy etc. [12].

5.1 Authentication

Authentication is the process of verifying the identity of a user by obtaining some sort of credentials and using those credentials to verify the user's identity. Authentication is an integral part of any real world OpenStack deployment. The OpenStack Identity service (keystone) supports multiple methods of authentication, including user name & password, LDAP, and external authentication methods. Using multi-factor authentication helps to reduce the risks of brute force attacks, social engineering and spear and mass phishing attacks [3]. Keystone supports, and many users implement, federated identity to establish trusts between identity providers and the services provided by an OpenStack cloud. Server may also enforce client-side authentication using certificates.

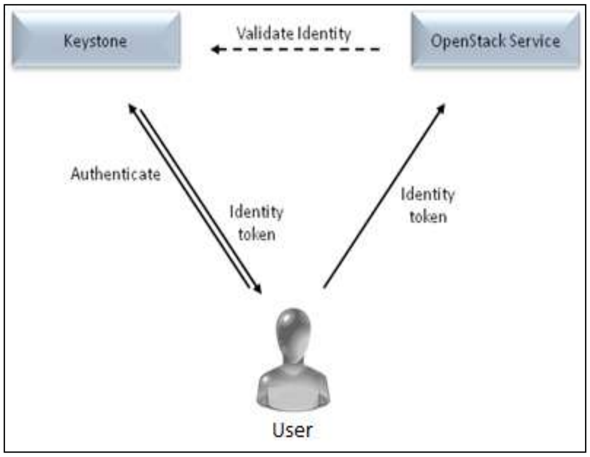


Figure 2: Keystone authentication process

Each service is written as a WSGI application. Keystone supports two type of auth mechanism: EC2 style and OS style. In Keystone, Authentication is centralized. The authorization part is still evolving (currently, supports rule based authorization controlled by each service). Centralized roles are supported but they act as a Meta data for the users/tenants. The security questions one should consider are what are my assets? What am I trying to protect? Is my particular deployment secure? And where am I likely to be attacked?

5.1.1. Authentication methods

A. Internally implemented authentication methods

The Identity service can store user credentials in an SQL Database, or may use an LDAP-compliant directory server. The Identity database may be separate from databases used by other OpenStack services to reduce the risk of a compromise of the stored credentials [3].

B. External authentication methods

Organizations may desire to implement external authentication for compatibility with existing authentication services or to enforce stronger authentication policy requirements. Although passwords are the most common form of authentication, they can be compromised through numerous methods, including keystroke logging and password compromise. External authentication services can provide alternative forms of authentication that minimize the risk from weak passwords. These include:

a) Password policy enforcement

Requires user passwords to conform to minimum standards for length, diversity of characters, expiration, or failed login attempts. In an external authentication scenario this would be the password policy on the original identity store.

b) Multi-factor authentication

The authentication service requires the user to provide information based on something they have, such as a one-time password token or X.509 certificate, and something they know, such as a password.

c) Kerberos

A mutual authentication network protocol using ‘tickets’ to secure communication between client and server. The Kerberos ticket-granting ticket can be used to securely provide tickets for a given service.

5.2 Authorization

Authorization is the process of allowing authenticated users to access the resources by checking whether the user has access rights to the system. Authorization helps you to control access rights by granting or denying specific permissions to an authenticated user. The Identity service supports the notion of groups and roles. Users belong to groups while a group has a list of roles. OpenStack services reference the roles of the user attempting to access the service. The OpenStack policy enforcer middleware takes into consideration the policy rule associated with each resource then the user’s group/roles and association to determine if access is allowed to the requested resource [3].

5.3 Identity service sample configuration files

We can find the following files in the /etc/keystone directory.

- keystone.conf
  The Identity service is configured in the /etc/keystone/keystone.conf file. We can use the keystone.conf file to configure most Identity service options.
- keystone-paste.ini
  We can use the keystone-paste.ini file to configure the Web Service Gateway Interface (WSGI) middleware pipeline for the Identity service.
- logging.conf
  We can specify a special logging configuration file in the keystone.conf configuration file.
- policy.json
  We can use the policy.json file to define additional access controls that apply to the Identity service

Table 1. Encryption algorithms used in OpenStack Keystone

Sr No.	Algorithm	Purpose	Configurable	Implementation	Details
1.	AES	Memcache backend encryption	No	PyCrypto	Optionally used for encrypting the token backend
2.	RSA	PKI token signing	Yes	OpenSSL	2048, sha1 defaults Configurable via openssl.conf Keys/Certs can be created outside of Keystone and dropped into place

