Intrusion Detection in OpenStack Using Snort

Subhadarshi Samal
Computer Science (CIDSE)
Arizona State University
Tempe, Arizona, USA
Email – ssamal1@asu.edu

Abstract: **Cloud Computing is no doubt a successful computational business model today. Because of the accessibility, reliability and cost-saving service, it is an attractive choice among the customers. However, despite being attractive, it also poses various security threats. In a traditional network environment, IDS is normally deployed at the edge of the defended network infrastructure to protect it from external attackers. But in Cloud Computing where the computing and communication resources are pooled together from various resources and shared among several users, such strategy cannot be effective. So we have to implement a proper defense strategy which can work effectively in a distributed environment.**

Keywords: Cloud Computing, Network Intrusion Detection, Snort, OpenStack

1. INTRODUCTION

Cloud Computing is a major and rapidly emerging part of the IT Industry today. In 2012 global spending on cloud services was $110.3B and is expected to grow to $210B by end of 2016. According to NIST "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics (On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, Measured Service); three service models (Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS), Cloud Infrastructure as a Service (IaaS)); and, four deployment models (Private cloud, Community cloud, Public cloud, Hybrid cloud). Key enabling technologies include: (1) fast wide-area networks, (2) powerful, inexpensive server computers, and (3) high-performance virtualization for commodity hardware" [1]. In other words, it provides a framework for supporting end users easily by attaching powerful services and applications over the internet. It is a method to increase the capacity or add capabilities dynamically without in new infrastructure, training new personnel or licensing new software. These are some basic features which makes Cloud Computing as a rapidly growing computational model in today's IT world without any doubt. What this computational model can deliver is- On demand network access to a shared pool of configurable computing resources such as Network, Storage, Servers, Applications etc. "as a service" for satisfying computational requirements of others. The important five characteristics which makes Cloud Computing so unique are 1. On-demand self-service 2. Ubiquitous network access 3. Resource pooling 4. Rapid elasticity 5. Measured service and on basis of the mentioned characteristics Cloud Computing is further classified into three main service models such as:

1. Software as a service (SaaS). 2. Platform as a service (PaaS) and 3. Infrastructure as a service (IaaS) [2]. The ability to dynamically provision resources makes cloud computing highly valuable from a cost prospective and this dynamic provisioning of resources is usually in the form of number of virtual machines. However, the cheap availability of significant amount of computational resources also poses the threats of distributed attacks. Evidences in recent research works have shown how it is possible to exploit some properties and features of cloud computing infrastructure [3]. For example: SIP brute force attack from Amazon EC2 cloud [4]. This type of vulnerabilities will create concerns towards security of cloud computing and hence will hinder its rapid growth. This survey report aims towards analyzing security of cloud infrastructure by deploying network based intrusion detection into Cloud Computing. NIDS captures the network packets and applies intrusion detection techniques on captured packets in order to detect network attacks. Snort[5] - an open source network intrusion detection and intrusion prevention system is considered for this survey. The report is structured as follows: We are going to start with Cloud computing models, mainly OpenStack and

Eucalyptus. Then we will discuss about Intrusion Detection mechanism in Cloud Environment. In this section we will go over Snort in detail. Then we are going to check the possible ways snort can be added to the Cloud network.

II. CLOUD COMPUTING MODELS

Cloud Computing is a set of resources and services offered through the Internet. Throughout the world datacenters are set up by different companies to provide cloud services. It is considered as the next generation computing platform that can provide dynamic resource pools, Virtualization and high availability.

A cloud can be either private, public or hybrid. A private cloud is used by a single organization. It can be hosted by the organization itself or by a cloud service provider company. A public cloud is provisioned for open use for the public by a particular organization, who also hosts the service. Community clouds are shared by several organizations and are typically externally hosted, but can be internally hosted by one of the organizations. A hybrid cloud is a composition of two or more clouds (private, community or public) that remain unique entities but are bound together offering the benefits of multiple deployment models [6].

Cloud Service is offered by various companies. For example: Google has its own cloud service named Compute Engine. Similarly, Amazon's cloud service is named as Elastic Compute Cloud (EC2) [7]. Again Eucalyptus, a free an open source software used for building of Amazon Web Services [8]. Another free and open source cloud computing solution is OpenStack [9].

II. A. OpenStack

It is an open source IaaS that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface [12].

OpenStack began in 2010 as a joint project of Rackspace Hosting and NASA. NASA, when facing the issue of how to host and process high-resolution images of the Moon and Mars they decided to build their own cloud computing fabric controller. OpenStack is a free and open source cloud-computing platform for private and public clouds released under Apache License 2.0. Today over 200 companies has joined the project, including big names such as Cisco, Dell, and IBM. It has seven major components. It has 7 major components.

