Intrusion Detection System for Cloud Environment

Hifaa Bait Baraka
School of Engineering and Built Environment
Glasgow Caledonina University
Glasgow, UK
networkeng83@gmail.com

Huaglory Tianfield
School of Engineering and Built Environment
Glasgow Caledonina University
Glasgow, UK
h.tianfield@gcu.ac.uk

ABSTRACT

Cloud computing environment is threatened by different types of cyber-attacks. This paper presents an implementation of intrusion detection system to secure virtualized servers on the cloud platform and validates intrusion detection system in detecting DDoS attack against the virtualized environment.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Systems and applications—*operating systems, applications, virtualization*; D.4.6 [**Security and Protection**]: Security—*threats, rules, alerts*

General Terms

Experiment, Security

Keywords

Cloud Computing; Intrusion Detection System (IDS); Virtualization; Kernel-based Virtual Machine (KVM); Snort

1. INTRODUCTION

The Cloud Computing infrastructure stores the application software and data. The concern of users and businesses is availability of data and applications, rather than comprehension of the complexities involved in delivering services. Cloud applications are accessible through any device, including a laptop, cell phone or smart phone, which is capable of connecting to the internet.

Virtualization is an essential technique in cloud computing, providing a resource infrastructure for cloud clients; it delivers the resources by deploying virtual machines over the virtual machine monitor, also known as the hypervisor. Virtualization is considered the main concept in the cloud because it manages the complexity through virtualizing hardware and software.

There are various threats that can affect the virtualization in the cloud environment like DDoS attack, therefore securing hypervisor and virtual machines in the cloud is important for protecting sensitive data from any intrusions. One of the methods has been used to monitor and detect threats and attacks against either the cloud or virtualized server is intrusion detection . Intrusion detection is considered a strong mechanism which plays a vital role in securing networks. However, virtualized server in cloud environment carries a huge amount of traffic, therefore implementing Intrusion Detection System (IDS) in a cloud environment requires scalable and virtualized infrastructure [3]. The reminder of the paper is arranged as follows. Section 2 discusses intrusion detection techniques and methods for cloud environment. Section 3 presents an overview of IDS in cloud environment. Section 4 discusses the design for implementing IDS in the cloud environment; then Section 5 presents experiment and evaluation. The last section sums up the paper.

2. INTRUSION DETECTION TECHNIQUES AND METHODS FOR CLOUD COMPUTING

The cloud computing environment is an easy target for intruders who search for vulnerabilities in the cloud system that can be exploited. These intrusions create problems for cloud confidentiality, integrity and availability. Threats that affect the cloud environment cause a security breach in virtual machines as well.

If DDoS attack occurs against the virtualized server in the cloud, then attacker can stop services that are provided by the server. Then availability of the data can be lost and authorized users cannot access their data and services. A common method to detect this type of threats is intrusion detection.

2.1 Types of IDS for Securing Virtualized Server in Cloud

2.1.1 Host-based IDS (HIDS)

This is intrusion detection software used to monitor the host machine. This type of system is responsible for collecting all inbound and outbound traffic from the user workstation and sends an alert when a suspicious event occurs [4]. HIDS can be installed on virtual machines or hypervisors, in which HIDS monitors log files, access control policies and user login information to detect any abnormal behaviors [4].

SIN '14, September 09 - 11 2014, Glasgow, Scotland Uk.
Copyright 2014 ACM 978-1-4503-3033-6/14/09 ...$15.00.
10.1145/2659651.2659682

2.1.2 *Network-based IDS (NIDS)*

Network-based intrusion detection is used to monitor network traffic instead of host traffic; thus it detects attacks by capturing and analyzing network packets. It is responsible for listening to network segments and protecting the segments and the hosts connected to them. NIDS is effective for the cloud, detecting intrusion events by comparing observed behavior in real time.

NIDS can be implemented on the cloud server which interacts with the external network (user network) to detect attacks against virtual machines and hypervisor. NIDS detects attacks by inspecting the IP and transport layer headers of each packet. One of the limitations of NIDS is that it may not be useful if an attack occurs within a virtual network which runs inside the hypervisor [4].

