Incorporating Honeypot for Intrusion Detection in Cloud Infrastructure

Bhavesh Borisaniya, Avi Patel, Dhiren Patel, Hiren Patel

Incorporating Honeypot for Intrusion Detection in Cloud Infrastructure

Bhavesh Borisaniya[1], Avi Patel[2], Dhiren Patel[1], and Hiren Patel[3]

[1] NIT Surat, India
{borisaniyabhavesh, dhiren29p}@gmail.com
[2] City University London, UK
avi2687@gmail.com
[3] SPCE Visnagar, India
hbpatel1976@gmail.com

Abstract. Cloud services delivered as utility computing over the Internet makes it an attractive target for cyber intruders. Protecting network accessible Cloud resources and services from ever increasing cyber threats is of great concern. Most of the Network based Intrusion Detection System (NIDS) being rule based and therefore only capable of identifying known attacks (through pattern matching). Traditional Anomaly Detection based IDS may generate more number of false positives.
In this paper, we attempt to amalgamate IDS with Cloud computing. Introducing Honeypot in Cloud IDS design can greatly help in detecting potential attacks with reduced number of false positives. This research work provides an impetus to strengthen network security aspects related to Cloud computing to make it more trustworthy.

Keywords: Cloud Computing, Intrusion Detection System, Honeypot, Eucalyptus IaaS framework

1 Introduction

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (such as server, storage, applications, services etc.) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. It is a major aid for start-ups offering online applications and services without investing much in storage, Web, or computing infrastructure. Using known Internet protocols, standards and formats; Cloud computing exposes a set of consumable services delivered to end-users/consumers. These services range from computing utilities to platforms for application development.

1.1 Need for Security in Cloud

Cloud services are executed on the Cloud provider's site along with data. These require large amount of data to be transferred over the network. Cloud providers

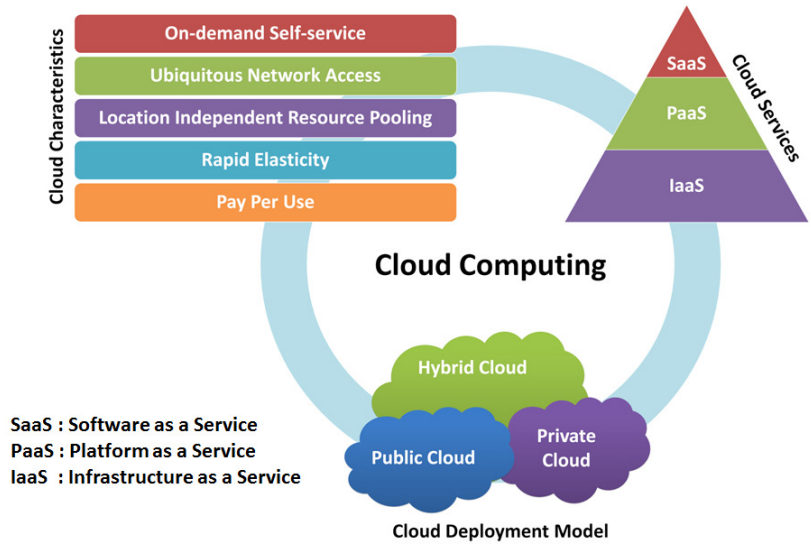

Fig. 1. Cloud computing

have to ensure about the quality of service, performance, reliability and basic security [2].

If an intruder gains unwanted access to Cloud services, he may also exploit the underlying architecture. In case of IaaS, the intruder may also be able to exploit the Virtual machine monitor (VMM) by using vulnerabilities in the implementation. Penetration to the hardware layer may allow an attacker to compromise any VM provided by the infrastructure.

Because of its provisioning through the Internet and vulnerabilities in involved (underlying) technologies; there are many issues related to security of Cloud infrastructure and its services. Major threats to Cloud computing includes insecure interface and APIs of shared technology, account and service hijacking [3] etc. Shared and distributed resources in the Cloud system make it difficult to develop a security model for detecting intrusion and ensuring the data security and privacy in Cloud. Because of transparency issue, no Cloud provider allows its customers to implement intrusion detection or security monitoring system extending into the management services layer providing back channel behind virtualized Cloud instances. IDS technology has been tested to be capable of working well in some large scale networks, however, its utilization and deployment in Cloud Computing is still a challenging task [2].

