ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1147-1158), Month: April - June 2015, Available at: www.researchpublish.com

SECURITY ISSUES IN OPENSTACK

[1]Ishan Gidwani, [2]Dasrath Mane

[1,2] Vivekanand education society Institute of technology, Chembur Mumbai, India

Abstract: **This paper basically derives various vulnerabilities in openstack and we have try provide a solutions to mitigate these exploits. In the first section we briefly look at the security issues that cloud computing faces. We take a look at the security architecture of Cloud computing in general and then understand the security architecture of OpenStack. In the second section we look at all the known vulnerabilities in OpenStack. We understand how these vulnerabilities can be exploited and how can they be mitigated. The majority of the work done on this project is explained in the second section. The third section consists of a security assessment of Openstack. Firstly, We make a security assessment of Openstack with nMap scans, then we attack Openstack with tools like Metasploit, THC Hydra, Acunetix etc. We also try to exploit OpenStack with known vulnerabilities and try to find new vulnerabilities. Finally, we conclude by providing solutions to mitigate these exploits.**

Keywords: **derives various, openstack, security issues, vulnerabilities.**

I. INTRODUCTION

Cloud computing in simple terms is computing services offered remotely by cloud vendors to customers at fairly reduced prices. In recent times cloud computing has become more and more popular and is applied for various purposes. Cloud computing itself is in principle an abstraction of the physical infrastructure which is offered as cloud services to service users. The abstraction levels of these cloud services are Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Popular examples are GoogleDocs [1] for SaaS, Google's AppEngine [2] for PaaS, and Amazon's EC2 [3] for IaaS.

The entire cloud computing infrastructure/services are hosted in data centers. If a service user orders a cloud service, e.g., a virtual machine in Amazon's EC2 or Windows Azure, this virtual resource is placed on a physical infrastructure within a data center of the cloud operator. The virtual resource might be moved within the data center from one physical machine to another, e.g., due to maintenance reasons. Migrating virtual resources to physical machines located in other data centers of the same operator, or to physical machines of other operators automatically are not possible. However, there are some use cases where a flexible placement of virtual resources is needed, e.g., for optimization reasons (reducing costs or latencies when accessing the virtual resource).

Cloud computing offers services that can be publically accessed, therefore, a well-defined security architecture has to be implemented in order to provide a secure working environment for the end users.

1.1 Security issues in the Cloud:

Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing utilizes three delivery models in which different types of services are delivered to the end user. The three delivery models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), which provide infrastructure resources, application platforms and software as services to the consumer. These service models also place different levels of security requirements upon the cloud environment. IaaS is the foundation of all cloud services, PaaS builds upon IaaS and SaaS, in turn, builds on PaaS. As capabilities are inherited by successive models, so too are information security issues and risks. There are important differences between each model in terms of merged features, complexity and security. Cloud service providers can provide the basic security architecture; consumers are responsible for implementing and managing the provided security features. Cloud Security Alliance (CSA)

and Institute of Electrical and Electronics Engineers (IEEE) report that small-and medium-sized enterprises in the public sector are careful when adopting cloud computing, although those securities are needed together to accelerate cloud adoption on a broad scale and to respond to regulative drivers. Organizations using cloud computing IaaS prefer to examine security and confidentiality threats to their business as critical insensitive applications. In addition knowledge and its management is a foundation for creating competitive advantages in organizations. However, ensuring the security of an enterprise's data in the cloud is difficult, but not impossible, if they supply services such as SaaS, PaaS and IaaS. Each of these services has its own security issues.

1.2 Security Issues in SaaS:

In SaaS, the client's security measures are dependent on the provider. The provider should ensure that each user's data are hidden from all other users. Security measures must be in place and the client must be confident that the application will be ready for use when needed. In SaaS, the cloud client will often replace old software applications with newer ones. Therefore, the focus lies not upon the portability of applications but rather upon protecting or developing the security functionality of legacy applications and attaining successful data migration. Vendors of SaaS services may host applications on their own private servers or use cloud computing IaaS provided by a third-party (e.g., Amazon, Google). The use of cloud computing, along with the pay-and-go approach, helps application service providers reduce the cost of infrastructure services and allows them to focus on providing the best possible service to customers.