2.1.3 *Hypervisor IDS*

This is designed and used for hypervisors only; it also monitors communication between virtual machines, communication between virtual machines and the hypervisors, or communication within the hypervisor. The advantage of hypervisor IDS is that it provides information availability[4].

2.2 Distributed IDS (DIDS)

Used in large networks, a distributed intrusion detection system consists of several IDS which can be network-based or host-based. Each IDS in the large network communicates with the other IDS or with a central server for analysis of events. Therefore, in the cloud environment IDS collects data from the network and hosts and converts them to standard form; then IDS sends the converted data to the centralized analyzer. DIDS takes advantage of the ability of both , NIDS and HIDS; and thus can detect known and unknown attacks.

In the cloud environment, DIDS can operate in user host machines or at the backend of the processing server [4].

2.3 Intrusion Detection Methods

2.3.1 *Signature-based detection*

Signature-based detection is also known as Pattern-based detection, which detects attacks based on the signature. In general, the signature-based detection method is not used to detect latest attacks because no matched rules or patterns have yet been configured. This type of detection method can be used in the host based or network based IDS. Therefore, in the cloud environment, signature-based detection can be used in virtual machines, hypervisors or virtual networks to monitor the activities and detect known attacks [5].

2.3.2 *Anomaly-based detection*

The anomaly-based detection method is used to identify abnormal behavior (anomalies) in the host machines, network segments or devices. At the start, anomaly-based detection constructs a clear view of the normal behavior of users, hosts or network segments, then it sends an alert if new events occur that contradict the normal behavior. In the cloud environment, anomaly-based detection uses different models to determine unusual behavior such as threshold detection, statistical model, rule-based model, and other models, including neural networks, genetic algorithm, and immune system model [5].

2.3.3 *Honeypot-based detection*

This is a computer or server which is used to attract attackers in order to monitor their behavior and check their methods of breaking in a computer system. This type of detection technique, like other detection methods, is used to protect data and resources of cloud, but the difference is that in the honeypot mechanism a fake server is used behind a firewall to attract hackers. The drawback of this method is that no all-time attackers want to attack fake servers[1].

3. OVERVIEW OF IDS IN CLOUD ENVIRONMENT

C. Modi et al. [3] proposed an architecture for a network intrusion detection system that can be used in cloud computing. The proposed network-based IDS (NIDS) consists of three components, i.e., Packet Preprocessing, Analyzer and Storage. The packet preprocessing prepares captured packets for detection. The analyzer implements two detection methods on captured packets, namely signature-based and anomaly-based technique. The NIDS uses the Bayesian classifier for anomaly detection and Snort to detect known attacks by a signature-based method. The analyzer contains an alert log module that sends NIDS logs and alerts to another server. The storage is used to store alerts logs and cloud events. There are two types of storage. The first type is knowledge storage, which collects the Snort rules used to detect known attacks. The second type is behaviour storage, which stores normal and non-normal events in the cloud network used by the Bayesian classifier.

Using both detection techniques (Signature and Anomaly) in NIDS enhances detection accuracy and reduces time. The NIDS is applied on Eucalyptus cloud and is installed in each node controller (NC). The NIDS can be used in front end, back end and virtual machine in the cloud platform with high detection accuracy. To speed up the detection process, the cloud provider should decide on the right place to deploy the IDS. The NIDS can be deployed in the front end of the cloud to detect intrusions occurring in the external network, while deploying it in the back end of the cloud detect attacks occurring in both internal and external networks. Implementing the NIDS on cloud virtual machines helps to detect attacks occurring in the virtual machines only.

The NIDS can reduce false positive and false negative alarms in the cloud environment by using two detection techniques that complement each other. The signature based technique can detect known attacks by matching the captured packet with the rules stored in the knowledge storage, while the anomaly based technique collects normal events in behaviour storage and applies statistical tests on the observed actions to determine whether or not the behaviour is normal. Finally, deploying the NIDS in the back end of the cloud is the most effective solution to detecting attacks against cloud from external networks.

