IJCSC 0973-7391 Volume 7 • Number 1 Sept 2015 - March 2016 pp. 250-257 available online at www.csjournalss.com

Identification of Trusted Elements with Malicious Behavior of Public Cloud against DoS Attack

Mishti Samani[1]Jitendra Bhatia[2]Miren Karamta[3] M.B.potdar[4]
[1,2]Nirma University, Ahmedabad-382481, Gujarat, India
[3,4]Bhaskaracharya Institute of Space Application and Geo-informatics, Gujarat, India
14mcei09@nirmauni.ac.in, Jitendra.bhatia@nirmauni.ac.in, karamta.miren@gmail.com, mbpotdar@yahoo.com

Abstract: Cloud has been blooming technology due to its numerous characteristics. Distributed nature, open internet, security issues related to cloud service models are some of the security threats that are associated with cloud. With this rise in threats, it is necessary to identify sources that are associated with security attacks. One of the sophisticated attack is Distributed Denial of Service attack whose identification is found to be challenging. Numerous techniques have been proposed. This paper focuses on hybrid technique which is based on rule based detection along with use of snort. The performance of server has been monitored using ganglia which is a scalable distributed monitoring tool.

Keywords: DDoS, Ganglia, Wireshark, Rule Based, TCP SYN.

1. INTRODUCTION

With Advancement in Technology, cloud has been most emerging technology. With its numerous characteristics and advantages, there has been rise in migration of cloud users which leads to raise in security concern. With this rise in security concerns it is necessary to identify the source that leads to Security threats. The Cloud resources that facilitates by providing numerous characteristics are exploited by Intruders. Cloud Security Alliance has reportedly mentioned that DDoS attack is one of top 15 cyber threats in Cloud [1] . There arise need for detection technique for DDoS Attack by identifying its sources of attack. Some of the key characteristics to identify DDoS attack is degradation of performance time with increase in response time. It mainly aims to make services unavailable to intended or legitimate users for some period of time. Varioustypes of DDoS attack have been experimented and analyzed. To detect the type of DDoS attack and its impact on the network is our primary goal in cloud. Detection of attack is very crucial process as it has no specific measures. Performance of Cloud server has been monitored using Ganglia. A distributed Monitoring System named Ganglia is used for monitoring high-performance systems such as Clusters and Grids [2]. Even hybrid technique is used to detect various attacks thus by helping to prevent this attack in future.

2. VARIOUS ATTACKS AND ITS DETECTION

Various attacks that are carried out by intruders have different motives by making resourcesunavailable to its legitimate users. Most of the target are CPU cycles, packet cache buffers,network bandwidth and so on. On victim side incoming packets are in huge number thancompared to outgoing packets. Packets are generated using packet generating tool such as LOIC,hyenae and so on in cloud. Even the botnet has been created using ufonet in cloud so that varioustraffics can be generated and analyzed. Various ICMP, UDP, TCP/IP and HTTP request aregenerated from this packet generating tool. The traffic that is generated between the nodes hasbeen recorded using analyzer named tshark and is dumped into a file. The traffic that is dumpedinto file consists of normal as well as malicious traffic which is then transferred to server withexpire in time interval. Some of the important factors that are considered for detection of DoSattack is CPU load, Memory Usage and Network Bandwidth consumption are monitored usingsome shell scripts as well as ganglia which is large scale monitoring tool. This attack can bedetected using Snort and Rule based Detection Technique.

Following all the attack has been monitored using ganglia that has been deployed in cloud.

1. ICMP Attack

Ping and its variation hping command are used to check the services of any particular system. Any packet can be maximum of 65,535 bytes. Communication between systems can be carried using ICMP Ping request and ICMP Ping reply. Attacker floods the system for sending thousands of packets to server using spoofed IP Address. So that replies are send to spoofed address. By using ping command to flood with excess number of packets the resources are consumed.

Snort rule for detecting ICMP Attack is**:**

DOI: 10.090592/IJCSC.2016.039

Alert icmp any -> 192.168.1.0/24 any (msg: "ICMP attack detected";sid:10000001;rev:001;)

Packet Size(bytes)	CPU Utilization(%)	RAM(mb)	Network Usage (Mbytes/sec)
10	60	213.5	6
100	68	213.8	10
500	79	450	18
1000	82	520	25

Table 1: ICMP Attacks Performance Matrix

2. SMURF Attack

The ICMP echo request is broadcasted with victim's IP Address. All the Intermediate machines respond with ICMP echo reply. This leads to flooding of network with thousands of reply. By spoofing the source IP Address same as destination IP Address the resources are exhausted. Snort rule for detecting Smurf Attack is:

Alert icmp any any -> 192.168.1.0/24 any (msg: "Smurf attack detected"; itype: 8; Sid: 5000002; rev: 1 ;)

Packet Size(bytes)	CPU Utilization(%)	RAM(mb)	Network Usage (Mbytes/sec)
10	15	680	1
100	36.6	710	1.8
500	48	724	2.8
1000	65	742.8	3.3

Table 2: Smurf Attacks Performance Matrix

3. HTTP DoS

HTTP Flooding is been created by use of Zombies i.e. Ufonet. Valid or Invalid Http request are sent to server by using three way handshake communication. By using zombie such as ufonet to perform HTTP DoS attack on Server by generating valid or invalid HTTP Request.

The following rule detects a pattern "GET" in the data part of all TCP packets that are leaving 192.168.1.0 network and going to an address that is not part of that network. The GET keyword is used in many HTTP related attacks; however, this rule is only using it to help you understand how the content keyword works.