1.3 Security Issues in PaaS:

In PaaS, developers build applications on a computing platform controlled by the provider. In addition, any security issues beneath the application level, such as network and host intrusion prevention, are under the control of the provider, who must offer strong guarantees that the data cannot be accessed by other applications. As a result, PaaS offers more flexibility than SaaS at the expense of customer-ready features. This trade-off extends to security features and capabilities, in that built-in capability are less complete, but, simultaneously, there is more flexibility to incorporate additional security. Applications which are sufficiently complex to take advantage of an Enterprise Service Bus (ESB), but which need to secure the ESB directly, benefit from protocols such as Web Service (WS) security (Oracle, 2013). In addition, is very beneficial to use PaaS for Successful Executive Information System Development for Education Domain. The capability to segment ESBS is not present in PaaS environments. Standards should be introduced to regulate the effectiveness of application security programs. Between direct application and security, specific metrics available patch coverage and vulnerability scores. These standards can indicate the quality of application coding. Attention should be paid to how malicious entities are adapting to new cloud application architectures that hide application components from their view. Hackers are likely to attack obvious code, although this is not necessarily restricted to code running in the context of the user. They are likely to attack the infrastructure and perform comprehensive black box testing.

1.4 Security issues in IaaS:

In IaaS, the developer has the best control over security, as long there is no security hole in the Virtualization Manager (VM). While in theory virtual machines might be able to address these issues as they arise, there are many security problems in practice. An additional factor is the reliability of the data stored in the provider's hardware. Due to the growing virtualization of the information society, enabling owners to maintain control over their data regardless of its physical location will become a topic of extreme interest. To obtain maximum trust and security on a cloud resource, several techniques need to be practiced. The security obligations of both the provider and the consumer vary greatly between cloud service models. Amazon's Elastic Compute Cloud (EC2) infrastructure presents an example in which the vendor's responsibility for security extends only to the hypervisor. This means that they can only address security controls such as virtualization security, physical security and environmental security. The consumer is responsible for the security controls corresponding to the system, including the applications, OS and data. IaaS gives rise to security issues whose severity depends on the cloud deployment model through which the services are delivered. The physical security of the infrastructure is extremely important; disaster management plans are necessary to prevent damage, either natural or intentional, to the infrastructure.

1.5 Cloud Security Alliance:

Cloud Security Alliance (CSA) is a non-profit organization that promotes the use of best practices for providing security assurance within Cloud Computing. It also has a goal to educate users/providers of cloud services on the uses of Cloud Computing to help secure all other forms of computing.

We take a look at "Identity and Access Management" in Cloud Security Alliance. This section provides controls for assured identities and access management.

Identity and Access Management includes people, processes, and systems that are used to manage access to enterprise resources by assuring the identity of an entity is verified and is granted the correct level of access based on this assured identity. Audit logs of activity such as successful and failed authentication and access attempts should be kept by the application / solution.

Core Functionalities:

- Provisioning/de-provisioning of accounts (of both cloud & on-premise applications and resources)

- Authentication (multiple forms and factors)

- Directory services

- Directory synchronization (multilateral as required)

- Federated SSO

- Web SSO (e granular access enforcement & session management - different from Federated SSO)

- Authorization (both user and application/system)

- Authorization token management and provisioning

- User profile & entitlement management (both user and application/system)

Threats Addressed:

- Identity theft

- Unauthorized access

- Privilege escalation

- Insider threat

- Non-repudiation

- Excess privileges / excessive access

- Delegation of authorizations / entitlements

- Fraud

II. OPENSTACK

OpenStack is simple terms is a set of software tools grouped together for building and managing cloud computing platforms for public and private clouds. Backed by some of the biggest companies in software development and hosting, as well as thousands of individual community members, many think that OpenStack is the future of cloud computing. OpenStack is managed by the OpenStack Foundation, a non-profit organization, which oversees both development and community-building around the project.

2.1 Introduction to OpenStack:

OpenStack lets users deploy virtual machines and instances which handle different tasks for managing a cloud environment on the fly. It makes horizontal scaling easy, which means that tasks which benefit from running concurrently can easily serve more or less users on the fly by just spinning up more instances. For example, a mobile application which needs to communicate with a remote server might be able to divide the work of communicating with each user across many different instances, all communicating with one another but scaling quickly and easily as the application gains more users.