J. Lee et al.[6] proposed a multi-level IDS with a log management method based on user behaviour. The main principle of multi-level IDS is to provide security while minimizing resource wasting and lessening the need to analyze a huge number of logs. The multi-level IDS monitors cloud user behaviour based on the anomaly detection technique, which places cloud users in different security groups based on each user's level known as the anomaly level. The identity of user is checked by an AAA server; after the user is authenticated then an anomaly level is generated to be assigned to that user.

The AAA server is used to select the IDS that is suitable for that user by virtue of having the security level corresponding to the anomaly level of user. The security levels are divided into three: High, providing strong security service, Medium, providing medium security strength, and Low, providing a low level of security. The IDS is anomaly behaviour based, in that it determines the risk based on the policy level.

P.Shelke et al.[7] proposed multi-threaded network intrusion detection to manage the huge amount of data in the cloud environment. This approach is proposed for a distributed cloud platform. It comprises three units: (i) Capture and queuing unit: The capture module captures inbound and outbound traffic which is sent to the queuing module to be analyzed. (ii) Analysis and process unit: The queuing module sends traffic to the analysis unit to be examined based on configured signatures and rules. Each process in the queuing unit has multiple threads to enhance the performance of the NIDS. (iii) Reporting unit: If the captured traffic matches any of the configured signatures, it means that intrusion traffic is identified and alerts are produced. The reporting unit reads the generated alert from the queuing unit and produces an alert report.

Multi-thread NIDS sends alerts to a third party which is responsible for notifying the cloud users about the attack against their system. At the same time, the third party can assist the cloud provider in any misconfiguration.

A. Bakshi et al.[2] proposed installing i.e., (Snort) on a VMware virtual ESX machine that captures inbound and outbound traffic to be audited by a database module. Then Denial of Service (TCP SYN) attack is simulated from an intruder machine to the target machine; Snort captures the arriving packets on Ethernet and analyzes them. When Snort detects the DoS TCP SYN attack based on the configured rule then Snort drops the packets from the intruder machine's IP address and sends alerts to administrator machine.

Snort is an open source intrusion detection tool using a rule-based method, which combines signature, protocol and anomaly detection techniques. In the cloud environment Snort uses detection methods at different levels, with Snort specifying the nature of the attack and notifying the virtual servers, so that the notified server will drop the packets coming from the IP address determined by Snort. In eucalyptus cloud, Snort is used in the cloud controller and virtual machines to detect any attacks; this solution is fast and cost-effective, with no need to deploy multiple IDSs [4].

4. IMPLEMENTING IDS IN THE CLOUD

There are different types of IDS, but in general the most popular types are host-based and network-based IDS. The host-based IDS is used in users' workstations, where as the network-based IDS can be deployed at different points in the cloud environment.

4.1 Positions in the Cloud To Deploy IDS

There are different places at which IDS can be deployed, as shown in Figure 1. IDS in the cloud can be deployed at the front end, at the back end or on the virtual machines.

Implementing IDS at the front end of cloud will detect attacks on the end user network, where deployment of IDS is not useful in detecting internal attacks [3]. Implementing IDS at the back end of the cloud environment (at server point) will detect all internal attacks on cloud and all external attacks which come from the end user network [3]. Implementing IDS on virtual machines (VM) within the cloud environment will detect attacks on those machines only [3].