Alert tcp 192.168.1.0/24 any -> ![192.168.1.0/24] any (content: "GET"; msg: "GET matched";)

The following rule does the same thing but the pattern is listed in hexadecimal.

alert tcp 192.168.1.0/24 any -> ![192.168.1.0/24] any (content: "|47 45 54|"; msg: "GET matched";)

Packet Size(bytes)	CPU Utilization(%)	RAM(mb)	Network Usage (Mbytes/sec)
10	30	600	2.1
100	45	630	2.8
500	65	685	3.5
1000	75	700	4

Table 3: HTTP DoS Performance Matrix

4. TCP SYN

The Basic step of three way handshaking is exploited. For communication purpose between servers TCP SYN and TCP ACK messages are exchanged. Attacker spoofs the IP Address so the SYN ACK packets are send to victims (spoofed) Address which completely fill ups maximum limit of SYN ACK Packets. Since packets waits for ACK until it times out and get dropped, victims machine is flooded with illegitimate request and would not be able to serve legitimate request. By exploiting the basic three way handshake the attack has been performed and has been monitored using ganglia.

Snort rule for detecting TCP SYN Attack is **:**

Alert tcp any any -> any any (msg: "TCP SYN Flood attack detected";flags:S; threshold: type threshold, track by_dst,count 20,seconds 60 ;sid:5000001;rev:1;)

Packet Size(bytes)	CPU Utilization(%)	RAM(mb)	Network Usage (Mbytes/sec)
10	55	700	5
100	70	680	5.2
500	79	747	4.9
1000	85	790.5	5.78

Table 4: TCP SYN Flood Performance Matrix

4. UDP SYN

Since UDP is connectionless protocol, the attacker generates enough UDP Packets to a random port in Victims Server. On the Victim Side, it will check for application that will be waiting for that particular packet unless it realize there is no application waiting for it. So ICMP with destination unreachable is generated to source address. If enough number of Packets are received at the victim end, the system would be flood and would be down.The ports that are open in victim side is targeted. Enough UDP Packets that can flood the victim's server are generated which would exhaust all the available resources such as CPU, bandwidth and memory.

Snort rule for detecting UDP SYN Attack is**:**

Alert udp any any -> 192.168.1.0/24 any (msg: "Land attack detected";flags:S; threshold: type threshold, track by_dst,count 20,seconds 60;sid:5000003;rev:1;)

Packet Size(bytes)	CPU Utilization(%)	RAM(mb)	Network Usage (Mbytes/sec)
10	60	420	2.5
100	65	432	4
500	75	450	6.5
1000	62	480	7.9

Table 5:UDP SYN Flood Performance Matrix

5. LAND Attack

The source IP is spoofed as of Destination IP. So the machine send huge request to itself and this conflict cannot be resolved at last victim gets crashed or rebooted. Spoofed

IP Address that is same as Victim is used by attacker so that request is send to itself and all the resources gets consumed**.**

Snort rule for detecting Land Attack is**:**

Alert tcp any any -> any any (msg: "Land attack detected"; flags: S; sameip; sid: 5000000; rev: 1 ;)

Packet Size(bytes)	CPU Utilization(%)	RAM(mb)	Network Usage (Mbytes/sec)
10	65	350	2.9
100	71	368	3.1
500	73	371	3.5
1000	75	380	3.7

Table 6: LAND Attack Performance Matrix

3. RELATED WORK

Various methods have been proposed for detection of TCP SYN Attack in which some of them are explained below:

S.H.C Haris et al. suggest that IP Header and TCP header payload are used for detection of TCP SYN Flood. Port, flag, IP address, Protocol behavior and so on are some of the key features used for attack detection. The focus of this paper is limited to detect attack in the local area network in File Transfer Protocol and has lower detection rate. The packet captured using tcpdump are filtered using packet filtering algorithm and thus would raise alarm based on deviation from normal behavior [1] .

Y.Ohsita et al.suggest to consider arrival time variations. This proposal is limited to detect normal TCP SYN Packets as lower rate traffic cannot be detected as it follows normal distribution model.By normal model distribution, the mechanism can detect attack accurately [2].

H. Wang et.al suggest that the detection system should be kept at the edge of routers or firewalls or proxy at the front end. It analyze the TCP SYN FIN pairs and the change in the sequences. Various alerts are generated based on the events and source of flooding can be identified. Thus the limitation of it is that system is more prone to flooding attacks but it does remove the overhead. Along with detecting attacks by generating alarms even the source location can be found using this technique [3].

M.Durairaj et al. proposed ThreV algorithm for detecting MAC spoof DoS attack as MAC address can easily be spoofed. The paper focuses on existing Infrastructure. Hybrid Mechanism is proposed which is amalgamation of four algorithms such as ThreV, Alternative NumberingMechanism, Traffic Pattern Filtering and Letter envelop protocol. The Basic Identity Check tables is compared with MAC address of all users in WLAN and based on that Intruders can be checked. The benefit of this technique is that it is deployed with minimum packet loss, reduced control overhead with reduced in packet drop and delay [4].

Maciej Korczynski et.al suggest that scheme that relies on sampling rate. To validate connection, TCP Packets are examined to check for ACK Segments coming from server. This method is effective when the rate of incoming packets is been controlled and then further compared with other detection methods. The ACK flag is mainly examined with set on means that connection is legitimate. Although this method is very effective by decreasing false positive rate but some information is lost while sampling data [5].

D.M.Divakaran et.al suggest to use exponential back off property of TCP segments to determine high intensity of attack. Linear Predictive analyze network traffic and various other types of DoS attack. Even the intensity of network can be detected using LP Detection Method. The low and high intensity SYN flooding attacks can be detected. There will be detection delay in source identification of TCP SYN flood [6].

S.H.C Haris et al. suggest that use of payload and unusable area in Hyper Text Transfer Protocol. ToS, IP Header, Unusable area are considered for detecting TCP SYN. To detect the TCP SYN attack it is necessary to recognize normal payload characters else would be time consuming. The need arises to make detection faster and effective [7].