OpenStack is an open source software, which means that anyone who chooses to can access the source code, make any changes or modifications they need, and freely share these changes back out to the community at large. It also means that OpenStack has the benefit of thousands of developers all over the world working in tandem to develop the strongest, most robust, and most secure product that they can build.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1147-1158), Month: April - June 2015, Available at: www.researchpublish.com

2.2 How is OpenStack used in a cloud platform?

The cloud is all about providing computing for end users in a remote environment, where the actual software runs as a service on reliable and scalable servers rather than on each end users computer. Cloud computing can refer to a lot of different things, but typically the industry talks about running different items "as a service" software, platforms, and infrastructure. OpenStack falls into the latter category and is considered Infrastructure as a Service (IaaS). Providing infrastructure means that OpenStack makes it easy for users to quickly add new instance, upon which other cloud components can run. Typically, the infrastructure then runs a "platform" upon which a developer can create software applications which are delivered to the end users.

2.3 What are the components of OpenStack?

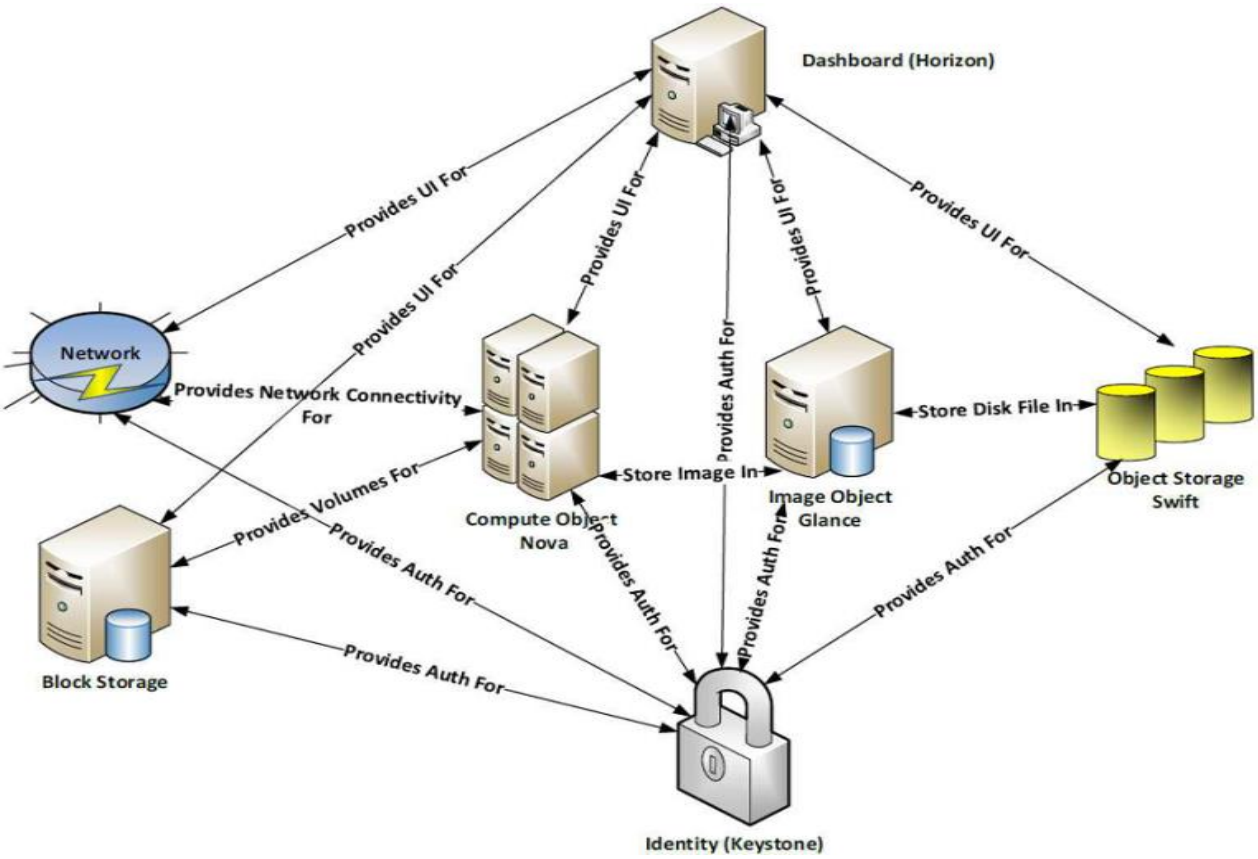

Figure.1 Components of OpenStack

OpenStack is made up of many different moving parts. Because of its open nature, anyone can add additional components to OpenStack to help it to meet their needs. However, the OpenStack community has collaboratively identified nine key components that are a part of the "core" of OpenStack, which are distributed as a part of any OpenStack system and officially maintained by the OpenStack community. Figure 1 above shows the components of OpenStack.