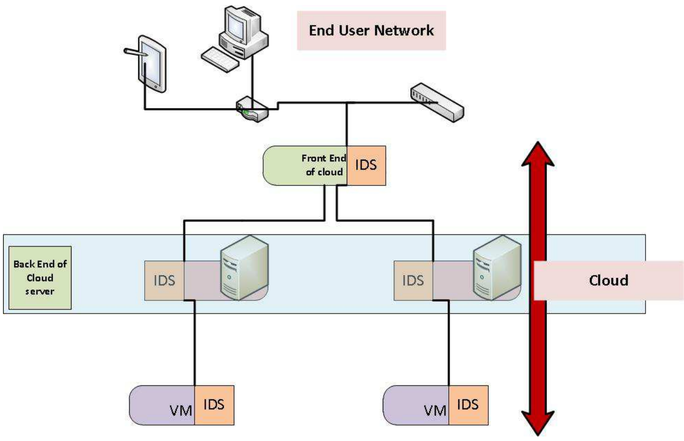


Figure 1: Positions in the cloud at which IDS can be deployed

4.2 Snort

Snort is powerful tool that can be used to monitor events in virtualized server in the cloud and detect attacks. Snort has different modes which can be used but the main function of Snort for IDS in networks is capturing all incoming packets, analyzing these packets and finally giving alerts if a packet is matching the configured rule. Figure 2 shows the flowchart of Snort.

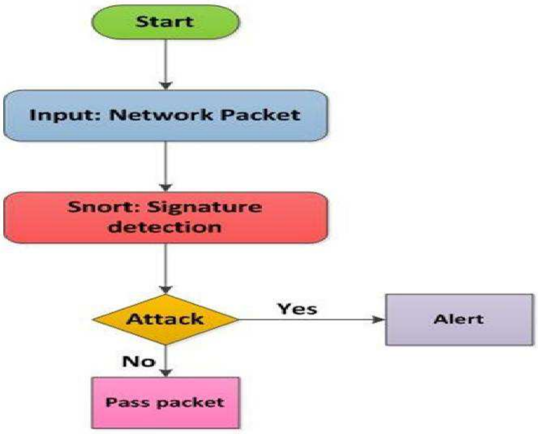


Figure 2: Flowchart of Snort

Snort has three modes, namely, logging mode, sniffer mode, and NIDS mode. In the Logging mode, every packet will be logged into log folder and using this mode is not useful. The sniffer mode prints TCP/IP packet header to the screen. The NIDS mode will create rules based on the administrator policies. This mode sent alerts to syslog server to be seen by the administrator.

Table 1: System specifications

Item	Spec
Model	Toshiba (L50-A-1FD)
Series	Satellite L50-A
Processor Type	Core i7-4700MQ (x86 processor)
Processor Speed	2.4 GHz
RAM Size	8 GB
Hard Disk	1 TB

If Snort gives alert that an attack occurs from known network, the administrator should shut down the connection with that network. Therefore, Snort can be used as security method to detect any attacks against virtualized environment.

5. EXPERIMENT OF IDS IN CLOUD

In this paper, a real experiment was conducted by building virtualized server in cloud platform, then implementing the intrusion detection system. The system specifications are outlined in Table 1.

5.1 Setup of Virtualized Servers

To build the virtualized server, the following requirements were considered:

5.1.1 Physical machine (Specifications)

The important factors of the physical machine that was used in this experiment are the processor type, RAM size and hard disk size. Processor type determines whether the physical machine supports virtualization or not. The RAM and hard disk should be large enough to avoid physical machine crashing because the host machine runs different types of virtual machines. In other words, RAM and hard disk should be large enough to run different platforms on one physical machine.

As shown in Table 1 processor type of the host machine is core i7-4700MQ, which supports virtualization technology, and the sizes of both RAM and Hard disk are enough to avoid physical machine crashing and increase the performance of virtual machines on the physical machine. Another factor should be considered in the physical machine which increases the transaction speed between virtual machines is the processor speed. As shown in Table 1 the processor speed of the experiment physical machine is 2.4 GHz, which is the minimum speed required in virtualization.

5.1.2 Physical machine operating system

To install Kernel-based Virtual Machine (KVM) and build the virtualized server, the operating system of the physical machine uses Ubuntu Desktop 13.04.

5.1.3 Hypervisor

Hypervisor is a program that allows multiple operating systems to share a single piece of hardware. In our experiment Kernel-based virtual Machine (KVM) was installed on Ubuntu OS. To manage the virtual machines, a tool known as virtual manager was installed. Then three virtual machines were created on top of the KVM, namely, FTP server, web server, and desktop server.