Parasa Harika et al. suggest to count and record SYN packets whose three way handshake is completed. Even all packets that are opposite to SYN packets are recorded. The Proposed Technique is combination of packet filtering and syn flood monitoring [8].

D.D.Rani et al. suggest to check open ports and its active connections in Server. Using Wireshark and IP table rule DoS attack is analyzed. Once DoS attack is detected its prevention can be done using shell scripts to block such network traffic. The experiment for detection is limited to client server program[9].

D.S.Rana et al used Wireshark to detect TCP SYN flood attack. The attack has been generated by Shell Script using random number function so that the request comes from Random IP address. Use of Inbuilt functions in Linux such as netstat is done. Around 2000 to 7000 packets are captured at network interface[10].

V.A.Siris et.al evaluated adaptive threshold algorithm and cumulative sum algorithm for change point detection. Adaptive threshold algorithm checks for network traffic and compares SYN packets with the threshold value. When the number of SYN Packets exceed number of FIN Packets the change has been noted using cusum algorithm. For low intensity attack there is degradation in performance. It is efficient for detecting high intensity attacks without being more complex[11].

V.L.L.Thing et.al proposed use for bloomed filter. The outgoing SYN packets values must be equal to incoming SYN & ACK values. The technique is more reliable in detecting SYN-SYN & ACK detection mechanism rather than SYN-FIN/RST detection mechanism. SYN-FIN/RST fails to detect Bot Buddy attack [12].

4. TCP SYN FLOOD ATTACK

TCP is Connection oriented protocol. TCP is The Server sends request to another server in form of SYN packets. The Server Responds with SYN ACK packet by reserving connection resources. Client sends an ACK to the server and thus connection is established. Attacker generates numerous SYN requests but never responds to ACK to complete connection. The new incoming SYN request are dropped as victim's server backlog queue is exhausted. The problem in detecting TCP SYN flood attack is that server cannot distinguish normal TCP traffic and SYN Flood Packets.

The Reason behind TCP SYN Flood Attack are:

DOI: 10.090592/IJCSC.2016.039

1. TCP Threeway handshake protocol which allows attacker to access resources as well as server.
2. Server has no control on the incoming SYN packets and exhaust the system resources.

The Attack can be detected by half open connection using netstat. This half open connection are described as SYN_RECV or SYN_RECEIVED.
#netstat –n –p TCP
For further description of TCP statistics, following command can be used.
#netstat –n –p TCP | grep SYN_RECV |grep : 23| wc

SYN Flood does not affect ingoing connection but denies new connection. There are several ways to detect TCP SYN Flood attack as follows:

1. **Anomaly Detection System:** In this technique, IP Header and TCP Header are checked. Normal Packets and infected Packets are categorized based on TCP Header. Authorized IP Address are maintained in database and is used for comparison for further analysis. Once Infected IP's are detected alerts are generated and ip address is further added into blacklist.
2. **Efficient Packet Marking:** In this technique, routers write IP address in header field so that network information path can be obtained. Since the space is limited in marking field, routers probabilistic decides to mark the routing information which contains partial routing information. The approach in which All received packets and marking information will lead to network path is called Probabilistic Packet marking.

5. EXPERIMENTAL SETUP

The Experiment has used six host machines deployed in cloud. Host A, B, C, D are attackers with OS Ubuntu 12.04.5 LTS. Host E is IDS which consists of Snort and Rule base detection techniques with OS Kali 2.0 Sana. Host F is client with OS Ubuntu 12.04.5 LTS deployed with ganglia to monitor the performance. Some of the tools used are Snort 2.9.7.6, DAQ 2.0.6, barnyard2 2-1.13, base 1.4.5, LOIC 1.0.8, hyenae 0-1.1, ufonet 0.6, Airmon-ng, Airodump-ng, Driftnet.

The attack is performed on Host F through use of different attack scripts, tools such LOIC, ufonet, hyeane. All the incoming request of Host F is been analyzed and monitored by Host E. Different Rules are configured based on Various Attacks. Both Host E and F are configured with ganglia so that performance can be analyzed.

6. RESULTS AND DISCUSSION

It is found that TCP SYN attack has maximum CPU Utilization as described in (1)**.** Even the RAM usage and Network usage has been founded maximum among other flooding attacks as described in (2). Comparative Study of CPU Utilization , RAM and Network Usage has been depicted in (3).It Can be concluded from the above analysis that TCP SYN Flood is more severe than any other flooding attacks followed by ICMP Attack. TCP SYN Attack is detected using base, snort, Wireshark and rule based detection (4).

MITIGATION TECHNIQUES FOR TCP-SYN

1. TCP Probing

An additional craft message is appended with SYN+ACK+Craft message which is replied by the Server. So, Craft message is to change in TCP Window size. This reply is checks the specification given by server to change the window size.

2. SYN Cookies

Server receives SYN Packet and calculates SYN Cookies. This Cookie is then send back to client in form of SYN+ACK and there is no allocation of resources for request send by the client. Once ACK Packets are received by server check for the valid cookies and based on that resource is been allocated so that resources cannot be exhausted.

3. Rise in Backlog queue of Server

Increase in Tcp_max_syn_backlog parameter value so that half open connections can be easily maintained.

5. CONCLUSION

In this paper, we have focused on identifying sources that facilitated with numerous characteristics but were exploited by Intruder. Since DDoS is major threat to cloud its detection is very challenging. An Hybrid Technique which is combination of Rule based Detection and Snort has been used for identifying the attacks. After Performance Comparison of Various types of DDoS attacks, it is concluded that TCP SYN Attack is more severe compared to other attacks (3). A hybrid technique is used for detection of attacks. The Rule based detection techniques works efficiently in cloud. With the several comparisons it is found that TCP SYN is more severe compared to various other DDoS attacks.