1.2 Intrusion Detection Systems (IDS)

Intrusion detection systems have proved to be a major tool for network administrators to protect their internal network from threats of cybercriminals and also

of internal threats. A Common Intrusion Detection Framework (CIDF) which illustrates a general IDS architecture, based on the consideration of four types of functional modules as shown in Figure 2 [4].

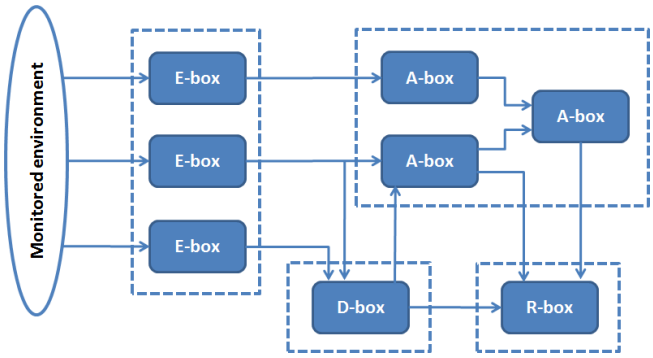

Fig. 2. Common Intrusion Detection Framework

The components of this IDS framework include the following:
E blocks (Event-boxes): These blocks contain sensor elements that monitor the target system and gather information events that can be analyzed by other blocks.
D blocks (Database-boxes): These are the blocks intended to store information from E blocks for subsequent processing by A and R boxes.
A blocks (Analysis-boxes): These are the processing modules which analyze the events and detect the potential hostile behaviour, so that some kind of alarm will be generated if necessary.
R blocks (Response-boxes): If any intrusion occurs, this block is responsible to provide a response to prevent the detected threat.

Network based IDS (NIDS) detects the intrusion by monitoring malicious activity in network traffic while Host based IDS (HIDS) inspects the unusual activity within the host by monitoring its file system.

Most of the NIDSs monitor network traffic and match it with the dataset of predefined attack patterns(signatures) to detect the attacks. For network intrusion detection, a signature can be as simple as a specific pattern that matches a portion of a network packet. However, signature-based technique fails to detect the unknown or new attacks whose signatures are not defined or not included in the dataset of signatures.

An anomaly based IDS establishes a baseline of normal usage patterns, and anything that widely deviates from it gets flagged as a possible intrusion [5]. A large number of false positives can limit this technique which can force it to sign even a genuine activity as an intrusion attempt.

The objective is to build efficient IDS which can work in Cloud environment with the capability of detecting known and unknown intrusions. The IDS must also prove to be a tool to the user to detect if the used-service or hosts are used to attack other victims.

The rest of the paper is organized as follows: In section 2, related work is reported. Section 3 discusses important design considerations for deploying IDS in Cloud. Section 4 discusses our approach to implement NIDS along with honeypot in Cloud framework. Section 5 describes implementation details using Eucalyptus Cloud framework and section 6 discusses experiments and results. We conclude in section 7 with references at the end.

2 Related Work

There has been relevant work done in the field about IDS for Cloud computing. The major approaches are listed as follows:

Sebastian Roschke et al. [2] points the need for deploying IDS in the Cloud by proposing extensible IDS architecture which can be used in a distributed Cloud infrastructure.

Noah Guilbault and Ratan Guha [6] shows a way for designing and implementing distributed grid based IDS using virtual servers deployed on Amazon's Elastic Compute Cloud service. Aman Bakshi et al. [7] proposed a framework for securing Cloud from DDoS attacks using an IDS in a virtual machine. This can be done by employing intrusion detection sensors installed in a virtual machine to sniff network traffic and to analyze packets over the Internet using Snort. Both these approaches incorporate IDS in each virtual machine, requires as many IDS as number of running virtual machine instances. Claudio Mazzariello et al. [8] has placed Snort as a NIDS on the virtual switch component of the physical machine. This physical machine hosts virtual machines of clients using open source Eucalyptus cloud computing framework. Virtual Switch enables the NIDS to monitor all in-bound and out-bound traffic from the entry-point. Chi-Chun Lo et al. [9] proposed a cooperative IDS framework for Cloud computing networks to reduce DDoS attacks. All these approaches use signature based technique, limited to detect only known attacks.