On the FTP server which is on Ubuntu virtual machine, cloud user can take backup file. On the Web Server which is on windows server 2008 virtual machine, cloud user can access different websites that are created. On the Desktop server which is on windows 7 ultimate virtual machine, cloud user can remotely access desktop placed on the virtualized platform in the cloud using remote desktop Protocol (RDP) which uses software known as Remote Desktop Connection.

5.2 Network Setup

Connection between the virtualized servers and external users were set up as shown in Figure 3. External users use public IP addresses to access the virtualized servers and the private IP addresses are known by the cloud provider only. Nat feature was configured to map public IP addresses to private IP addresses.

Then Snort was used on separate windows machine to detect any attack against the virtualized servers. Snort in this topology is considered to be as network IDS in the back-end of the cloud environment, so threats can be detected in virtualized servers.

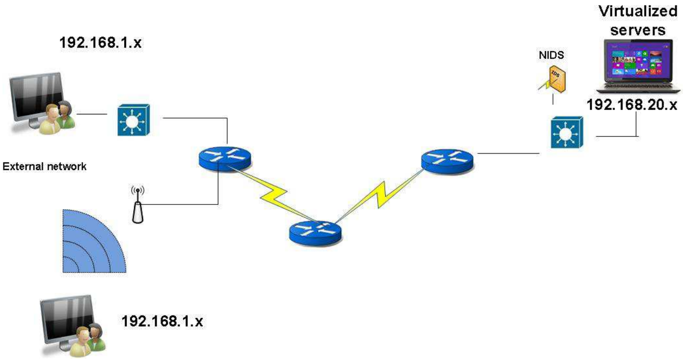


Figure 3: Network setup

5.3 Experiment Validation

5.3.1 Virtualized server validation

The function of the virtulized servers was validated by accessing the servers from external users. One example to validate the function of these servers is shown in Figure 4 in which user accessed ftp server and downloaded file into his machine.

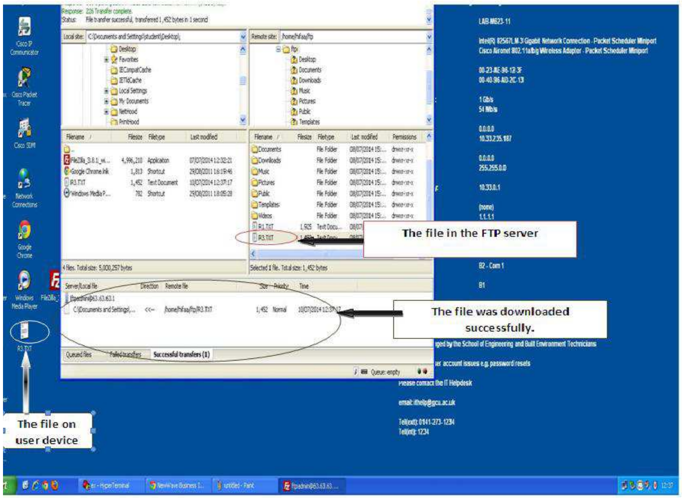
5.3.2 Snort validation

SPAN port was configured on the switch port that Snort was connected to. This SPAN port sent copy of each incoming packet to the Snort. Then Snort can work on incoming packet based on the mode used. To validate the Snort function, one rule was configured into Snort configuration file. The rule function is to give alert if external user ping any visualized server. As shown in Figure 5 the Snort sends alert to syslog server on the same machine to alert the administrator based on the configured policy.

Table 2 elaborates the parameters formatted of a Snort alert (the first line in Figure 5)

5.4 Snort IDS Evaluation

To evaluate Snort, DDoS attack was simulated against the virtualized servers. Low Orbit ION Canon (LOIC) DDoS tool was used to simulate the attack. This tool has three



```
Response: 226 Transfer complete.
Status:   File transfer successful, transferred 1,452 bytes in 1 second
```