- *Compute (Nova)* - is a Cloud computing fabric controller, the main part of an IaaS. It manages and automate pools of computer resources and can work with several virtualization technologies. Its architecture allows it to scale horizontally on standard hardware.

- *Object storage (Swift)* - is a scalable redundant storage system. Software logic ensures integrity of data by replicating and distributing it across different devices. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster.

- *Block Storage (Cinder)* – it provides persistent block-level storage devices for use with OpenStack compute instances. The block storage system manages the creation, attaching and detaching of the block devices to servers.

- *Networking (Neutron)* - is a system for managing network and IP addresses. OpenStack Networking ensures the network is not a bottleneck or limiting factor in a cloud deployment and gives users self-service ability, even over network configurations. Users can create their own networks, control traffic, and connect servers and devices to one or more networks. Administrators can use software-defined networking (SDN) technologies like OpenFlow to support high levels of multi-tenancy and massive scale.

- *Dashboard (Horizon)* - is a graphical interface to access, provision and automate cloud-based resources. The design accommodates third party products and services, such as billing, monitoring, and additional management tools.

- *Identity Service (Keystone)* - is a central directory of users and acts as a common authentication system. It provides a central directory of users mapped to the OpenStack services they can access.

- *Image Service (Glance)* - is used for discovery, registration and delivery of services for disk and server images.

Other than the above components, there are other components being developed such as Telemetry

(Ceilometer), Database (Trove), Messaging (Zaqar) and Orchestration (Heat).

Networking in OpenStack [10]: The OpenStack *Networking* service provides an API that allows users to set up and define network connectivity and addressing in the cloud. The project code-name for Networking services is neutron. OpenStack Networking handles the creation and management of a virtual networking infrastructure, including networks, switches, subnets, and routers for devices managed by the OpenStack Compute service (nova). It is completely standalone and can be deployed to a dedicated host.
To configure rich network topologies, you can create and configure networks and subnets and other OpenStack services like Compute will request to be connected to these networks by requesting virtual ports. In particular, Networking supports each tenant having multiple private networks and enables tenants to choose their own IP addressing scheme, even if those IP addresses overlap with those that other tenants use. Refer - Figure -1.

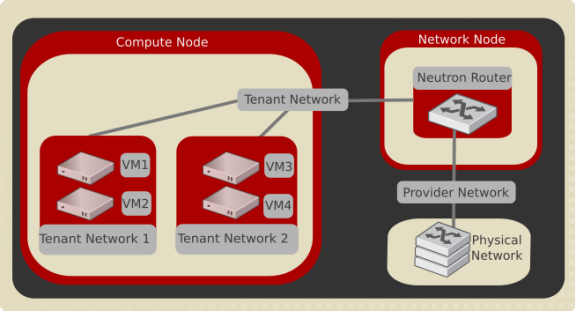

Figure1

When traffic is coming from the internet to the cloud, it first arrives at the network node. Network node has three major components for the management of network such as DHCP, L2 and L3 agent. DHCP dynamically allocates IP Addresses to the VMs on the data networks. L2 agent is an Open vSwitch (OVS) agent that enables distributed virtual router functionality at the L2 layer. The L3 agent, known as the "neutron-l3- agent", creates virtual routers that connect L2 networks. It uses the Linux IP stack and iptables to perform L3 forwarding and NAT. Connecting the Network node to the Compute nodes is a physical switch. Each Compute node can consist of multiple tenants, i:e basically consumers or customers of the cloud. They are isolated resource containers forming the organizational structure within the Compute servers. Each tenant consists of a separate VLAN, volumes, instance, images keys and users. Refer- Figure2.

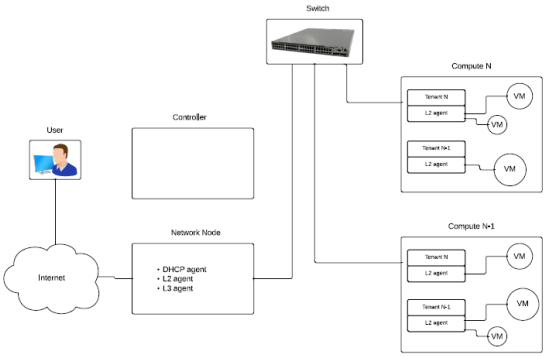

Figure2 – Logical Network Layout

A standard OpenStack Networking setup has up to four distinct physical data center networks [18] (Refer Figure 3):