- **Nova:**

Nova is the primary computing engine behind OpenStack. It is a "fabric controller," which is used for deploying and managing large numbers of virtual machines and other instances to handle computing tasks.

- **Swift:**

Swift is a storage system for objects and files. Rather than the traditional idea of a referring to files by their location on a disk drive, developers can instead refer to a unique identifier referring to the file or piece of information and let OpenStack decide where to store this information. This makes scaling easy, as developers don't have the worry about the capacity on a single system behind the software. It also allows the system, rather than the developer, to worry about how best to make sure that data is backed up in case of the failure of a machine or network connection.

- **Cinder:**

Cinder is a block storage component, which is more analogous to the traditional notion of a computer being able to access specific locations on a disk drive. This more traditional way of accessing files might be important in scenarios in which data access speed is the most important consideration.

- **Neutron:**

Neutron provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently.

- **Horizon:**

Horizon is the dashboard behind OpenStack. It is the only graphical interface to OpenStack, so for users wanting to give OpenStack a try, this may be the first component they actually see. Developers can access all of the components of OpenStack individually through an application programming interface (API), but the dashboard provides system administrators a look at what is going on in the cloud, and to manage it as needed.

- **Keystone:**

Keystone provides identity services for OpenStack. It is essentially a central list of all of the users of the OpenStack cloud, mapped against all of the services provided by the cloud which they have permission to use. It provides multiple means of access, meaning developers can easily map their existing user access methods against Keystone.

- **Glance:**

Glance provides image services to OpenStack. In this case, images refers to images (or virtual copies) of hard disks. Glance allows these images to be used as templates when deploying new virtual machine instances.

- **Ceilometer:**

Ceilometer provides telemetry services, which allow the cloud to provide billing services to individual users of the cloud. It also keeps a verifiable count of each user's system usage of each of the various components of an OpenStack cloud.

- **Heat:**

Heat is the orchestration component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application. In this way, it helps to manage the infrastructure needed for a cloud service to run.

III. KNOWN VULNERABILITIES IN OPENSTACK

OpenStack comprises of different sets of components that provide individual functionality. These components can be individually targeted by an attacker. Therefore, to make OpenStack secure as a whole, these individual components have to be assessed for security vulnerabilities. We take a look at some of the vulnerabilities in OpenStack that could have caused serious harm if they weren't mitigated in time.

- **Session-fixation vulnerability:**

The session-fixation vulnerability was discovered in Horizon. This occurred while using the default signed cookie session. The default setting in Horizon is to use signed cookies to store session state on the client side. This creates the possibility that if an attacker is able to capture a user's cookie, they may perform all actions as that user, even if the user has logged out. The services that were affected by this vulnerability were Horizon, Folsom, Grizzly, Havana and Icehouse.

The vulnerability:-

When configured to use client side sessions, the server isn't aware of the user's login state. The OpenStack authorization tokens are stored in the session ID in the cookie. If an attacker can steal the cookie, they can perform all actions as the target user, even after the user has logged out. There are several ways attackers can steal the cookie. One example is by intercepting it over the wire if Horizon is not configured to use SSL. The attacker may also access the cookie from the filesystem if they have access to the machine. There are also other ways to steal cookies that are beyond the scope of this note.

By enabling a server side session tracking solution such as memcache, the session is terminated when the user logs out. This prevents an attacker from using cookies from terminated sessions.

It should be noted that Horizon does request that Keystone invalidate the token upon user logout, but this has not been implemented for the Identity API v3. Token invalidation may also fail if the Keystone service is unavailable. Therefore, to ensure that sessions are not usable after the user logs out, it is recommended to use server side session tracking.