6. REFERENCES

[1]Rajendran, Praveen Kumar, B. Muthukumar, and G. Nagarajan. "Hybrid Intrusion Detection System for Private Cloud: A Systematic Approach." *Procedia Computer Science* 48 (2015): 325-329.

[2] Choo, Kim-Kwang Raymond. "Cloud computing: challenges and future directions." (2010):

[3] Modi, Chirag, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. "A survey of intrusion detection techniques in cloud." *Journal of Network and Computer Applications* 36, no. 1 (2013): 42-57.

[4] Patel, Ahmed, Mona Taghavi, Kaveh Bakhtiyari, and Joaquim Celestino Júnior. "An intrusion detection and prevention system in cloud computing: A systematic review." *Journal of Network and Computer Applications* 36, no. 1 (2013): 25-41.

[5] Deka, Rup Kumar, Kausthav Pratim Kalita, D. K. Bhattacharya, and Jugal K. Kalita. "Network defense: Approaches, methods and techniques." *Journal of Network and Computer Applications* 57 (2015): 71-84.

[6] Ali, Mazhar, Samee U. Khan, and Athanasios V. Vasilakos. "Security in cloud computing: Opportunities and challenges." *Information Sciences* 305 (2015): 357-383.

[7] Deshmukh, Rashmi V., and Kailas K. Devadkar. "Understanding DDoS Attack & its Effect in Cloud

Environment." *Procedia Computer Science* 49 (2015): 202-210.

[8] Jabez, J., and B. Muthukumar. "Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach." *Procedia Computer Science* 48 (2015):338-346.[9] Hosseini, BS Mojtaba, Behnam Amiri, Mahboubeh Mirzabagheri, and Yong Shi. "A New Intrusion Detection Approach using PSO based Multiple Criteria Linear Programming." *Procedia Computer Science* 55 (2015): 231-237.

[10] Che, Jianhua, Yamin Duan, Tao Zhang, and Jie Fan. "Study on the security models and strategies of cloud computing." *Procedia Engineering* 23 (2011): 586-593.

[11] Fatema, Kaniz, Vincent C. Emeakaroha, Philip D. Healy, John P. Morrison, and Theo Lynn. "A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives." *Journal of Parallel and Distributed Computing* 74, no. 10 (2014): 2918-2933.

[12] Liao, Hung-Jen, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. "Intrusion detection system: A comprehensive review." *Journal of Network and Computer Applications* 36, no. 1 (2013): 16-24.

[13] Di Pietro, Roberto, and Luigi V. Mancini. *Intrusion detection systems*. Vol. 38. Springer Science & Business Media, 2008.

[14] Zissis, Dimitrios, and Dimitrios Lekkas. "Addressing cloud computing security issues." *Future Generation computer systems* 28, no. 3 (2012): 583-592.

[15] Narwane, S. V., and S. L. Vaikol. "Intrusion Detection System in Cloud Computing Environment." In *InInternational Conference on Advances in Communication and Computing Technologies (ICACACT)*. 2012.

[16] Mohod, Akash G., and Satish J. Alaspurkar. "Analysis of IDS for Cloud Computing." *International Journal of Application or Innovation in Engineering & Management (IJAIEM) Vol* 2: 344-349.

[17] Subashini, Subashini, and V. Kavitha. "A survey on security issues in service delivery models of cloud computing." *Journal of network and computer applications* 34, no. 1 (2011): 1-11.

[18] Mazzariello, Claudio, Roberto Bifulco, and Roberto Canonico. "Integrating a network IDS into an open source cloud computing environment." In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pp. 265-270. IEEE, 2010.

[19] Kene, Snehal G., and Deepti P. Theng. "A review on intrusion detection techniques for cloud computing and security challenges." In *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, pp. 227-232. IEEE, 2015.

[20] Girma, Anteneh, Moses Garuba, Jiang Li, and Chunmei Liu. "Analysis of DDoS Attacks and an Introduction of a Hybrid Statistical Model to Detect DDoS Attacks on Cloud Computing Environment." In *Information Technology-New Generations (ITNG), 2015 12th International Conference on*, pp. 212-217. IEEE, 2015.