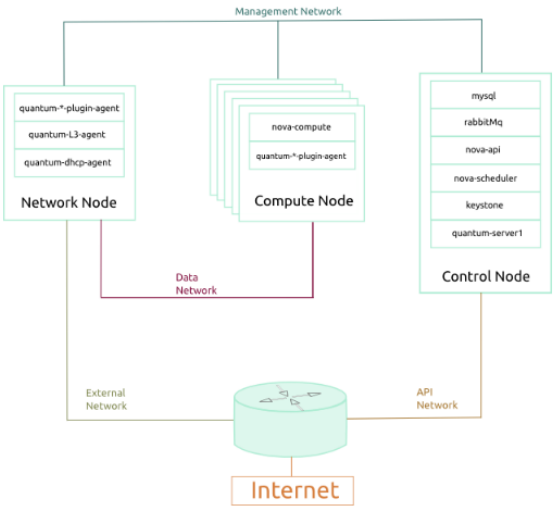

Figure -3 Network Diagram

- Management network: Used for internal communication between OpenStack Components. The IP addresses on this network should be reachable only within the data center.
- Data network: Used for VM data communication within the cloud deployment. The IP addressing requirements of this network depend on the OpenStack Networking plug-in in use.
- External network: Used to provide VMs with Internet access, in some deployment scenarios. The IP addresses on this network should be reachable by anyone on the Internet.
- API network: Exposes all OpenStack APIs, including the OpenStack Networking API, to tenants. The IP addresses on this network should be reachable by anyone on the Internet. This may be the same network as the

external network, as it is possible to create a subnet for the external network that uses IP allocation ranges to use only less than the full range of IP addresses in an IP block.

II.B Eucalyptus Cloud Computing Model [11]

This is another open source framework for cloud computing that implements Infrastructure as a Service. It has been designed to be interface-compatible with Amazon's Elastic Computing Cloud. Its design is based on three components and each component has well defined web-service interface. The software architecture has been organized according to a three-level hierarchy. The bottom layer consists of Node Controllers (NC), responsible for managing virtual machines running on top of physical machine. of a physical machine. The middle layer contains Cluster Controllers (CC). Each CC manages a set of NCs residing on the same physical subnet. The topmost layer is the Cloud
Manager (CM), that manages all the CCs and takes care of high-level resource scheduling. The Cloud Manager is the entry-point to the whole Eucalyptus system for end users as well as administrators. Eucalyptus supports both KVM and Xen virtualization techniques. The important network configuration is done using managed mode.
Eucalyptus provides all the functionality present in Amazon EC2, including instances, subnetworks isolation. Network isolation is obtained through the use of VLANs, which impose appropriate configurations in data center's switches. Following the Eucalyptus terminology, each instance's network is referred to as a security group. Each user is bound to at least one security group, but association to multiple groups can be defined as well if
needed. When configuring Eucalyptus for managed mode, the administrator must define an IP subnet entirely dedicated to the cloud. Moreover, the administrator must define the number of IP addresses available for each security group, actually defining how subnetting is performed. To guarantee access to external networks, each security group includes the cluster controller among its hosts. Instances are configured to use the CC as default gateway (Fig-4).

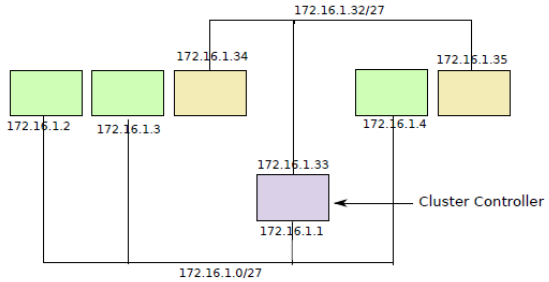

Fig.4 – Eucalyptus managed mode networking

The CC provides both DHCP and NAT services. NAT is realized using standard GNU/Linux's netfilter functionality. Features like elastic IPs are provided by means of rules configured on the CC's configured as a NAT. Eucalyptus exploits software bridges and Xen's virtual Network Interface Cards (NIC) to build virtual networks: when a security group is firstly created, Eucalyptus tags the physical NIC with the security group's VLAN tag and creates a software bridge for each physical machine; the bridge is actually created in the management virtual machine which starts at the boot of a physical machine. Such machine is usually named Dom0 in Xen context. The tagging process creates an abstract NIC to which tagged traffic is forwarded; such interface is then attached to the software bridge. Since Xen creates a new pair of "connected virtual ethernet interfaces", with one end of each pair in the virtual machine and the other end within Dom0, each newly created instance's virtual NIC that resides in Dom0 is attached to the corresponding security group's bridge (Fig- 5). Access to security groups is controlled by the CC's firewall. By default, a security group is not accessible from external networks, and allowed traffic must be specified in terms of source network/address and port number through the Eucalyptus' API.