Table 2. Hashing algorithms used in OpenStack Keystone

Sr No.	Algorithm	Purpose	Configurable	Implementation	Details
1.	md5	Token hashing	No	Hashlib	Hash is used as an internal identifier in the token backend. The data being hashed is the entire cryptographically signed token (which uses the configured signing key). The chance for collisions should be low.
2.	sha1	S3 credentials	No	Hashlib	Used for signature validation of S3 credentials. Required for S3 compatibility, so it can't be configurable.
3.	sha1	OAuth1	No	Oauthlib	Used for signature validation of OAuth1 tokens. Keystone only uses the HMAC-SHA1 signature for OAuth1 tokens (as described in RFC 5849). OAuth support can be disabled. Likely uses hashlib for the actual algorithm.
4.	sha256	EC2 tokens	No	Hashlib	Required for EC2 compatibility, so it can't be configurable.
5.	sha384	Memcache signing	No	Hashlib	Used for signing and verification when memcache encryption is enabled.
6.	sha512	Password hashing	No	PassLib	The algorithm is non-configurable, but the number of rounds is configurable via CONF.crypt_strength (default=40000).

MD5 is a weak and depreciated hashing algorithm. It can be cracked using brute force attack. Identity tokens are sensitive and need to be protected with a stronger hashing algorithm to prevent unauthorized disclosure and subsequent access. The appropriate choice is SHA-256 or SHA-512. They differ in the word size; SHA-256 uses 32-bit words where SHA-512 uses 64-bit words.

VI. KEYSTONE ARCHITECTURE

Keystone is organized as a group of internal services exposed on one or many endpoints. Many of these services are used in a combined fashion by the frontend, for example an authenticate call will validate user/project credentials with the Identity service and, upon success, create and return a token with the Token service [12].

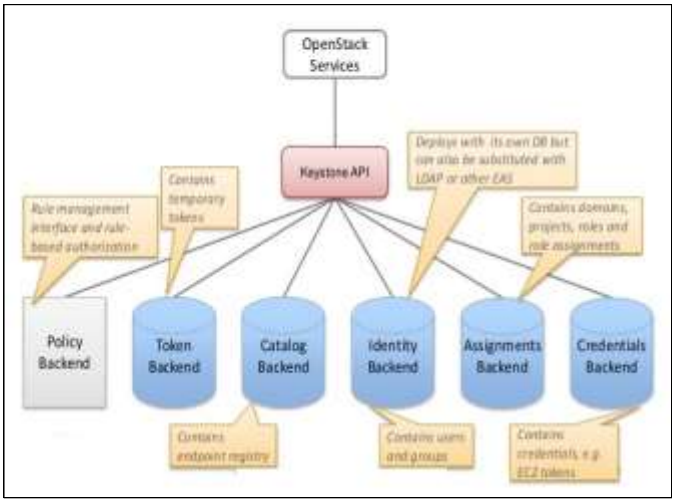

OpenStack Services
Keystone API
Rule management interface and rule-based authorization
Contains temporary tokens
Deploys with its own DB but can also be substituted with LDAP or other EAS
Contains domains, projects, roles and role assignments
Policy Backend
Token Backend
Catalog Backend
Identity Backend
Assignments Backend
Credentials Backend
Contains endpoint registry
Contains users and groups
Contains credentials, e.g. EC2 tokens

Figure 3: Keystone-architecture

- *Identity*

  The Identity service provides auth credential validation and data about users and groups. In the basic case all this data is managed by the service, allowing the service to manage all the CRUD associated with the data. In other cases, data is managed from an authoritative backend service.

- *Users*

  Users represent an individual API consumer. A user itself must be owned by a specific domain, and hence all user names are not globally unique, but only unique to their domain.

- *Groups*

  Groups are a container representing a collection of users. A group itself must be owned by a specific domain, and hence all group names are not globally unique, but only unique to their domain.

- *Resource*

  The Resource service provides data about projects and domains.

- *Projects*

  Projects (known as Tenants) represent the base unit of ownership in OpenStack, in that all resources in OpenStack should be owned by a specific project. A project itself must be owned by a specific domain, and hence all project names are not globally unique, but unique to their domain.