Kleber Vieira et al. [10] have described an intrusion detection based on Grid and Cloud computing system which can identify unknown as well as known attacks. However, this approach is only suitable for PaaS.

3 Design Considerations

Deploying IDS in the Cloud is a tricky issue. From the users' perspective, they need to make sure that the service they use is not subjected to any kind of attack. They should also know whether these services are being used to attack other hosts or not. On the other hand, Cloud providers need to ensure if its infrastructure is subjected to any attack or not. With knowledge of attacks and their behavior, the provider should be prompted to take appropriate actions.

In order to justify our approach and make it useful for a Cloud environment, we explored various Cloud frameworks for implementing IaaS as a service model and configure them for analysis in our lab environment. There are several open-source frameworks for Cloud computing viz; Eucalyptus [11], OpenNebula [12], Globus Nimbus [13] etc. Amongst them, we have zeroed into Eucalyptus because it provides simpler interface, supports different virtual machine monitors (or hypervisors) and modular architecture, which provides us an easy alternative to incorporate honeypot with IDS.

Network based intrusion detection tools are usually deployed over a perimeter of an organization network in order to monitor inbound and outbound network traffic. We have looked at various intrusion detection tools chosen Snort[14], as it is configurable, widely used and constantly updated. We have augmented the simple honeypot to a dynamic honeypot and incorporated it into our proposed approach. For testing the proposed architecture, we have setup a configurable private Cloud using Eucalyptus framework. We have created and tested the installed machine images of different operating systems which can be delivered as virtual machine instances with different configurations (RAM and CPUs) to the Cloud users in context of private Cloud.

3.1 Eucalyptus Architecture

Figure 3 shows the Eucalyptus Cloud architecture containing various components [15, 16]. Each high-level system component in the Eucalyptus design is implemented as a stand-alone Web service. These Web services expose a well defined language-agnostic API in the form of a WSDL (Web Service Descriptive Language) document, which contains operations that the service can perform and input/output data structures. It also support secure communications using WS-Security policies and rely upon industry-standard Web services software packages like Axis2, Apache and Rampart [15, 16].

Basic Components of Eucalyptus Architecture and their functions are summarized in Table 1.

Table 1. Eucalyptus Components

No	Component	Function
1	Cloud Controller(CLC)	High level Scheduling decisions
2	Node Controller(NC)	Management of Virtual machine instances and its execution
3	Cluster Controller(CC)	Scheduling of Virtual Machine execution on specific hosts and Virtual network management
4	Storage Controller(SC)	Storage of user data as well as storage service for Virtual Machine Images

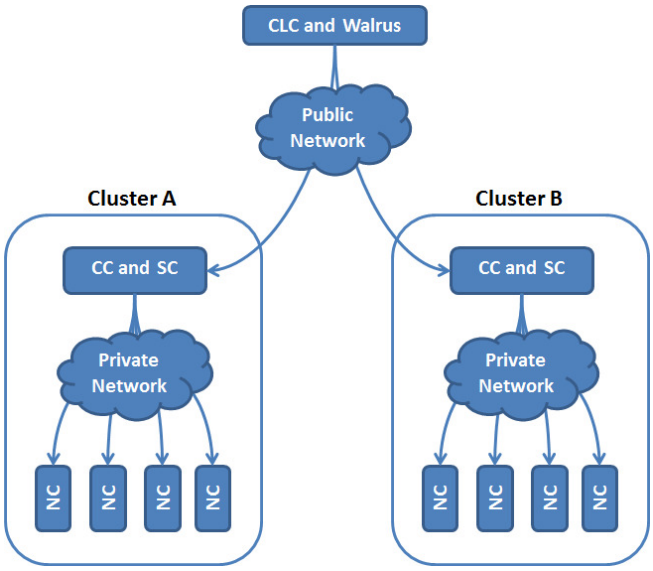

Fig. 3. Eucalyptus Cloud Architecture

Eucalyptus provides a functionality called security groups which acts as a firewall for running machine instances. It is a named collection of network access rules, defining which incoming traffic is delivered to instances. A user can add or remove a security group to meet his security requirement. By using this, a user can open or close ports to control the inbound or outbound network traffic over it. By restricting the number of open ports in an instance; a user can only decrease the probability of an attack to some extent. If the services running on these ports are vulnerable, then its easy for an intruder to exploit it.