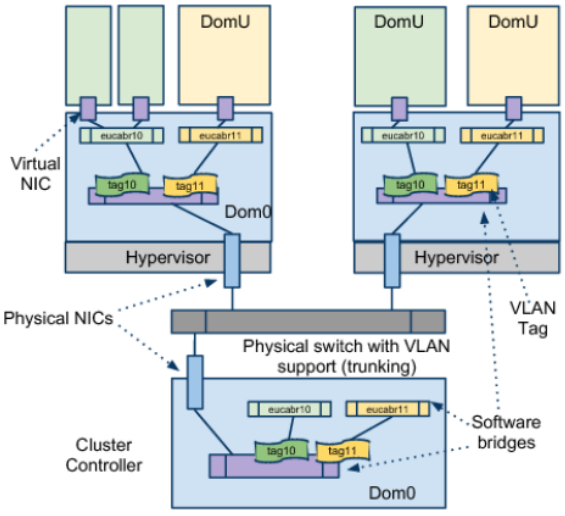

Fig-5 – Eucalyptus Managed mode – Infrastructure view.

III. Intrusion Detection System in Cloud

With the popularity of various Cloud Computing models, the biggest concern arose is Security. So there have been many research on deploying Intrusion Detection System into cloud. There have been many

proposed architectures for this. We will explore the scenario for OpenStack.

In general, Intrusion Detection System (IDS) is the tool for monitoring traffic in the network. Whenever, it identifies vulnerabilities in the network, it sends a report to the system or network administrator.

An IDS is like an alarm on the computer. It alerts the administrator to block the unauthorized persons from accessing data. IDS control the intrusion detection process while IPS has all the features from IDS as well as tools to prevent the attacks from intruders. Currently IDS are coming along with intrusion prevention mechanism (for eg: Snort). Again the intrusion detection mechanism is classified into following categories.

Signature Based Detection: Signature based detection works by comparing packets to a set of attack signatures, which are stored in some form of database and must be frequently updated as new attacks emerge. This is also known as pattern based or knowledge based detection. Its working is similar to many of the virus scanners. If the data matches one of the signatures in the database, then the system decides there is a possible virus. If an antivirus fails to detect, then this means no signatures are available in the database for the current virus attack or the database is out-of-date. It is very effective for known attacks, but it may not work for the attacks whose pattern is not in the database or for zero-day attacks. Signature based IDS can be placed in the network as well as on a host. The advantages to a pattern-matching IDS include an easier implementation and smaller learning curve for the administrator, as well as the generating fewer false positives. However, they are more easily fooled by the obfuscation and evasion techniques of black hats [13].

Anomaly Based Detection: This is also known as Heuristic based or behavior based detection where it discovers intrusions by looking for activity that is different from a user's or system's normal behavior. Events are classified into either normal or anomalous based on the rules or heuristic instead of signatures. Heuristic is nothing but a practical technique for learning, discovering and problem solving etc. Therefore, by using this method, the process moves quickly for finding a better solution and also detects incorrect system activities. Normal network activities for various users are defined based on the user group they belong. This type of detection mechanism will no doubt capture any new type of attack but at the same time, since anomaly detection techniques signal all anomalies as intrusions, false alarms are expected when anomalies are caused by behavioral irregularity instead of intrusions. To minimize this problem, pattern recognition techniques and anomaly detection techniques are often used together to complement each other [13].

Network Based Detection: This type of intrusion detection system is implemented on particular network segments or devices. It monitors the network by analyzing both network as well as application protocol activity to identify any kind of suspicious activity. It also monitors all the incoming and outgoing packets on a network. This type of detection mechanism is good for detecting unauthorized outsider access and bandwidth theft or denial of service attacks. It is fairly easy to implement and doesn't affect the speed of the network. But it should be implemented with proper planning and considering the traffic growth. Else, once the IDS gets overloaded, it will start dropping packets [13].

Host Based Detection: This type of detection mechanism is implemented on for specific nodes or host computers to identify network intrusions. It uses the system logs, system services and registry events for effective intrusion detection. But it is not effective as compared to other models of intrusion detection [13].

IV. NIDS USING SNORT

Snort is currently the most popular free network intrusion detection software. It has multiple advantages such as: it can perform protocol analysis, content searching/matching and can also be used to detect variety of attacks like buffer overflow, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts etc. It is easy to configure, rules are very flexible, easily written and easily inserted into the rule base. In case of a new attack is found, a rule to prevent the attack can be added to snort rule base immediately. It also allows for examination of a packet down to the payload to determine what caused the alert, why the something caused the alert, and whether action needs to be taken. Snort monitors or "sniffs" network packets and logs and can alarm the administrator of a packet matching a specific rule or containing specific information [14].