Mitigation:-

It is recommended that one should configure Horizon to use a different session backend rather than signed cookies. One possible alternative is to use memcache sessions. To check if signed cookies are being used, look for this line in Horizon's local_settings.py

```
SESSION_ENGINE = 'django.contrib.sessions.backends.signed_cookies'
```

If the SESSION_ENGINE is set to value other than 'django.contrib.sessions.backends.signed_cookies' this vulnerability is not present. If SESSION_ENGINE is not set in local_settings.py, check for it in settings.py.

If the SESSION_ENGINE is set to value other than 'django.contrib.sessions.backends.signed_cookies' this vulnerability is not present. If SESSION_ENGINE is not set in local_settings.py, check for it in settings.py.

The steps to configure memcache sessions are:

1. Ensure the memcached service is running on the system

2. Ensure that python-memcached is installed

3. Configure memcached cache backend in local_settings.py

```
CACHES = {
    'default': {
       'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
       'LOCATION': '127.0.0.1:11211',
    }
}
```

Make sure to use the actual IP and port of the memcached service.

4. Add a line in local_settings.py to use the cache backend:

```
SESSION_ENGINE ='django.contrib.sessions.backends.cache'
```

- **Heartbleed vulnerability**

The Heartbleed vulnerability was discovered in April 2014. It is a vulnerability in OpenSSL that can lead to OpenStack being compromised. The services that can be affected from this vulnerability are Grizzly, Havana and OpenSSL.

The vulnerability:-

A vulnerability in OpenSSL code-named Heartbleed was recently discovered that allows remote attackers limited access to data in the memory of any service using OpenSSL to provide encryption for network communications. This can include key material used for SSL/TLS, which means that any confidential data that has been sent over SSL/TLS may be compromised. While OpenStack software itself is not directly affected, any deployment of OpenStack is very likely using OpenSSL to provide SSL/TLS functionality.

Mitigation:-

It is recommended that an immediate update should be implemented for the OpenSSL software on the systems that runs OpenStack services. In most cases, the upgrade would be to OpenSSL version 1.0.1g, though it is recommended that one reviews the exact affected version details on the Heartbleed website referenced in this paper.

After upgrading your OpenSSL software, you will need to restart any services that use the OpenSSL libraries. You can get a list of all processes that have the old version of OpenSSL loaded by running the following command:

```
lsof | grep ssl | grep DEL
```

Any processes shown by the above command will need to be restarted, or the entire system can be restarted if desired. In an OpenStack deployment, OpenSSL is commonly used to enable SSL/TLS protection for OpenStack API endpoints, SSL terminators, databases, message brokers, and Libvirt remote access. In addition to the native OpenStack services, some commonly used software that may need to be restarted includes:

- Apache HTTPD
- Libvirt
- MySQL
- Nginx
- PostgreSQL
- Pound
- Qpid
- RabbitMQ
- Stud

It is also recommended that the existing SSL/TLS keys are compromised as compromised and new keys are generated. This includes keys used to enable SSL/TLS protection for OpenStack API endpoints, databases, message brokers, and libvirt remote access.

In addition, any confidential data such as credentials that have been sent over a SSL/TLS connection may have been compromised. It is recommended that cloud administrators change any passwords, tokens, or other credentials that may have been communicated over SSL/TLS.

IV. OVERVIEW OF SECURITY COMPONENTS IN OPENSTACK

- **Identity provision:**

User provisioning is the process of registering a new user to the system, and deprovisiong is the process of user removal from the system. It is very important to automate user management tasks. OpenStack Object Storage uses an “out-of-the-box” solution: tempAuth and swAuth authentication/authorization system. The difference between the two is the back-end repository into which the user data is stored. TempAuth uses configuration file where user data is stored in plain text. Unlike tempAuth, swAuth claims to be more a"scalable authentication and authorization system that uses Swift itself as its backing store". Swift account is created on a Swift cluster and theuser information isstored as JSON-encoded data text files, which are Swift objects. Both swAuth and tempAuth provide possibility for an on-demand identity provisioning and deprovisiong, which is in compliance with industry standards.

User management is roles based in OpenStack Object Storage. The following roles exist:

- Users are not granted to administrate any users themselves.

- Admin can add users to an account which he is allowed to administrate. swAuth enables deletion ofusers from administering accounts.

- Reseller Adminhas Admin permissions on all of the accounts and cannot add other Reseller Admins.

- Super Admin is the most powerful user who can perform all user management procedures, including adding Reseller Admins.