3.2 Placement of a NIDS in Eucalyptus based private Cloud

Figure 4 shows the architecture of the system which comprises the NIDS. We have placed the NIDS in each Cluster Controller to monitor the network traffic of all the Node Controllers which report to the respective CC. NIDS will capture all the packets passing through the CC intended to the instances hosted by NCs and examine them for the malicious content. If malicious content is found, it generates alerts and logs that network activity into the central database. The Cloud administrator can view and analyze these logged alerts and network activities (i.e. packets) by accessing the database. Also, the owner of the instance can analyze the logged attack alerts related to it by querying the central database using interface through the instance.

This approach allows the Cloud administrator (Instance provider as well as instance owner) to monitor the type and source of the attack, which in turn can be used to prevent the similar future attacks.

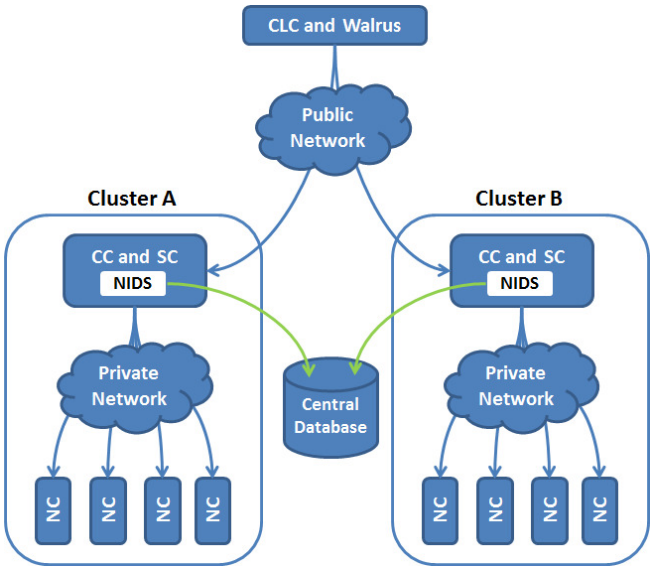

Fig. 4. Architecture of Eucalyptus Private Cloud with NIDS

3.3 Use of Honeypot

A honeypot is a deception system which allures the attackers. It has no production value and is intended to be compromised. All the traffic sent to a honeypot is almost certainly unauthorized meaning no false positives, false negatives or large data sets to analyze [17]. Any connection with honeypot can be considered as an attack and an attacker who breaks into a honeypot is comprehensively monitored. Honeypots are serving several purposes that include the following [17]:

1. They can distract attackers from more valuable machines on a network.
2. They can provide early warning about new attack and exploitation trends.
3. They allow in-depth examination of adversaries during and after exploitation of a honeypot.

In our approach, honeypot plays an important role. Any attempt to access honeypot is labeled as an attack.

4 Proposed Approach: Incorporating a Honeypot in Eucalyptus based private Cloud

Unknown attacks, for which signatures are not available, have to be dealt with caution and it requires a more efficient IDS mechanism. In our approach, we have incorporated a honeypot to enhance the working of a NIDS in a Cloud

environment. The introduction of a honeypot allows identification of suspicious activities by monitoring those network packets which were previously marked as non-suspicious by a normal NIDS. In order to deploy a honeypot in a Cloud environment, we have considered a design in which the administrator launches instances through the honeypot manager. Honeypot manager is an administrative tool used for managing honeypot instances, which are made vulnerable and attractive for intruders to exploit (by running various services accessible through the Internet through open ports).