How Snort works: Packet comes from internet and enters into packet decoder and it goes through several phases, required action is taken by snort at every phase like if detection engine found any miscellaneous content in packet then it drops that packet and in the

way towards output module packet is logged in or alert is generated. Snort is composed of five components [15] (Fig-6).

- Packet Decoder - The packet decoder captures the packet from different types of network interfaces and setup packets for preprocessing.

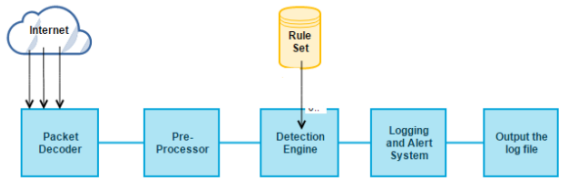

Fig-6 Snort Work Flow.

- Packet Preprocessor – it is used to arrange and modify packets before analyzed by the detection engine. And they try to detect some basic anomalies in the packet header. Preprocessors are very important for any IDS to prepare data packets to be analyzed against rules in the detection engine.
- Detection Engine – it is the heart of the snort. Its responsibility is to analyze all the packets passing through it for the sign of intrusion by making use of certain predefined rule. If the packet matches the rule appropriate action will be taken; otherwise the packet is dropped. It may also take different amount of time to respond for different packets irrespective of how many rules we define.
- Logging and Alert System - Depending upon what detection engine finds out, the activity is logged or alert is generated. Logs are kept in simple text files or tcp-dump style files. The location of logs and alert can be modified using –l command in the command prompt.
- Output Modules - It is used to control the type of output produced by the logging and alert system. Some of its function includes may be generating log reports, sending SNMP traps, logging into databases (like MySQL), sending a message to syslog server etc.

Snort Rules: Snort uses a simple, lightweight rules description language that is flexible and quite powerful. Snort rules must be completely contained on a single line. The Snort rule parser doesn't know how to handle rules on multiple lines. Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken [17].

Snort when used along with some other applications/tools such as PulledPork, Barnyard2 and Snorby, becomes more effective as it is then more easy to manage and monitor [16].

PulledPork - This is a rule management application, i:e it manages the rule we use in Snort.

Barnyard2 - This processes the alerts generated by snort and processes them in to a database format. One of the issues that came with snort is that how snort can keep processing the network traffic without dropping packets and performing extensive output operations such as send alerts and log them to the syslog or a database. Barnyard2 satisfies the purpose.

Snorby - Snorby is front end web application for any application that logs events in the unified2 binary output format. It doesn't perform any network monitoring tasks by itself, instead it depends on other IDS's such as Snort to report data to it.

Snort is a really powerful tool as it can act as a sniffer, packet logger and NIDS at the same time [19].

Snort as Sniffer: Snort acts like tcpdump in sniffer mode. The Snort sniffer mode output is slightly different than the other command-line sniffers. It is actually very easy to read and you may find you prefer it for quick captures. One very nice feature of sniffer mode Snort is the network traffic summary at the end of the capture. # snort –v –d –e -v Dump the packet header to standard output -d Dump the packet payload (includes all TCP, UDP, ICMP packets) -e Display the link layer data

Snort as a Packet Logger: Once after the sniffing is done packet has to be logged. Logging is as simple as adding the -l option, followed by the directory in which you wish to store the logs. Snort's default logging directory is \snort\log. We can use the -d, -a, and -e options to control the amount of information logged for each packet # snort –l {log-directory} will log the packet to specified folder.

Snort as NIDS: When used as an NIDS, Snort provides near real-time intrusion detection capability. Snort does not log the captured file but it applies the rule, if any matches found then it will log or alert the system. snort –c C:\Snort\etc\snort.conf will starts the snorts in NIDS mode. This command displays the alert in the log directory. #snort –vde –l c:\snort\log –c c:\snort\etc\snort.conf

V. SNORT RULE EXPLAINED

Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken.

Here is an example rule: alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg: "mountd access";)

The text up to the first parenthesis is the rule header and the section enclosed in parenthesis is the rule options. The words before the colons in the rule options section are called option keywords. Note that the rule options section is not specifically required by any rule, they are just used for the sake of making tighter definitions of packets to collect or alert on (or drop, for that matter). All of the elements in that make up a rule must be true for the indicated rule action to be taken. When taken together, the elements can be considered to form a logical AND statement. At the same time, the various rules in a Snort rules library file can be considered to form a large logical OR statement.