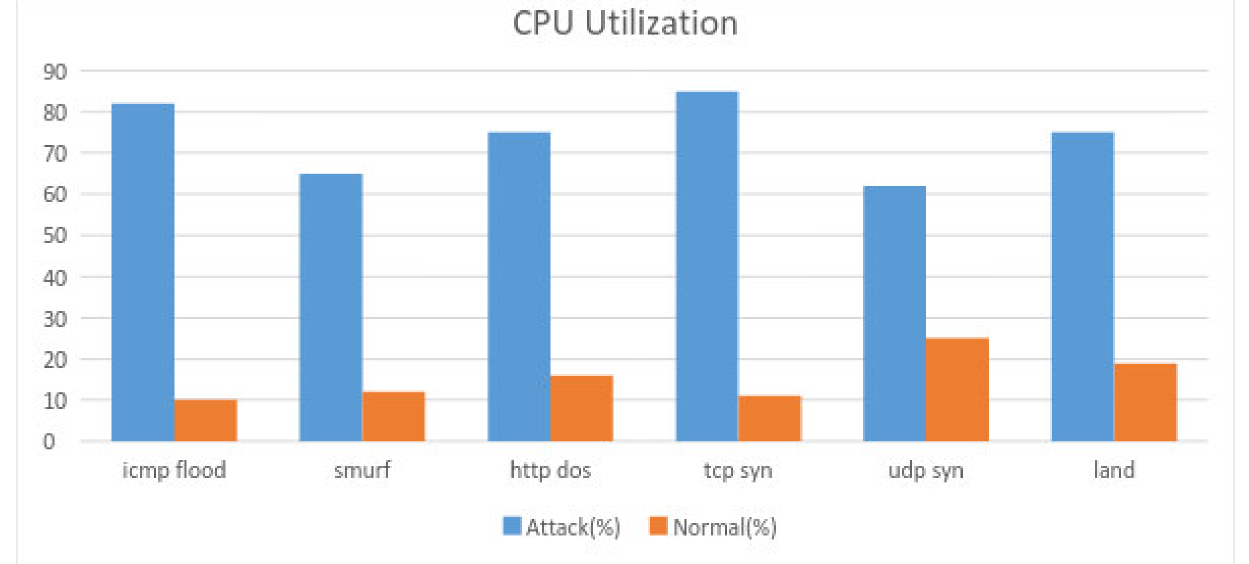
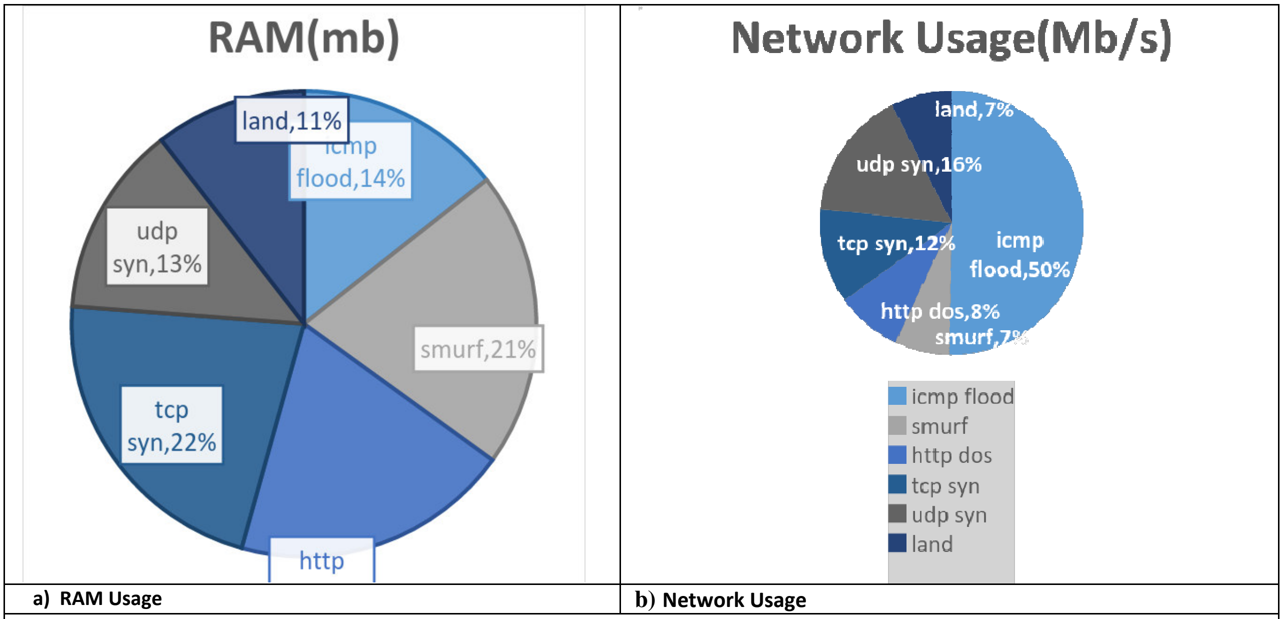