- **User authentication:**

Both authentication systems tempAuth and swAuth use username and password authentication. When an authentication has been successfully performed, theuser receives a token which is usedto identify himto the system afterwards. The provided token has a configurable expiration time and its default value is set to 24 hours. All documents for cloud security

specify itis necessary to allow authentication delegation by accepting confirmations in SAML format; however this feature is not available yet in OpenStack.

Password strength:-

Since all OpenStack projects provide username and password combination to authenticate user it is important to take acloser look and study aboutpassword strength requirements. The "Electronic Authentication Guideline" created by NIST provides some rules to prevent users from choosing bad passwords, including checking a password against adictionary of commonly used passwords, specifying minimal password length, and requiring the use of different characters (lower-case, upper-case, non-alphabetic). Unfortunately, no such requirements (password length and special characters) exist in OpenStack. There are also no dictionary checks, so users are able to register with password as short as one character.

- **Password storage:**

Storing passwords is awell known problem in all information systems thatuse password authentication. The general practice for information security requires the administrator to ensure passwords are not stored in clear-text, but rather encrypted. It is also important to provide alimited access to stored passwords.

As we noted previously, tempAuth stores username and password in a configuration file whereall passwords are stored in plain text format. The location of super user credential is also stored in the same file. By default each user in the system possessesread access to this file. Such approach enables system users to obtain password of other users and gain access to their account easily. This is why tempAuth was never considered by developers to be an option for production deployment. Such weakness should be noted and user should be warned in thesetup documentations. Initial setup should change the access rights of this file to only giveread permissions to the system administrators.

SwAuth uses configuration file where super admin password is stored. Unlike the tempAuth, swauth possessesproperly configured access rights to secure password data. The only security concerns that arise with swAuth are clear-text stored passwords within this file. Because of this issue, aninside attackerwould gainsuperuser account on the system, thus being ableto find out user passwords. OpenStack should consider hashing passwords before storing them in the password file.

To conclude, both tempAuth and swAuth lack anappropriate protection of passwords. A recommendation for both authentication systems taken from NIST's "Electronic Authentication Guideline" would be to store passwords "concatenated to a salt and/or username and then hashed with an approved algorithm, so that the computations used to conduct a dictionary or exhaustion attack on a stolen password file would not beuseful to attack other similar password files".

- **Tokens of authentication:**

Authentication tokens play similar roles as identifiers for web applications. An API, such as an OpenStack service, is used to authenticate a user. Successful authentication generates a token that is used to authorize service requests. The password and username are given as input to the API interface. When authentication succeeds, the resulting feedback includes an authentication token and service catalogue. Note that tokens remain valid for 12 hours. Issued tokens become invalid in two situations:

- If the token is expired
- If the token has been cancelled

It is important that the authentication be executed over a secure channel, such as Transport Layer Security (TLS); otherwise, an attacker could obtain a user token by executing a man-in-the-middle-attack and remove the user who received the token from the authentication system.

- **Malicious Data:**

Most cloud providers do not encrypt data before saving it to a cluster. In fact, OpenStack does not provide any data encryption at all; thus, users would need to encrypt their data before uploading it and manage their encryption keys themselves.

It may be difficult to track security issues in cloud computing environments. Therefore, the primary aim of this study is to highlight the implications of the major security issues.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1147-1158), Month: April - June 2015, Available at: www.researchpublish.com

V. PENETRATION TESTING ON OPENSTACK

A penetration test is a method of finding flaws or bugs in a web application. These bugs can be errors or coding mistakes that might lead to an exploit. They can also be technical faults in a web application that can give away sensitive information, which can be used to understand the logical flow of different components. A penetration test gives a security overview of a web application and helps us in mitigating vulnerabilities.

What can be attacked?

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 2 below shows the possible exploits that can be performed on OpenStack.