- *Domains*

Domains are a high-level container for projects, users and groups. Each is owned by exactly one domain. Each domain defines a namespace where an API-visible name attribute exists. Due to their container architecture, domains may be used as a way to delegate management of OpenStack resources.

- *Assignment*

  The Assignment service provides data about roles and role assignments.

- *Roles*

  Roles dictate the level of authorization the end user can obtain. Roles can be granted at either the domain or project level. Role can be assigned to the individual user or at the group level.

- *Role Assignments*

  A 3-tuple that has a role, a resource and an identity.

- *Token*

  The Token service validates and manages tokens used for authenticating requests once a user's credentials have already been verified.

- *Catalog*

  The Catalog service provides an endpoint registry used for endpoint discovery.

- *Policy*

  The Policy service provides a rule-based authorization engine and the associated rule management interface.

A generic Identity and Access Management (IAM) system consists of an access management service which controls the authentication and authorization; an identity management service for administering the accounts, a provisioning service for propagating accounts and privileges, a data store for storing user and policy information. The core of an IAM system is authentication and authorization. Authentication is the process of verifying credentials of a subject/resource/entity while authorization grants a subject to perform an action on a target resource. A generic IAM system has the following security objectives:

- — Authentication of users and administrators for each resource request
- — Access control of resources i.e., only authorized users are allowed to access resources
- — Recording of all security operations and transactions
- — Account provisioning is only performed for entitled users
- — Correct association of account attributes

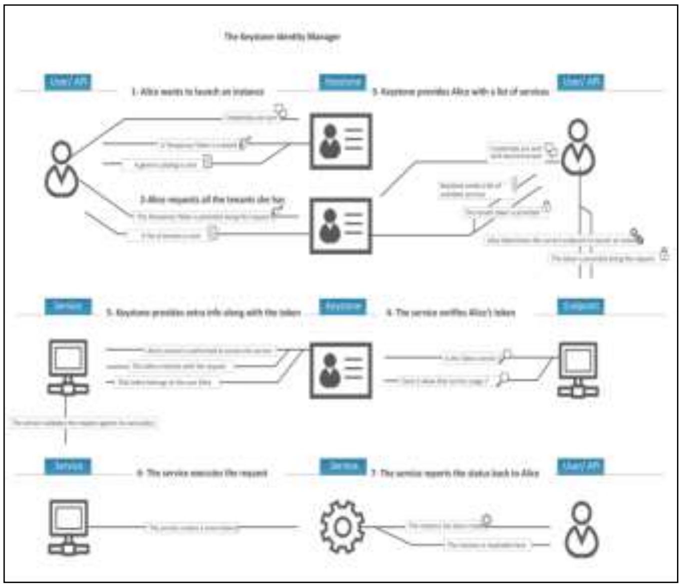

The Keystone Identity Manager
User/ API
1. Alice wants to launch an instance
Keystone
3. Keystone provides Alice with a list of services
User/ API
2. Alice requests all the tenants she has
Service
5. Keystone provides extra info along with the token
Keystone
4. The service verifies Alice's token
Endpoint
Service
6. The service executes the request
Service
7. The service reports the status back to Alice
User/ API

Figure 4: Keystone-workflow

The Identity service generates authentication tokens that permit access to the OpenStack services REST APIs. Clients obtain this token and the URL endpoints for other service APIs by supplying their valid credentials to the authentication service. Each time you make a REST API request to an OpenStack service, you supply your authentication token in the X-Auth-Token request header. Like most OpenStack projects, OpenStack Identity protects its APIs by defining policy rules based on a role-based access control (RBAC) approach. The Identity service configuration file sets the name and location of a JSON policy file that stores these rules [13].