Figure (1). Comparison of CPU Utilization



a) RAM Usage

b) Network Usage

Figure (2). RAM(a) and Network Usage(b) During Attack

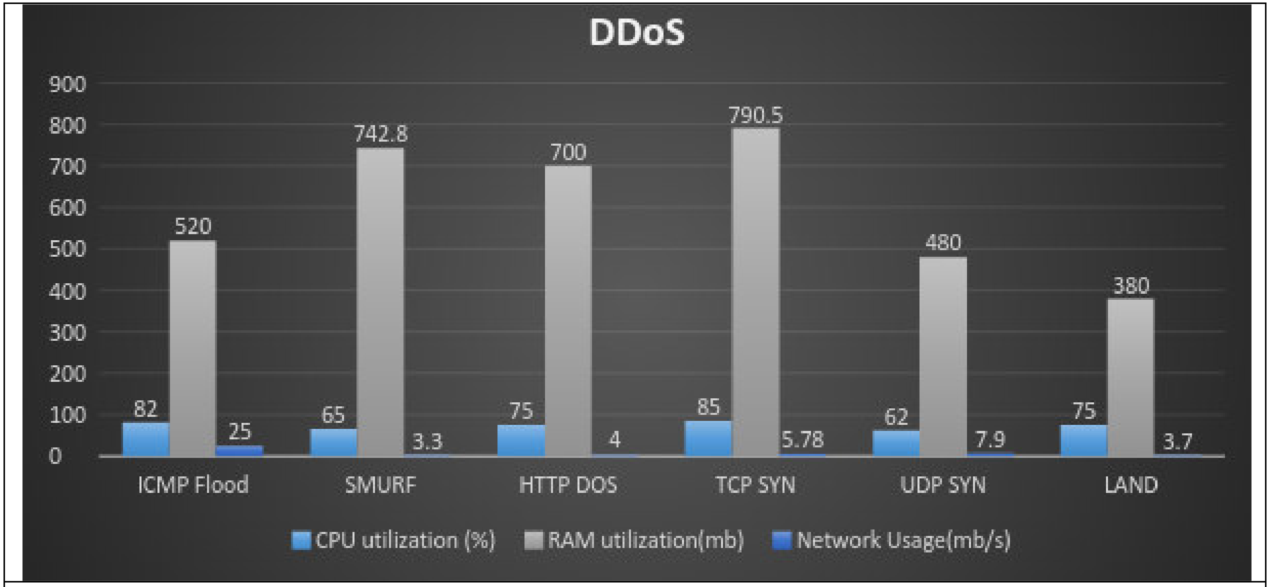
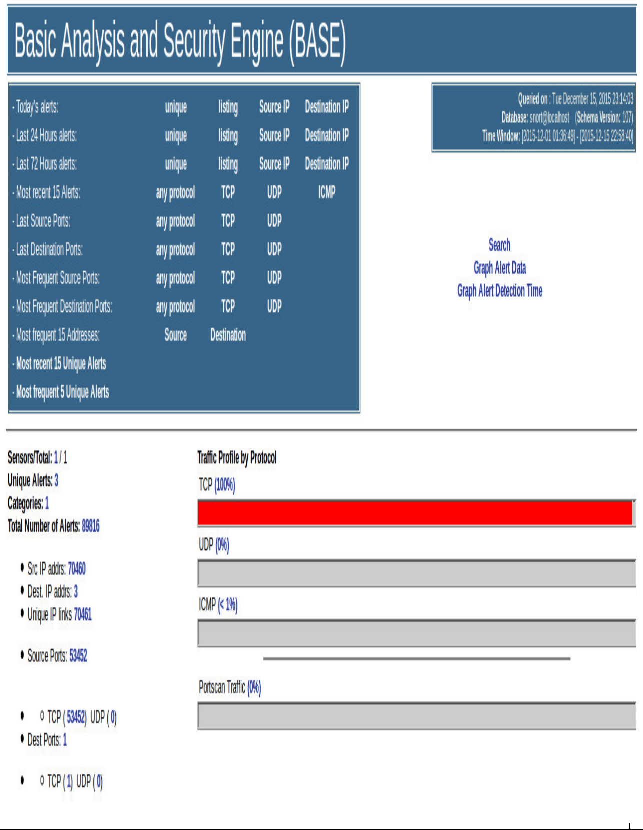


Figure 3. Comparative Study of DDoS based on CPU, RAM, Network Usage.



Basic Analysis and Security Engine (BASE)

- Today's alerts:	unique	listing	Source IP	Destination IP
- Last 24 Hours alerts:	unique	listing	Source IP	Destination IP
- Last 72 Hours alerts:	unique	listing	Source IP	Destination IP
- Most recent 15 Alerts:	any protocol	TCP	UDP	ICMP
- Last Source Ports:	any protocol	TCP	UDP	
- Last Destination Ports:	any protocol	TCP	UDP	
- Most Frequent Source Ports:	any protocol	TCP	UDP	
- Most Frequent Destination Ports:	any protocol	TCP	UDP	
- Most frequent 15 Addresses:	Source	Destination		
- Most recent 15 Unique Alerts				
- Most frequent 5 Unique Alerts				

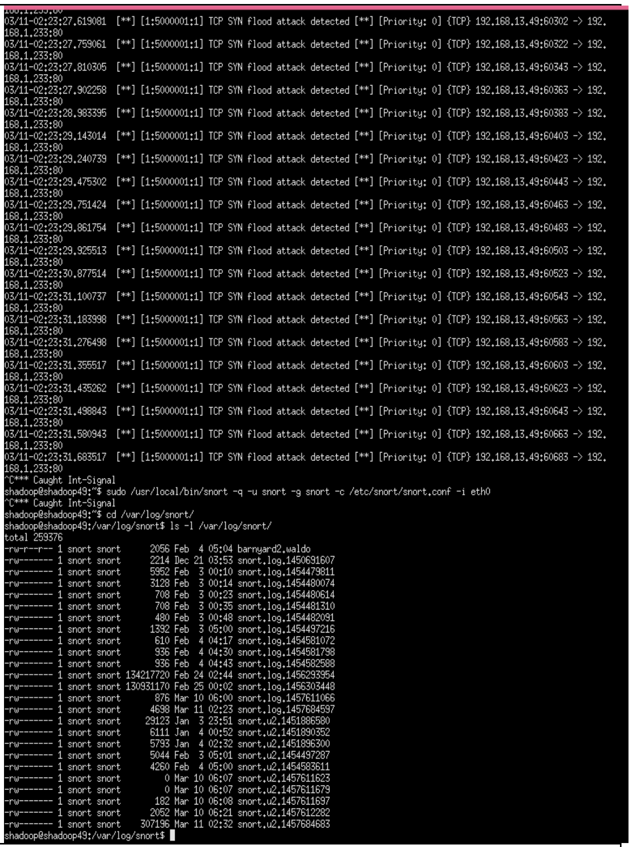
Queried on : Tue December 15, 2015 23:14:03
Database: snort@localhost (Schema Version: 107)
Time Window: [2015-12-01 01:36:49] - [2015-12-15 22:58:40]

Sensors/Total: 1 / 1
Unique Alerts: 3
Categories: 1
Total Number of Alerts: 89816

- Src IP addrs: 70460
- Dest. IP addrs: 3
- Unique IP links 70461
- Source Ports: 53452
- ○ TCP (53452) UDP (0)
- Dest Ports: 1
- ○ TCP (1) UDP (0)

Traffic Profile by Protocol
TCP (100%)
UDP (0%)
ICMP (< 1%)
Portscan Traffic (0%)