Local site: C:\Documents and Settings\student\Desktop\
Remote site: /home/hifaa/ftp

```
Filename               Filesize  Filetype       Last modified
FileZilla_3.8.1_wi...  4,996,210 Application    07/07/2014 12:32:21
Google Chrome.lnk      1,813     Shortcut       29/08/2011 16:19:46
R3.TXT                 1,452     Text Document  10/07/2014 12:37:17
Windows Media P...     782       Shortcut       29/08/2011 18:05:28
4 files. Total size: 5,000,257 bytes
```

```
Filename   Filesize  Filetype     Last modified      Permissions
Documents            File Folder  08/07/2014 15:...  drwxr-xr-x
Downloads            File Folder  08/07/2014 15:...  drwxr-xr-x
Music                File Folder  08/07/2014 15:...  drwxr-xr-x
Pictures             File Folder  08/07/2014 15:...  drwxr-xr-x
Public               File Folder  08/07/2014 15:...  drwxr-xr-x
Templates            File Folder  08/07/2014 15:...  drwxr-xr-x
Videos               File Folder  08/07/2014 15:...  drwxr-xr-x
R1.TXT     1,925     Text Docu... 08/07...
R3.TXT
Selected 1 file. Total size: 1,452 bytes
```

```
Server/Local file                 Direction  Remote file              Size   Priority  Time
ftpadmin@63.63.63.1
C:\Documents and Settings\...     <<--       /home/hifaa/ftp/R3.TXT   1,452  Normal    10/07/2014 12:37:17
```

Queued files | Failed transfers | Successful transfers (1)

Queue: empty

LAB-M623-11

Intel(R) 82567LM-3 Gigabit Network Connection - Packet Scheduler Miniport
Cisco Aironet 802.11a/b/g Wireless Adapter - Packet Scheduler Miniport

00-23-AE-96-12-3F
00-40-96-AD-2C-13

1 Gb/s
54 Mb/s

0.0.0.0
10.33.235.187

0.0.0.0
255.255.0.0

10.33.0.1

(none)
1.1.1.1

B2 - Com 1

B1

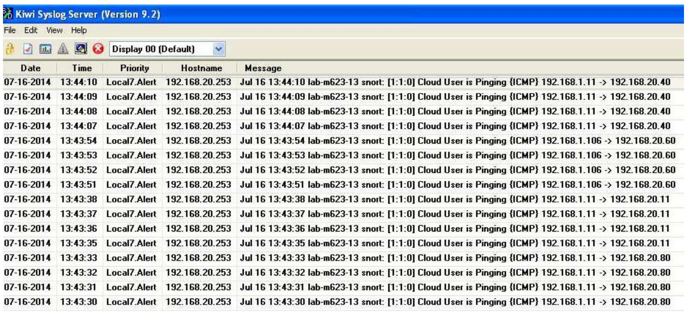
...ged by the School of Engineering and Built Environment Technicians

...er account issues e.g. password resets
Please contact the IT Helpdesk

email: ithelp@gcu.ac.uk

Tel(ext): 0141-273-1234
Tel(int): 1234

Figure 4: Virtualized server validation



Kiwi Syslog Server (Version 9.2)

```
Date        Time      Priority      Hostname         Message
07-16-2014  13:44:10  Local7.Alert  192.168.20.253   Jul 16 13:44:10 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.40
07-16-2014  13:44:09  Local7.Alert  192.168.20.253   Jul 16 13:44:09 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.40
07-16-2014  13:44:08  Local7.Alert  192.168.20.253   Jul 16 13:44:08 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.40
07-16-2014  13:44:07  Local7.Alert  192.168.20.253   Jul 16 13:44:07 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.40
07-16-2014  13:43:54  Local7.Alert  192.168.20.253   Jul 16 13:43:54 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.106 -> 192.168.20.60
07-16-2014  13:43:53  Local7.Alert  192.168.20.253   Jul 16 13:43:53 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.106 -> 192.168.20.60
07-16-2014  13:43:52  Local7.Alert  192.168.20.253   Jul 16 13:43:52 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.106 -> 192.168.20.60
07-16-2014  13:43:51  Local7.Alert  192.168.20.253   Jul 16 13:43:51 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.106 -> 192.168.20.60
07-16-2014  13:43:38  Local7.Alert  192.168.20.253   Jul 16 13:43:38 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.11
07-16-2014  13:43:37  Local7.Alert  192.168.20.253   Jul 16 13:43:37 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.11
07-16-2014  13:43:36  Local7.Alert  192.168.20.253   Jul 16 13:43:36 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.11
07-16-2014  13:43:35  Local7.Alert  192.168.20.253   Jul 16 13:43:35 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.11
07-16-2014  13:43:33  Local7.Alert  192.168.20.253   Jul 16 13:43:33 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.80
07-16-2014  13:43:32  Local7.Alert  192.168.20.253   Jul 16 13:43:32 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.80
07-16-2014  13:43:31  Local7.Alert  192.168.20.253   Jul 16 13:43:31 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.80
07-16-2014  13:43:30  Local7.Alert  192.168.20.253   Jul 16 13:43:30 lab-m623-13 snort: [1:1:0] Cloud User is Pinging {ICMP} 192.168.1.11 -> 192.168.20.80
```