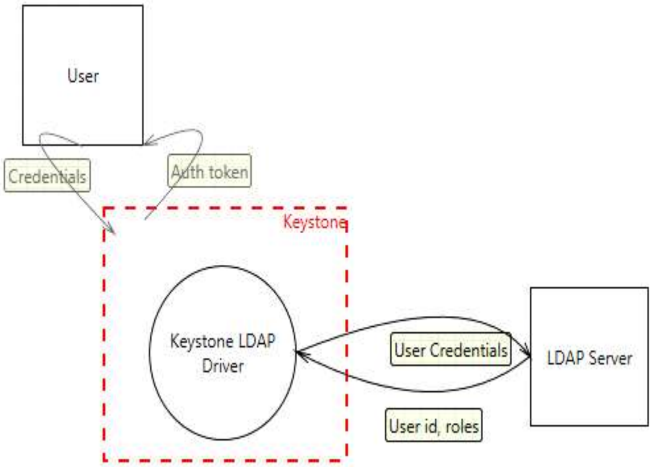


Figure 5: Keystone threat modeling

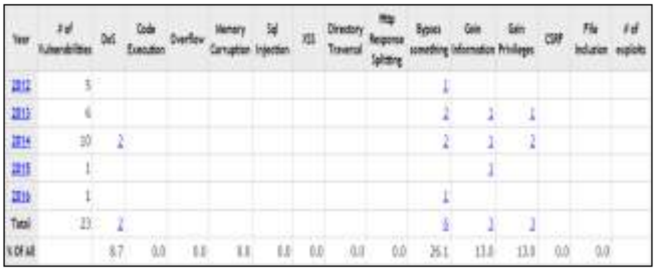
The possible threats associated with keystone are Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges [14].

The following questions should be considered while deploying OpenStack as infrastructure as a service model.

- Did I get the right images and distros?
- Am I running what I think I'm running?
- Could something malicious be injected into the deployment process?
- Am I running the most secure patch level?

VII. SECURITY VULNERABILITIES OF KEYSTONE

Vulnerability statistics provide a quick overview for security vulnerabilities of this software [15].



Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2012	5									1					
2013	6									2	1	1			
2014	10	2								2	1	2			
2015	1										1				
2016	1									1					
Total	23	2								6	3	3			
% Of All		8.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	26.1	13.0	13.0	0.0	0.0	

Figure 6: Vulnerability trends over time

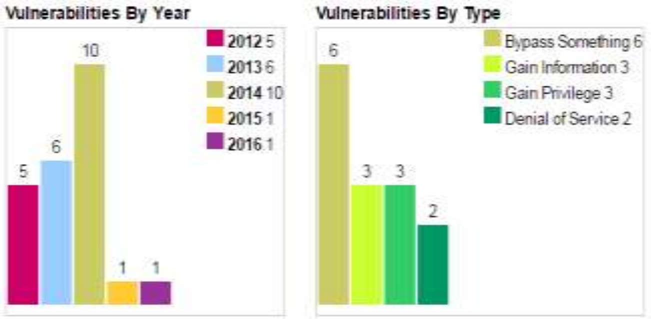


Figure 7: Vulnerabilities by type & year

In older releases of OpenStack, Identity (Keystone), keystone middleware and liberty does not properly invalidate authorization tokens when using the PKI or PKIZ token providers, which allows remote authenticated users to bypass intended access restrictions and gain access to cloud resources by manipulating byte fields within a revoked token. OpenStack Identity (Keystone) logs the backend_argument configuration option content, which allows remote authenticated users to obtain passwords and other sensitive backend information by reading the Keystone logs. OpenStack Identity (Keystone) and Juno does not properly handle chained delegation, which allows remote authenticated users to gain privileges by leveraging a trust or OAuth token with impersonation enabled to create a new token with additional roles. The V3 API in OpenStack Identity (Keystone) and icehouse allows remote attackers to cause a denial of service (CPU consumption) via a large number of the same authentication method in a request, aka "authentication chaining."

VIII. EXPERIMENTAL SETUP

In the experimental setup, only the OpenStack Keystone was used in order to identify the security issues related to this platform. The softwares used in performing the experiment were:

i. Latest version of OpenStack Keystone installed on Ubuntu 16.04 LTS
ii. VirtualBox 5.1
iii. Host Computer with 4 GB RAM and at least 30 GB free disk space having internet connection.

The First step in the experimental setup was to install and configure VirtualBox. This gave a complete multi-node cluster which can be accessed and managed from the computer running VirtualBox. The next step was to create different virtual machines, configuring them. All of this is performed in the VirtualBox which was created initially. VirtualBox can run on Windows, Mac OS X, Linux and Solaris. For this experiment, we had created three virtual machines. These virtual machines are connected to the windows host machine through the virtual box.

We have done the platform test by the experiment of OpenStack based virtual machine against DDoS attack. According to the characteristics of DDoS attacks, it needs multiple attack units to implement attack test, the platform has created multiple virtual machines as an attack unit.