Rule to prevent ICMP flood attack: alert icmp any any -> any any (msg:"Alert! ICMP FLOOD DETECTION!!!"; itype: 8; threshold: type threshold, track by_dst, count 10, seconds 30; rev:001; sid:10000001; priority:1;) Explanation:

- A rule Action: In this rule the action is "alert" which means that an alert will be generated when conditions are met. Remember that packets are logged by default when an alert is generated. Depending on the action field, the rule options part may contain additional criteria for the rules.
- Protocol. In this rule the protocol is ICMP, which means that the rule will be applied only on ICMP-type packets. In the Snort detection engine, if the protocol of a packet is not ICMP, the rest of the rule is not considered in order to save CPU time. The protocol part plays an important role when you want to apply Snort rules only to packets of a particular type.
- Source address and source port. In this example both of them are set to "any", which means that the rule will be applied on all packets coming from any source. Of course port numbers have no relevance to ICMP packets. Port numbers are relevant only when protocol is either TCP or UDP.
- Direction. In this case the direction is set from left to right using the -> symbol. This shows that the address and port number on the left hand side of the symbol are source and those on the right hand side are destination. It also means that the rule will be applied on packets traveling from source to destination. You can also use a <- symbol to reverse the meaning of source and destination address of the packet. Note that a symbol <> can also be used to apply the rule on packets going in either direction.
- Destination address and port address. In this example both are set to "any", meaning the rule will be applied to all packets irrespective of their destination address. The direction in this rule does not play any role because the rule is applied to all ICMP packets moving in either direction, due to the use of the keyword "any" in both source and destination address parts.
- The msg rule option tells the logging and alerting engine, the message to print along with a packet dump or to an alert.
- The sid keyword is used to uniquely Identify Snort rules. This information allows output plugins to identify rules easily. This option should be used with the rev keyword. 2#2100 Reserved for future use. 100-999,999 Rules included with the Snort distribution 36#361,000,000 Used for local rules. The rev keyword is used to uniquely identify revisions of Snort rules. Revisions, along with Snort rule id's, allow signatures and descriptions to be refined and replaced with updated information. This option should be used with the sid keyword.
- The priority tag assigns a severity level to rules. A classtype rule assigns a default priority (defined by the config classification option) that may be overridden with a priority rule.
- The itype keyword is used to check for a specific ICMP type value.
- The threshold keyword defines a rate which must be exceeded by a source or destination host before a rule can generate an event.

VI. SNORT PLACEMENT IN CLOUD

Snort can be placed at different positions in the OpenStack cloud environment individually or together depending upon the largeness of the cloud network. The main goal is it should cover whole range of the traffic: external, internal and local. Various types of traffics can be well understood from the below diagrams [18].

External Traffic: External traffic is traffic between virtual machines running on Compute nodes and the Internet (Fig-7). This traffic will pass through the Network node and the physical switch before arriving at a Compute node.

Internal Traffic: Internal traffic is traffic between virtual machines running on different Compute nodes (Fig-8). This traffic never reaches the Network node, but is being sent from a Compute node to the physical switch, and then directly to another Compute node.

Local Traffic: Local traffic is traffic between virtual machines on the same Compute node. This traffic never leaves the Compute node. It is being sent from a virtual machine to another through the interface br-int (Fig-9).

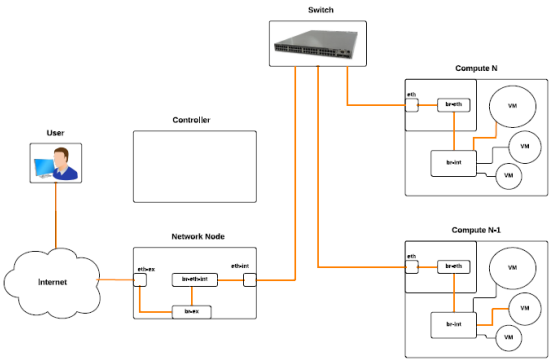

Fig-7 – External traffic

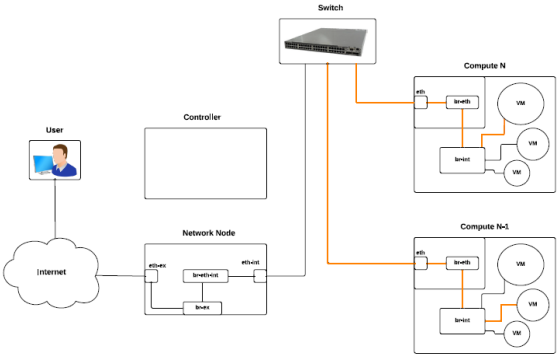

Fig-8 – Internal Traffic

The first position to place an IDS can be network node. Network node is the gateway between internet and the cloud infrastructure. But this place covers the traffic to/from external traffic but doesn't cover internal or local traffic.