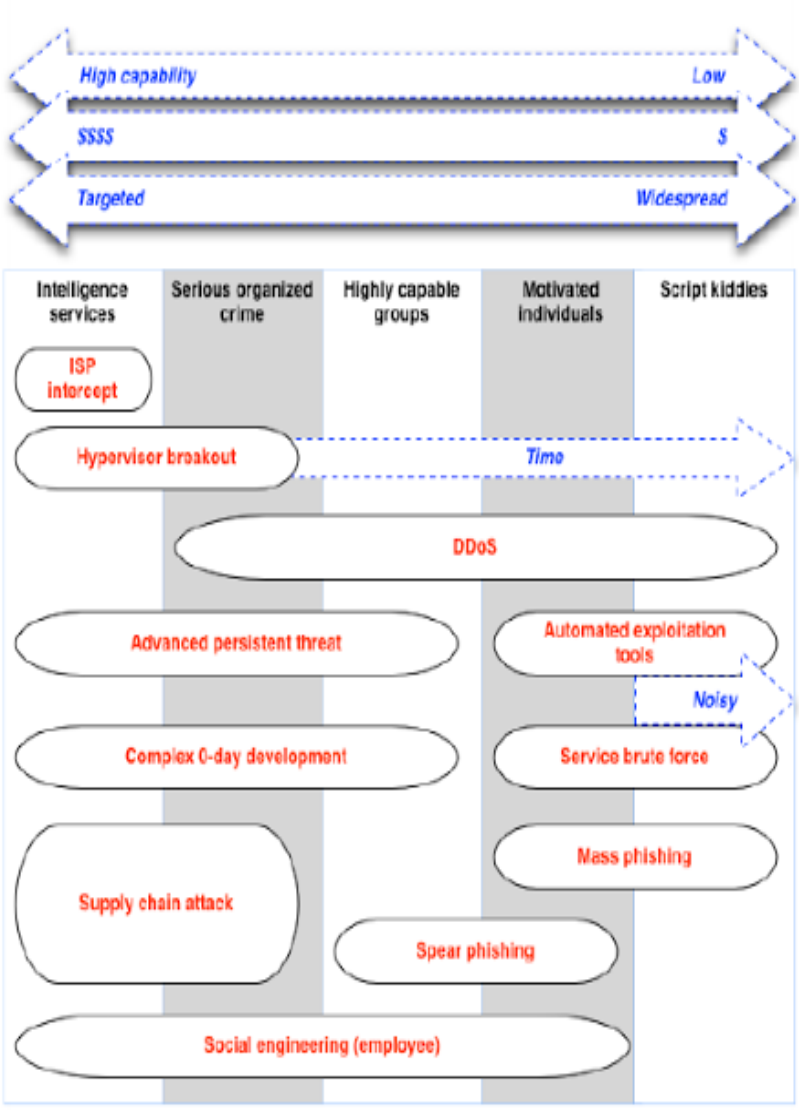

Figure.2 Exploits on OpenStack

- **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 crendentials 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.

A hypervisor breakout can be executed by the following methods -

(1) Highly capable groups:-

These are extremely talented groups of hackers who hack for criminal purposes or sometimes funded by the some organizations to harm the other organization.

(2) Motivated individuals:-

These people work only for themselves; they can retrieve other personal information for their own evil purposes or just for fun.

(3) Script Kiddies:-

Script kiddies are those people who use automated tools to exploit the organization systems and they do not know what they are doing and how is the attack actually happening. They might be carrying out the attacks on the attackers' behalf.

- **DDOS Attack:**

A Disributed 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 hacktivist 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 developements 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.

VI. EXPLOITS

To carry out the exploits we first had to set up a test environment. We used 2 machines to carry out our attacks. The first machine was the victim running OpenStack on Ubuntu 14.04 LTS with Apache server. The second machine was the attacker that was running Kali Operating system. Kali comes with a wide range of attacking tools that can be used to execute exploits. After setting up the test environement, our first step was to gather as much information as possible from the victim.

- **nMap scans:**

In penetration testing, the first step is information gathering; to gather the information about the victim machine we used nMap. It is one of the best tools to gather the information about the target like open ports, running services, operating system fingerprinting etc.

First, we use the nMap hosts scan to get the basic idea about the host. To do a host scan we can use the following command–

root@kali:~# nmap –sp 192.168.1.4

The information we get from the nMap hosts scan revels that the host is up and running services.

Then we performed an nMap port scan to get details of all the open ports on the victim machine. To perform a nMap port scan use the following command-

root@kali:~# nmap 192.168.1.4

The nMap port scan reveals that there are several ports open on the victim machine. But, the one that we are interested in is port 5000. This is the port that OpenStack is running on.