The computation time required by VM under attack is very high as compared to normal VM under operation. Under high overwhelming attack the VM operation may be disrupted and legitimate user will be denied from service.

IX. PLAN OF WORK

We have proposed to build a new OpenStack software component by using Java programming language. This software component consists of a web server that can be accessed only over HTTPS protocol. The server will expose RESTful API which has the very similar structure and signature as the already existing RESTful API offered by OpenStack.

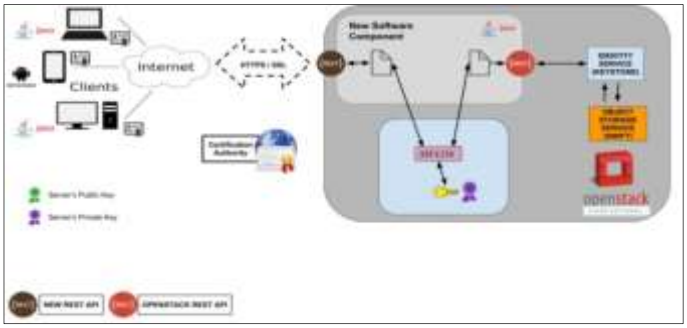


Figure 9: Design draft for proposed authentication model

By using the RESTful API, this new component will interact with existing OpenStack component, in particular with Keystone, in order to use the Identification functionalities via the token mechanism.

X. CONCLUSION AND FUTURE SCOPE

We have systematically examined the threat model against, and security requirements of, OpenStack keystone. Security is a critical concern for OpenStack and it is constantly being evaluated and addressed at all layers – the base operating system and third-party tools through applications and everything in between. The critical elements of a secure OpenStack cloud includes management, secure communications, API endpoints, identity, dashboard, compute, block and object storage, shared file systems, networking, message queuing, data processing, databases, tenant data privacy, instance security management, compliance, monitoring and logging. Keystone, the centralized identity and access management system for OpenStack, provides basic functionalities for authentication and authorization.

We have analyzed security issues in open-source cloud computing project - OpenStack Keystone. We started with finding out which security issues should be taken care of when using cloud services. By examining security-related documents for cloud computing, we are able to compile a list of security issues to be used when evaluating security of OpenStack cloud solution.

We have presented systematic authentication model for OpenStack Keystone. Our future work will focus on to create a secure identity service mechanism integrated in OpenStack environment by using the RESTful API.

REFERENCES

[1]. Ristov S, Gusev M, Kostoska M. Security assessment of OpenStack open source cloud solution, Proceedings of the 7th South East European Doctoral Student Conference (DSC2012). 2012: 577-587.

[2]. http://www.hytrust.com/cloud-sddc-study/

[3]. https://docs.openstack.org/security-guide/identity.html

[4]. Slipetskyy R. Security issues in OpenStack , Master's thesis, Norwegian University of Science and Technology, 2011.

[5]. 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

[6]. 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

[7]. 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.

[8]. Ericsson, Keystone Security GAP and Threat Identification (Quick Study), OpenStack Folsom Release, 2014

[9]. Cirrus, O.: Open cirrus - open cloud computing research testbed (Apr 2012), https://opencirrus.org/

[10]. 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 the 2011, 12th ACM/IFIP/USENIX International Conference on Middleware. pp. 81–100.

[11]. https://wiki.OpenStack.org/keystone

[12]. https://docs.OpenStack.org/developer/keystone/architecture.html

[13]. https://developer.OpenStack.org/api-ref/identity/v3

[14]. https://www.OpenStack.org/assets/presentation-media/OpenStack-Summit-Atlanta-Keystone-Security.pptx

[15]. http://www.cvedetails.com/product/22720/?q=Keystone

[16]. Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: toward an open-source solution for cloud computing, International Journal of Computer Applications, 2012, 55(3): 38-42.

[17]. Khan R H, Ylitalo J, Ahmed A S. OpenID authentication as a service in OpenStack, Information Assurance and Security (IAS), 2011 7th International Conference on. IEEE, 2011: 372-377.

[18]. Ristov S, Gusev M, Donevski A. OpenStack cloud security vulnerabilities from inside and outside, CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization. 2013: 101-107.

[19]. http://superuser.openstack.org/articles/section/user-stories

[20]. Various: Elliptic Curve Cryptography. OpenSSL Wiki (2016). https://wiki.openssl.org/index.php/Elliptic_Curve_Cryptography