Figure 5 shows the NIDS incorporating a honeypot in a Cloud architecture. We have shown two such machine instances which work as honeypot. These instances have no production value and hence any inbound or outbound network activity with these instances is considered as malicious. Any packet passing through CC intended for honeypot machine instances will be captured by a packet sniffer and logged into the central database for later analysis.

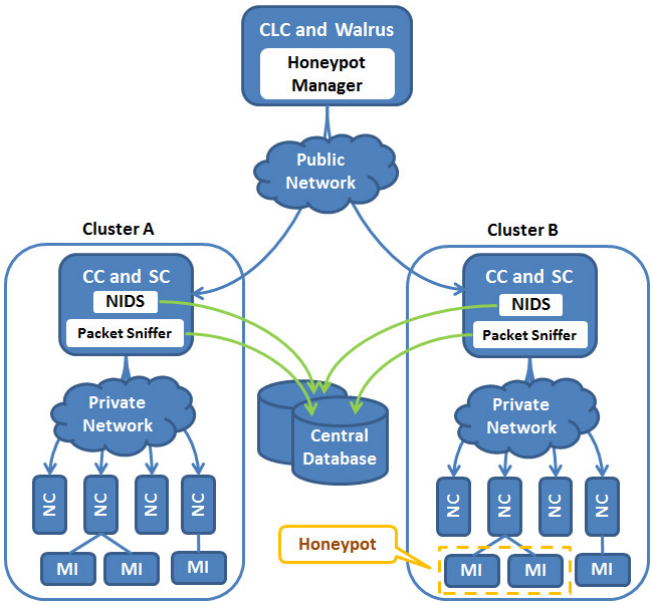

Fig. 5. Architecture of intrusion detection system using honeypot in Eucalyptus Cloud

Here, NIDS is placed in CC to listen the traffic intended for instances and generates (and logs) the alerts to the central database if any malicious activity is found. A packet sniffer sniffs and logs all network packets related to the honeypot instances in the central database.

All packets intended towards honeypot instances, also passes through NIDS. As cloud provider is not delivering honeypot instances to any client, any activity towards it can be considered as malicious. Hence, by querying the database for

activities which are captured by honeypots and passed through NIDS, we are able to find such activities/attacks, which were not detected earlier. From these logged activities, we can find information like source (IP) of attack and services they are trying to access using destination and source port. Figure 6 depicts this intrusion detection process flow diagram.

The compromised honeypot instance image can also be used as a means to learn new ways, tools and methods to get into the system. It is also helpful to understand the motive of attacker, to avoid the future attacks and to make the existing NIDS more efficient.

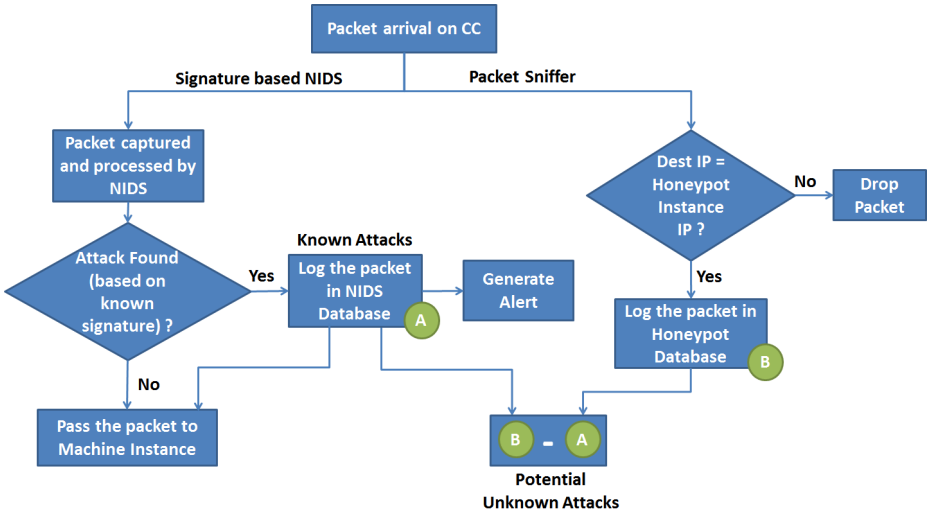

Fig. 6. Intrusion detection process in proposed framework

The compromised honeypot instance image can also be used as a means to learn new ways, tools and methods to get into the system. It is also helpful to understand the motive of attacker, to avoid the future attacks and to make the existing NIDS more efficient.