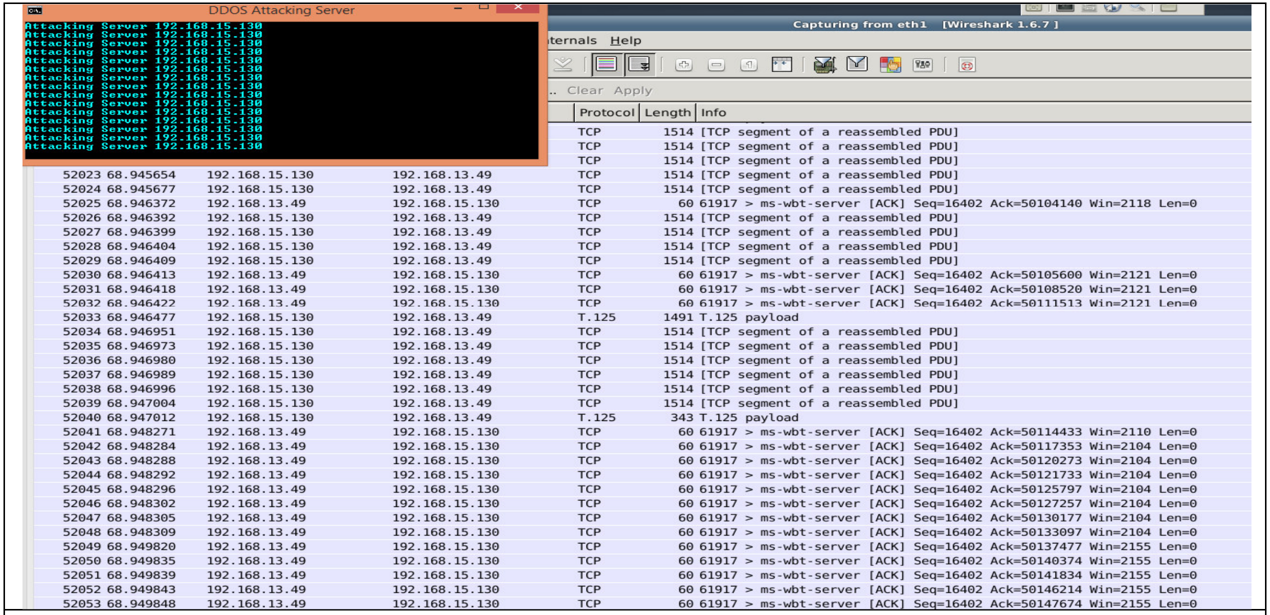
Figure 4(a). tcp detection using base



```
168.1.233:80
03/11-02:23:27.619081  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60302 -> 192.
168.1.233:80
03/11-02:23:27.759061  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60322 -> 192.
168.1.233:80
03/11-02:23:27.810305  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60343 -> 192.
168.1.233:80
03/11-02:23:27.902258  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60363 -> 192.
168.1.233:80
03/11-02:23:28.983395  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60383 -> 192.
168.1.233:80
03/11-02:23:29.143014  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60403 -> 192.
168.1.233:80
03/11-02:23:29.240739  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60423 -> 192.
168.1.233:80
03/11-02:23:29.475302  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60443 -> 192.
168.1.233:80
03/11-02:23:29.751424  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60463 -> 192.
168.1.233:80
03/11-02:23:29.861754  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60483 -> 192.
168.1.233:80
03/11-02:23:29.925513  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60503 -> 192.
168.1.233:80
03/11-02:23:30.877514  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60523 -> 192.
168.1.233:80
03/11-02:23:31.100737  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60543 -> 192.
168.1.233:80
03/11-02:23:31.183998  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60563 -> 192.
168.1.233:80
03/11-02:23:31.276498  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60583 -> 192.
168.1.233:80
03/11-02:23:31.355517  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60603 -> 192.
168.1.233:80
03/11-02:23:31.435262  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60623 -> 192.
168.1.233:80
03/11-02:23:31.498843  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60643 -> 192.
168.1.233:80
03/11-02:23:31.580943  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60663 -> 192.
168.1.233:80
03/11-02:23:31.683517  [**] [1:5000001:1] TCP SYN flood attack detected [**] [Priority: 0] {TCP} 192.168.13.49:60683 -> 192.
168.1.233:80
^C*** Caught Int-Signal
shadoop@shadoop49:~$ sudo /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
^C*** Caught Int-Signal
shadoop@shadoop49:~$ cd /var/log/snort/
shadoop@shadoop49:/var/log/snort$ ls -l /var/log/snort/
total 259376
-rw-r--r-- 1 snort snort      2056 Feb  4 05:04 barnyard2.waldo
-rw------- 1 snort snort      2214 Dec 21 03:53 snort.log.1450691607
-rw------- 1 snort snort      5952 Feb  3 00:10 snort.log.1454479811
-rw------- 1 snort snort      3128 Feb  3 00:14 snort.log.1454480074
-rw------- 1 snort snort       708 Feb  3 00:23 snort.log.1454480614
-rw------- 1 snort snort       708 Feb  3 00:35 snort.log.1454481310
-rw------- 1 snort snort       480 Feb  3 00:48 snort.log.1454482091
-rw------- 1 snort snort      1392 Feb  3 05:00 snort.log.1454497216
-rw------- 1 snort snort       610 Feb  4 04:17 snort.log.1454581072
-rw------- 1 snort snort       936 Feb  4 04:30 snort.log.1454581798
-rw------- 1 snort snort       936 Feb  4 04:43 snort.log.1454582588
-rw------- 1 snort snort 134217720 Feb 24 02:44 snort.log.1456293954
-rw------- 1 snort snort 130931170 Feb 25 00:02 snort.log.1456303448
-rw------- 1 snort snort       876 Mar 10 06:00 snort.log.1457611066
-rw------- 1 snort snort      4698 Mar 11 02:23 snort.log.1457684597
-rw------- 1 snort snort     29123 Jan  3 23:51 snort.u2.1451886580
-rw------- 1 snort snort      6111 Jan  4 00:52 snort.u2.1451890352
-rw------- 1 snort snort      5793 Jan  4 02:32 snort.u2.1451896300
-rw------- 1 snort snort      5044 Feb  3 05:01 snort.u2.1454497287
-rw------- 1 snort snort      4260 Feb  4 05:00 snort.u2.1454583611
-rw------- 1 snort snort         0 Mar 10 06:07 snort.u2.1457611623
-rw------- 1 snort snort         0 Mar 10 06:07 snort.u2.1457611679
-rw------- 1 snort snort       182 Mar 10 06:08 snort.u2.1457611697
-rw------- 1 snort snort      2052 Mar 10 06:21 snort.u2.1457612282
-rw------- 1 snort snort    307196 Mar 11 02:32 snort.u2.1457684683
shadoop@shadoop49:/var/log/snort$ █
```