Figure 5: Snort validation

flooding methods: UDP, TCP and HTTP. The attack method, the target IP address and the port number are illustrated in Figure 6. Before DDoS attack was simulated, a number of rules were configured on the Snort to give alert if it detected any attack. The example below shows one of the rules that were configured in Snort configuration file.

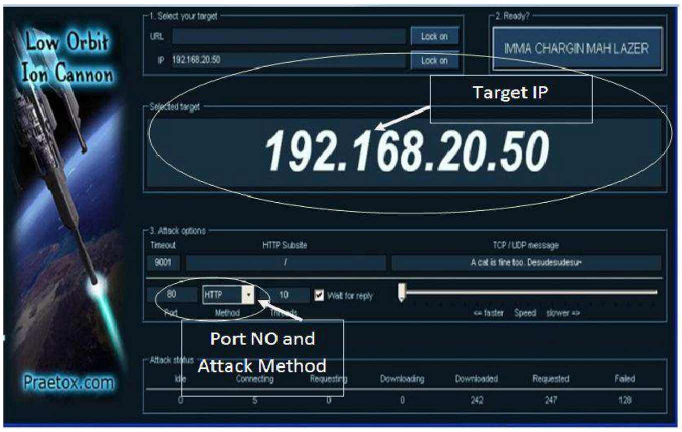
alert udp 192.168.1.0/24 any -> 192.168.20.0/24 21 (msg: "DoS attack UDP Flooding on FTP SERVER"; sid:554;

This means that the rule will alert if any flooding of udp from external network (192.168.1.0) using any udp port goes to virtualized server network (192.168.20.0) using port 21.

Then LOIC tool was installed on user's computer, the target IP was specified using udp flooding to port 21. When Snort is run and the flooding attack is detected, then Snort will send alerts as shown in Figure 7.

Table 2: ICMP alert format

Date	07-16-2014
Time	13:44:10
Snort IP	192.168.20.253
Date, Time and RN	July 16 13:44:10Labm623
Policy Message	User is pinging
Port/Protocol	ICMP
Source IP:	192.168.1.11
Destination IP:	192.168.20.40



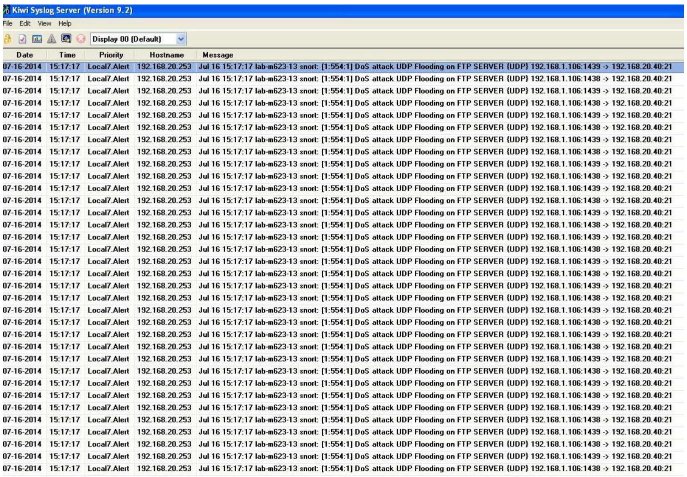
URL
IP 192.168.20.50

Timeout 9001
HTTP Subsite /
TCP / UDP message A cat is fine too. Desudesudesu~

Port 80
Method HTTP
Threads 10
☑ Wait for reply
<= faster Speed slower =>

Idle	Connecting	Requesting	Downloading	Downloaded	Requested	Failed
0	5	0	0	242	247	128