5 Implementation Issues

Figure 7 shows the experimental setup of a Eucalyptus private Cloud with the proposed framework of NIDS.

Our setup consists of three machines, a Node Controller (NC), a Cloud Controller (CLC) and a machine consisting of both Storage Controller (SC) and Cluster Controller (CC). We have used a separate cluster controller (i.e. SC and CC) and database server (i.e MySQL) on different machines. They can also be

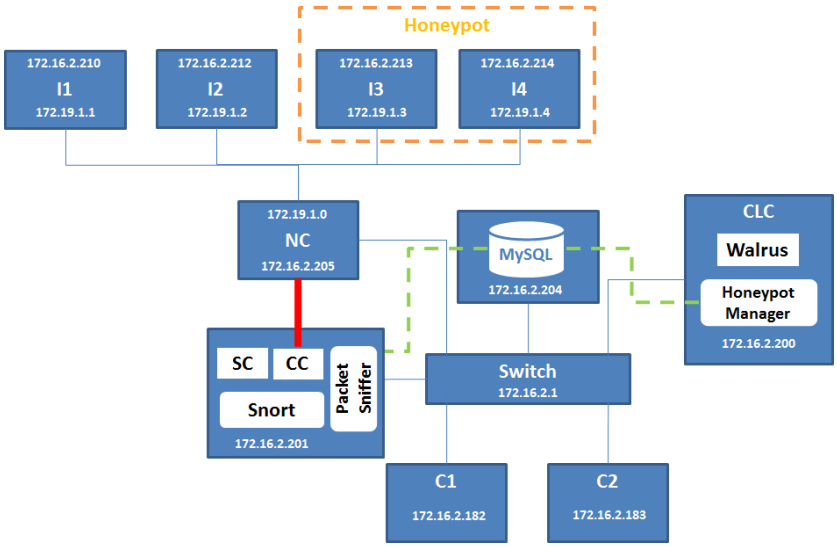

Fig. 7. Experimental Setup for NIDS using Honeypot in Cloud

placed on the same machine in which the CLC resides. I1, I2, I3 and I4 are four machine instances, made available over an external network as a Cloud service model i.e. IaaS, whereas C1 and C2 are clients utilizing these services from the Cloud.

Snort is configured in the CC machine along with a packet sniffer, while the Honeypot Manager is placed in the CLC. A set of machine images of different OS is used to create the honeypot in our environment. Under ideal circumstances, it should cover machine images of all the OS, whose instances are provided by the Cloud provider. These images are made in a way that can attract the attackers. In our experiments we have used machine images of Windows XP, Windows Server 2003 and Ubuntu 10.04. The central database contains the information of machine images that can be used to launch as honeypot instances. It also maintains the information of the fake Cloud users (for experimentation) like their username and credentials that can be used to run the honeypot instances with different instance ownership.

The honeypot manager is responsible for launching the honeypot instances. It gathers the information about the operating system and the state of open and closed ports for different services for all instances. Accordingly, the honeypot manager schedules the type of operating system to be used as a honeypot. The honeypot manager launches each machine instance of a different OS having the ownership of fake users. It also opens different ports for vulnerable services through a security group mechanism to make it more attractive. These machine instances are not delivered to normal users and also its owners credentials are only with the Cloud administrator (nobody can access these instances directly).

Hence, any connectivity with these machine instances can be considered malicious.

A packet sniffer captures all the network packets which pass through the CC whose source or destination IP is one of the honeypot IPs. It logs all the captured packets into the central database. Snort examines those packets that are sniffed by the packet sniffer and generates alerts if it finds a known attack pattern within the packet content.