Another good location to place IDS can be the Physical switch as it covers both internal and external traffic.

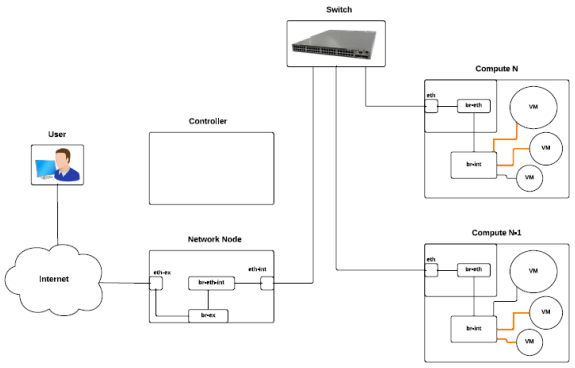

Fig-9 Local Traffic

It is not possible to install snort on the switch itself but entire traffic can be mirrored with a span port to separate machine with snort installed. This type of implementation gets the advantage as processing power for snort comes from a separate machine, not from any element forming the cloud infrastructure.

Again, we have to consider the local traffic as well. Local traffic mainly flows among the virtual machines of a compute node. So snort needs to be installed in the compute node. As there can be many compute nodes within an OpenStack cloud, Snort will act as a load balancer among the compute nodes. Since all virtual machines of a compute node are connected to br-int, it is good to place snort on this interface.

VII. ANOMALY DETECTION IN OPENSTACK

Snort is no doubt an effective IDS for signature-based intrusion detection system. But it is not effective for anomaly detection. Therefore, additional anomaly detection technique should be additionally used along with pattern-based intrusion detection to make the Cloud more secure.

Here is a proposed anomaly detection mechanism [20] (Fig-10). The model consists of three components. Back Propagation Neural Network Algorithm (BPN), an Optimization Algorithm and an Anomaly Detection Database. In the learning phase, the BPN classifier is

trained using malicious and normal packets stored in anomaly detection database. In detection phase, the BPN predicts the class of the given network packets.
If it is normal, it is allowed to access to Cloud Infrastructure, else it is denied and 'alert system' is notified.
Alert System: It generates alerts about intrusions that are determined either by snort or BP neural network classifier. It stores alerted intrusion in central log database.
Central log of malicious packets: It is used by other hosts to update their database with the alerts found in alert database. So that the next time such intrusion can be easily detected by snort on other hosts. This results the reduction of computational cost and detection time in the overall cloud.

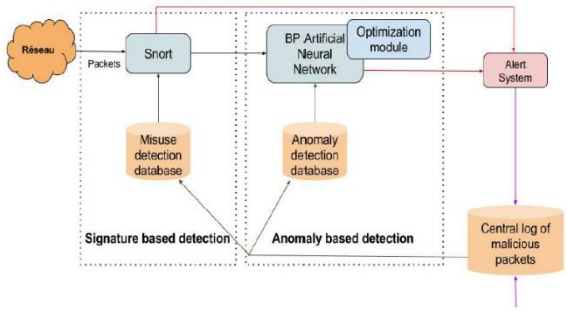

Fig-10 Anomaly detection model

Workflow: Network packets are captured from physical and virtual network. Then signature based technique is applied on captured packets to detect intrusions using Snort. It consists of matching the captured packets with rules stored in attack signature database. Then, alerted packet is denied.
Non-intrusion packets are forwarded to optimized BPN classifier, which is applied to predict class label (normal or intrusion) of the packets. If any intrusion is found, then it will be alerted and stored in the central log base. Else, BPN considers them as legitimate packets and allows them to access the system. NIDS on the
other servers update their base with alerts logging in central log base (Fig -11).