Our next step was to perform an nMap aggressive scan. Although, the nmap aggressive scan generates a huge amount of traffic, this helps us understanding areas that can be targeted for our exploits. To perform an nMap aggressive scan we can use the following command –

root@kali:~# nmap –sV 192.168.1.4

In our case the nMap aggressive scan does not reveal many details about the possible vulnerabilities that can be exploited, but it helps us getting a big picture overview of our victim machine.

- **Attacking with THC Hydra:**

After collecting enough information about our victim we implement our first attack. When we installed OpenStack we noticed that a Captcha was not used to check the authentication of OpenStack Dashboard access. This means that we can execute a dictionary attack and try different user name and password combinations.

A Captcha code can be used to limit the number of attempts for a user to login. Since, this is not used, we use THC Hydra to implement any number of Brute Force attempts to gain access to the system.

The following command in Kali can be used to implement a Dictionary attack. A Dictionary of username and passwords can be downloaded from http://www.FreeRainbowTables.com/

root@kali:~# hydra –l user –P passlist.txt ftp://192.168.1.4

VII. CONCLUSION

Cloud computing provides an important benefit to companies looking for an advantage in today's economy. Many providers are offering cloud computing services; this competition will lead to increasingly affordable rices over time. Lower prices enable businesses to use staff for other tasks and allow them to consume resources more efficiently by paying for services only as they are needed. These features, supported by an attractive and economical pay-as-you-go approach, have led to growing support for this model. One important threat posed by cloud computing is the obscuring of boundaries between internal and external security concerns. To understand how well companies' data are kept safe, security services in the cloud must be closely studied. In second level will be the vailability, as providers can be victims of attacks that stop the running of their operations. We looked at the general security structure of OpenStack by understanding individual components in OpenStack. There were a few weaknesses in the storage of user credentials and authorization in OpenStack.

This study discusses issues that arise with the deployment model of cloud computing; in particular, this study focuses on OpenStack security issues and threats. Certain parts of OpenStack are considered secure while others need to be improved. OpenStack does not support minimum password complexity requirements and passwords are stored in plain text format. There are no controls to regulate access to sensitive files, including those containing passwords. Information transferred within the cloud is not protected through the use of file encryption techniques. A

We performed penetration test against OpenStack with all well-known and industry leading security auditing tools but we were not able to exploit the OepnStack some low severity vulnerabilities found but those are not too much critical. Then we also tried to exploit the apache server with version 2.4.7 which resulted in a failed attempt because. We conclude that

OpenStack is a secure cloud platform, however keeping OpenStack secure for everyone is a challenege in itself, as new vulnerabilities are discovered with time. But, the current result from our project shows that OpenStack is a secure cloud software.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Mr. Dashrath Mane for his continuous support in my project and his willingness to bring his breadth of experience to this project. I would like to thank my lecturers and colleagues in VESIT, MCA program for the great learning experiences and interaction we shared which helped me in my project to a great extent.

REFERENCES

[1] Hala Albaroodi, Selvakumar Manickam and Parminder Singh, "Critical review of openstack security: issues and weeknesses" Journal of Computer Science 10 (1): 23-33, 2014, National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia, 11800, Penang, Malaysia

[2] "Google Docs," June 2014. [Online]. Available: http://docs.google.com

[3] "Google App Engine," June 2014. [Online]. Available: http://code. google.com/appengine/

[4] "Amazon Virtual Private Cloud," June 2014. [Online]. Available: http://aws.amazon.com/ec2/

[5] "Session-fixation vulnerability in Horizon when using the default signed cookie sessions," June 20 2014.[Online].Available: https://wiki.openstack.org/wiki/OSSN/OSSN-0017/

[6] "OpenSSL Heartbleed vulnerability can lead to OpenStack compromise," April 10 2014. [Online]. Available: https://wiki.openstack.org/wiki/OSSN/OSSN-0012/

[7] "Glance allows non-admin users to create public images," May 31 2014. [Online]. Available: https://wiki.openstack.org/wiki/OSSN/OSSN-0015/

[8] "Nova Network configuration allows guest VMs to connect to host services," June 20 2014. [Online]. Available: https://wiki.openstack.org/wiki/OSSN/OSSN-0018/

[9] "Cloud Security Alliance, White papers and educational material," June 2014. [Online]. Available: https://cloudsecurityalliance.org/education/white-papers-and-educational-material/

[10] "The Heartbleed bug," April 10 2014. [Online]. Available: http://heartbleed.com/