Controlled monitoring of vulnerable images can assist the behavioral analysis of an intruder. Vulnerable images are crafted to exploit with absolute zero or no security and placed in a DMZ. These images are monitored periodically for intrusion attempts. In case of an intrusion, an alert is reported on the primary basis. A local cache directory can be included on top of the Cloud architecture which saves copies of vulnerable images periodically on a version basis. Careful analysis can be done of these vulnerable images to record intruder activity and to enhance the security of the Cloud architecture there after using appropriate security measures.

6 Experiments and Results

We have used Snort as a NIDS and launched vulnerable machine instances to work as honeypots. In order to compare the vulnerability in both original as well as vulnerable copies of operating system images, we conducted a scan using Nessus by enabling all available plug-in modules. The statistics collected are shown in Table 2.

Table 2. Nessus vulnerability scan result of machine images

Operating System	Machine Instance	Total	High	Medium	Low	Open Port
Windows XP	Original	20	0	2	14	4
	Vulnerable	48	0	5	29	14
Windows 2003 Server	Original	11	0	1	8	2
	Vulnerable	32	5	1	18	8
Ubuntu 10.04	Original	21	0	0	17	4
	Vulnerable	64	1	2	43	18

We launched two instances of each operating system (i.e. Windows XP, Windows 2003 and Ubuntu 9.10) in the Cloud and opened the required ports for the services that run on different operating system instances using security group. Honeypot manager also launches instances of the vulnerable images relative to these operating systems.

We attacked all the nine machine instances (3 for each operating system - 2 normal and 1 launched by the honeypot manager) using Nmap, Nessus and Metasploit to test the designed system. Nmap scans all the open ports while

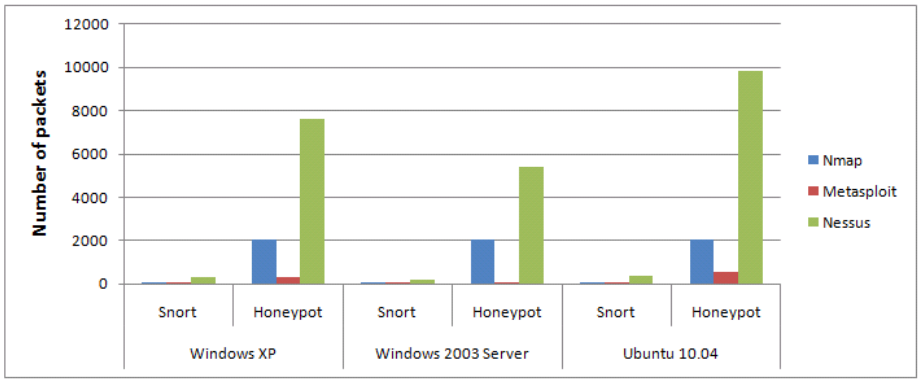

Fig. 8. Comparison of total number of packets logged by Snort and Honeypot in Cloud environment

Nessus and Metasploit send bad packets in order to find vulnerabilities and exploit them. Then we compared the alerts generated by Snort and honeypot in response to the attacks made by each tool.

Figure 8 shows the graphical representation for the comparison of the total number of logged packets by the honeypot and Snort.