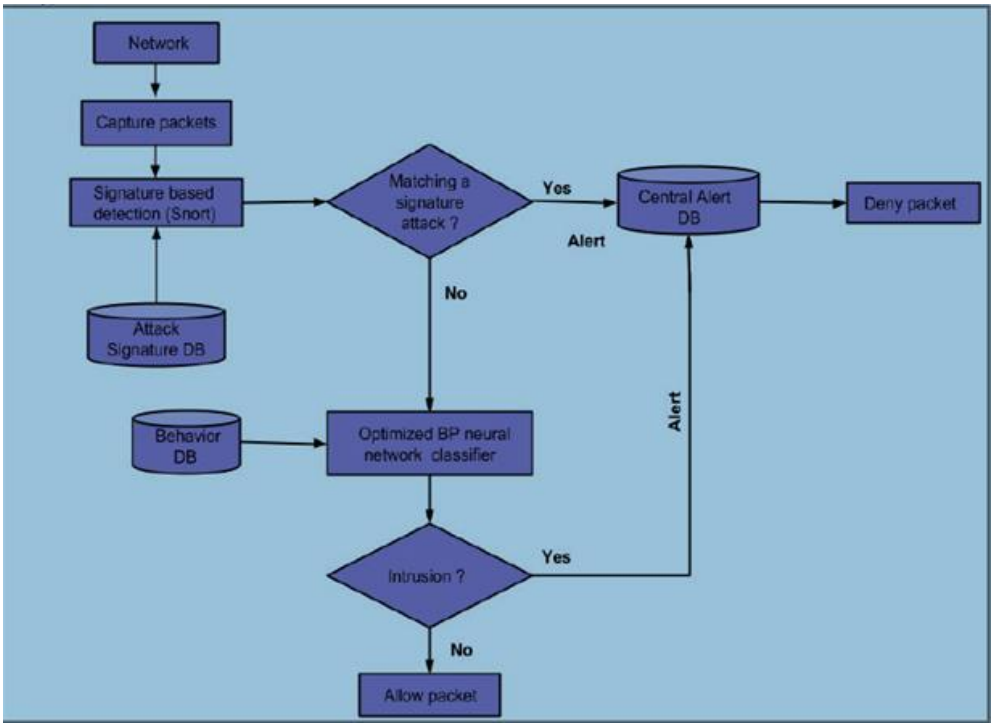

Fig-11 Workflow of anomaly detection model.

VIII. CONCLUSION

Cloud Computing has been growing tremendously for several years now. Day by day, more and more customers are going to be attracted towards this. The easiness and portability of the cloud computing services are really amazing. In near future, it will be an integral part of common man' s day to day life. However, skyrocketing success of this computational model highly depends upon the security of the services hosted upon it. This survey report extensively analyzed how to detect intrusion into the Cloud Infrastructure using Network Intrusion Detection Device such as Snort. It also discussed easiness in implementing snort and how the rules can be updated to defend a new attack. In addition to Snort, it tried to check the possibility of implementing a Neural Network based anomaly detection mechanism. No doubt, a successful implementation of above ideas will not only exponentially raise the security of the Cloud infrastructure, but also it will result in an exceptional intrusion detection mechanism with high accuracy of detection, very low false positives and negatives along with low computational cost. In addition, there should be more and more research conducted to find out vulnerabilities in the existing Cloud computing mechanism and to propose updated solution it.

IX. ACKNOWLEDGEMENT

I express my sincere gratitude to Dr. Dijiang Huang for providing an opportunity to explore my area of interest related to Cloud and OpenStack through Cloud Computing course. Also, I would like to thank our TA, Ankur Chowdhary for his guidance and continuous support through the course.

REFERENCES

1] https://www.nist.gov/itl/cloud-computing

2]https://www.ibm.com/blogs/cloud-computing/2014/02/cloud-computing-basics/

3]http://cseweb.ucsd.edu/~hovav/dist/cloudsec.pdf

4] Sip attacks from amazon ec2 cloud continue

5] https://www.snort.org/

6]https://www.duo.uio.no/bitstream/handle/10852/42149/m-hagen-master.pdf?sequence=7

7] https://cloud.google.com/compute/docs/

8]https://en.wikipedia.org/wiki/Eucalyptus_(software)

9] https://en.wikipedia.org/wiki/OpenStack

10]http://docs.openstack.org/liberty/networking-guide/intro-networking.html

11]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.688.2280&rep=rep1&type=pdf

12]https://myasucourses.asu.edu/bbcswebdav/pid-13673375-dt-content-rid-85412918_1/courses/2016Fall-T-CSE546-93699/Openstack.pdf

13]https://www.academia.edu/8645303/IDS_in_a_Openstack

14]https://www.sans.org/reading-room/whitepapers/detection/snort-distributed-intrusion-detection-system-352

15]http://www.ijsr.net/archive/v4i2/21021503.pdf

16] https://techanarchy.net/2015/01/home-ids-with-snort-and-snorby/

17]http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node28.html

18]https://www.duo.uio.no/bitstream/handle/10852/42149/m-hagen-master.pdf?sequence=7

19] Snort User Manual.
https://www.snort.org/documents

20]http://www.sciencedirect.com/science/article/pii/S1877050916302824