Figure 6: LOIC DDoS tool



Kiwi Syslog Server (Version 9.2)

```
Date        Time      Priority      Hostname        Message
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1439 -> 192.168.20.40:21
07-16-2014  15:17:17  Local7.Alert  192.168.20.253  Jul 16 15:17:17 lab-m623-13 snort: [1:554:1] DoS attack UDP Flooding on FTP SERVER {UDP} 192.168.1.106:1438 -> 192.168.20.40:21
```

Figure 7: Snort alert after detecting UDP flooding against virtualized server

Table 3 elaborates the parameters formatted of a Snort alert (the first line in Figure 7). Snort can detect threats in NIDS mode based on the rules that are created in the configuration file. By detecting DDoS attack we evaluate the Snort function of monitoring virtualized environment in the cloud platform and detecting threats against virtualized servers on the cloud.

6. SUMMARY

Cloud computing and virtualized servers are environments that provide all resources to users. This paper presents a experiment to test and evaluate the IDS (Snort) function in the virtualized server environment.

To test the IDS function, a virtualized environment is built in which the IDS is implemented. Then the Snort is validated by creating ICMP rule that gives alert when external users try to ping the virtualized servers. To evaluate the IDS function of Snort, a number of rules are created in the configuration file of Snort then a DDoS attack is simulated and Snort detected this attack and sent alert to kiwi syslog server. The future work may implement intrusion prevention system (IPS) to detect and prevent the threats against the virtualized environment.

Table 3: Format of Snort alert (DDoS attack)

Date	07-16-2014
Time	15:17:17
Snort IP	192.168.20.253
Date, Time and RN	July 16 15:17:17Labm623
Policy Message	DoS attack UDP flooding
Port/Protocol	ICMP
Source IP:	192.168.1.106
Destination IP:	192.168.20.40

7. REFERENCES

[1] A. A. Thu, ”Integrated Intrusion Detection and Prevention System with Honeypot on Cloud Computing Environment”, International Journal of Computer Applications, Volume 67: Issue 4, p. 11, 2013.

[2] A. Bakshi, B. Yogesh, ”Securing Cloud from DDOS Attacks Using Intrusion Detection System in Virtual Machin”, munication Software and Networks, 2010. ICCSN ’10. Second International Conference, pp.260,264, 26-28 Feb. 2010.

[3] C. Modi, D. Patel, A. Patel and R. Muttukrishnan, ”Bayesian Classifier and Snort based network intrusion detection system in cloud computing”, Computing Communication Networking Technologies (ICCCNT), Coimbatore, India, pp. 1-7, 26-28July,2012.

[4] C. Modi, D. Patel, H. Patel B. Borisaniya and A. Patel, ”A survey of intrusion detection techniques in Cloud”, Journal of Network and Computer Applications, Volume 36: Issue 1, pp. 42-57, 2012.

[5] H. Alsafi, W. Abduallah and A. Pathan, ”IDPS:An Integrated Intrusion Handling Model for Cloud Computing Environment”, 2012 Available [Online] http://arxiv.org/ftp/arxiv/papers/1203/1203.3323.pdf

[6] J. Lee, M. Park, J. Eom, T. Chung ”Multi-level Intrusion Detection System and log management in Cloud Computing” Advanced Communication Technology (ICACT), pp.552,555, 13-16 Feb, 2011.

[7] P. Shelke, S. Sontakke, A. Gawande, ”Intrusion Detection System for Cloud Computing”, International Journal of Scientific Technology Research ,Volume 1, Issue 4, pp.67-71 ,May 2012.