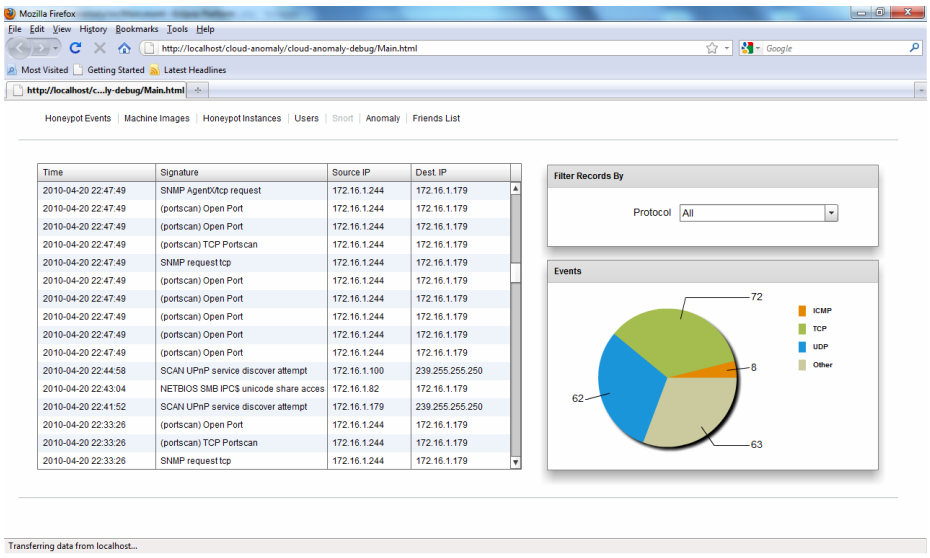

Fig. 9. Web application(screen shot) for analyzing proposed system

In order to verify the results, we formally developed a web application which gives the details of intrusion attempts logged by Snort as well as the honeypot. Screen shot of this web application is shown in Figure 9.

7 Conclusion

By incorporating honeypot, the proposed IDS for Cloud not only alerts the users (about possible network attacks) but also helps Cloud administrator to monitor unknown attacks to enhance its Intrusion Prevention Strategy. The proposed system can be implemented in a Cloud environment to make it more trustworthy by providing an intrusion alert mechanism for attacks against Cloud.

The core benefits of the proposed approach are:

1. It can detect known as well as potential unknown attacks.
2. Controlled use of honeypot generates less number of false alarms for unknown attacks making it an efficient solution for intrusion detection specific to private Cloud.

Though, the proposed scheme is implemented with a Eucalyptus framework, it may work well for other Cloud platforms. It can serve as model to study the behavior of NIDS for distributed environment.

References

1. Grance, T., Mell, P.: The nist definition of cloud computing. National Institute of Standards & Technology (NIST) (2009). URL `http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf`
2. Roschke, S., Cheng, F., Meinel, C.: Intrusion detection in the cloud. Dependable, Autonomic and Secure Computing, IEEE International Symposium on **0**, 729–734 (2009)
3. Top threats to cloud computing (2009). URL `https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf`
4. Garca-Teodoro, P., Daz-Verdejo, J., Maci-Fernndez, G., Vzquez, E.: Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers and Security **28**, 18–28 (2009)
5. Marinova-Boncheva, V.: A short survey of intrusion detection systems. Problems of Engineering Cybernetics and Robotics (2007). URL `http://www.iit.bas.bg/PECR/58/23-30.pdf`
6. Guilbault, N., Guha, R.: Experiment setup for temporal distributed intrusion detection system on amazon's elastic compute cloud. In: Intelligence and Security Informatics, 2009. ISI '09. IEEE International Conference on, pp. 300 –302 (2009)
7. Bakshi, A., Dujodwala, Y.B.: Securing cloud from ddos attacks using intrusion detection system in virtual machine. Communication Software and Networks, International Conference on pp. 260–264 (2010)
8. Mazzariello, C., Bifulco, R., Canonico, R.: Integrating a network ids into an open source cloud computing environment. In: Sixth International Conference on Information Assurance and Security (IAS), 2010, pp. 265 –270 (2010)

9. Lo, C.C., Huang, C.C., Ku, J.: A cooperative intrusion detection system framework for cloud computing networks. In: Proceedings of the 2010 39th International Conference on Parallel Processing Workshops, ICPPW '10, pp. 280–284. IEEE Computer Society (2010)
10. Vieira, K., Schulter, A., Westphall, C., Westphall, C.: Intrusion detection for grid and cloud computing. It Professional **12**(4), 38–43 (2010)
11. Eucalyptus. URL `http://www.eucalyptus.com/`
12. Opennebula. URL `http://www.opennebula.org/`
13. Nimbus. URL `www.nimbusproject.org/`
14. Snort, network intrusion detection and prevention system. URL `http://www.snort.org/`
15. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, pp. 124–131. IEEE Computer Society (2009)
16. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: A technical report on an elastic utility computing architecture linking your programs to useful systems. open.eucalyptus.com (2008)
17. Mokube, I., Adams, M.: Honeypots: concepts, approaches, and challenges. In: Proceedings of the 45th annual southeast regional conference, ACM-SE 45, pp. 321–326. ACM (2007)