Figure 4(b). tcp detection using Snort

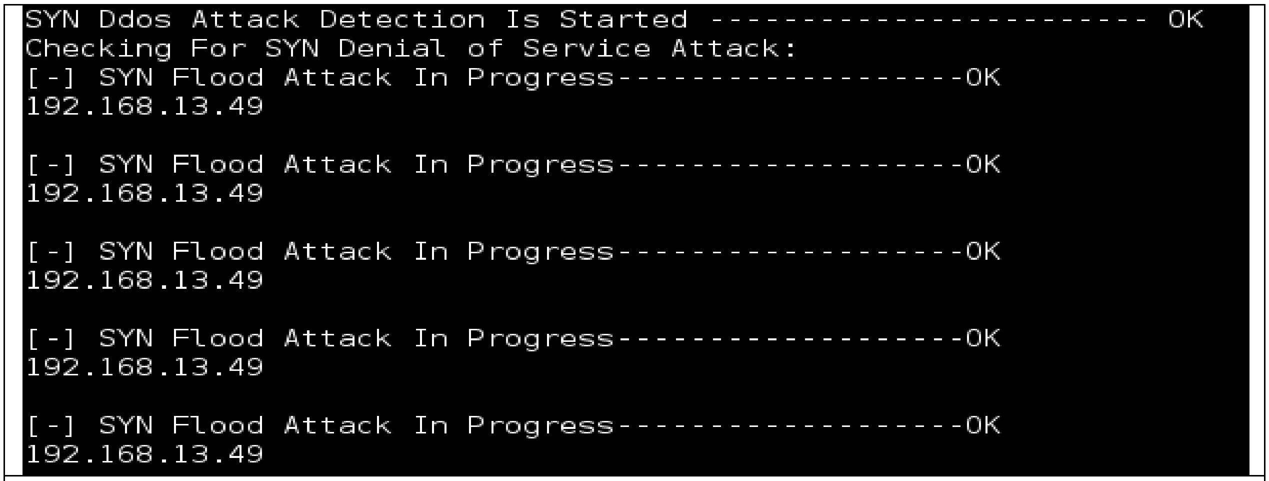
DOI: 10.090592/IJCSC.2016.039



```
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
Attacking Server 192.168.15.130
```

```
                                                            Protocol Length Info
                                                            TCP        1514 [TCP segment of a reassembled PDU]
                                                            TCP        1514 [TCP segment of a reassembled PDU]
                                                            TCP        1514 [TCP segment of a reassembled PDU]
52023 68.945654   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52024 68.945677   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52025 68.946372   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50104140 Win=2118 Len=0
52026 68.946392   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52027 68.946399   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52028 68.946404   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52029 68.946409   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52030 68.946413   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50105600 Win=2121 Len=0
52031 68.946418   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50108520 Win=2121 Len=0
52032 68.946422   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50111513 Win=2121 Len=0
52033 68.946477   192.168.15.130     192.168.13.49          T.125      1491 T.125 payload
52034 68.946951   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52035 68.946973   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52036 68.946980   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52037 68.946989   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52038 68.946996   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52039 68.947004   192.168.15.130     192.168.13.49          TCP        1514 [TCP segment of a reassembled PDU]
52040 68.947012   192.168.15.130     192.168.13.49          T.125       343 T.125 payload
52041 68.948271   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50114433 Win=2110 Len=0
52042 68.948284   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50117353 Win=2104 Len=0
52043 68.948288   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50120273 Win=2104 Len=0
52044 68.948292   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50121733 Win=2104 Len=0
52045 68.948296   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50125797 Win=2104 Len=0
52046 68.948302   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50127257 Win=2104 Len=0
52047 68.948305   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50130177 Win=2104 Len=0
52048 68.948309   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50133097 Win=2104 Len=0
52049 68.949820   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50137477 Win=2155 Len=0
52050 68.949835   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50140374 Win=2155 Len=0
52051 68.949839   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50141834 Win=2155 Len=0
52052 68.949843   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50146214 Win=2155 Len=0
52053 68.949848   192.168.13.49      192.168.15.130         TCP          60 61917 > ms-wbt-server [ACK] Seq=16402 Ack=50147674 Win=2155 Len=0
```

Figure 4(c). tcp detection using Wireshark



```
SYN Ddos Attack Detection Is Started ------------------------ OK
Checking For SYN Denial of Service Attack:
[-] SYN Flood Attack In Progress-------------------OK
192.168.13.49

[-] SYN Flood Attack In Progress-------------------OK
192.168.13.49

[-] SYN Flood Attack In Progress-------------------OK
192.168.13.49

[-] SYN Flood Attack In Progress-------------------OK
192.168.13.49

[-] SYN Flood Attack In Progress-------------------OK
192.168.13.49
```

Figure 4(d). tcp detection using rule based detection

DOI: 10.090592/IJCSC.2016.039