ISSN 0146-4116, Automatic Control and Computer Sciences, 2017, Vol. 51, No. 1, pp. 13–23. © Allerton Press, Inc., 2017.

An Approach for Detecting and Preventing DDoS Attacks in Campus[1]

Mehdi Merouane

Department of Electronics, University of BLIDA BP 270, Route Soumaa, Algeria

e-mail: mmehdi_m@hotmail.com

Received February 5, 2016; in final form, August 22, 2016

Abstract—Nowadays, Denial of Service (DoS) attacks have become a major security threat to networks and the Internet. Therefore, even a naive hacker can launch a large-scale DoS attack to the victim from providing Internet services. This article deals with the evaluation of the Snort IDS in terms of packet processing performance and detection. This work describes the aspect involved in building campus network security system and then evaluates the campus network security risks and threats, mainly analyses the attacks DoS and DDoS, and puts forward new approach for Snort campus network security solutions. The objective is to analyze the functional advantages of the solution, deployment and configuration of the open source based on Snort intrusion detection system. The evaluation metrics are defined using Snort namely comparison between basic rules with new ones, available bandwidth, CPU loading and memory usage.

Keywords: DoS/DDoS attacks, intrusion detection system, anomaly detection, snort

DOI: 10.3103/S0146411616060043

1. INTRODUCTION

The denial of services attacks are generally described as actions where legitimate users or Institutions are dispossessed of certain services (network connectivity, web or e-mail). The distributed denial of services attacks are mainly used for flooding a particular server with huge useless information. The botnets are the major vectors to be taken into consideration to divert the DDoS attacks.

There are more than 50% colleges attacked by cyber-attacks in one year, such as viruses/worms/trojans/malware, unauthorized access or DDoS. Those attacks may cause crash of campus network (system or device), network access interruption, service system and terminal system destruction, and illegal access, extremely affecting normal operation of campus network.

The main goal is to perpetrate damage on the victim. Frequently the ulterior motives are personal reasons (a significant number of DDoS attacks are effected against home computers, presumably for purposes of revenge), or prestige (successful attacks on popular Web servers gain the respect of the hacker community). However, some DDoS attacks are performed for material gain (damaging a competitor's resources or blackmailing companies) or for political reasons (a country at war could perpetrate attacks against its enemy's critical resources, potentially enlisting a significant portion of the entire country's computing power for this action). In some cases, the true victim of the attack might not be the actual target of the attack packets, but others who rely on the target's correct operation [1].

The idea behind a DOS or DDoS attack is simple – it's to take down the server. Normally attacks such as these are done to major sites as we pointed out earlier in this article, because they are high profile and affect a large amount of customers. There are different ways this can be done but they all do the same thing. DDoS threat attacks the following services [2]:

- Network Bandwidth.
- Server memory.
- CPU usage.
- Database space.
- Hard disk Space.

[1] The article is published in the original.

Our approach is to develop an improved algorithm by considering previously defined methodologies of snort IDS tool by adding a new approach in snort detection engine to identify the DoS and DDOS attacks. This engine filters all the files and loads the attacked or infected files into its loader by ".conf" file command. With the help of this, an efficient detection can be done. However, security, accuracy and reliability will be the main concern during the detection process. The main objective of the study is to analyze the problems, prospective and opportunities of various aspects in IDS Snort. In this broader domain, the following will be specific objectives of the study:

(1) To study DoS and DDoS attacks and preventive mechanisms.

(2) To analyze the campus network security risks and threats.

(3) To identify the capabilities of SNORT IDS.

(4) Implementation new approaches in Snort open source IDS.

(5) To build a prototype model or a change in architectural design to filter and delete the DoS and DDoS attack automatically in real time network.

(6) To survey the performance of snort as it becomes down during heavy network traffic.

All tests are accomplished in the university campus network environment, hackers outside or malicious attack inside the campus are potential danger sources. The network security is located in the priority of the campus network planning. Our Campus University network has a 150 Megabits per second and 100 Mbytes/s backup Internet link provided by the Academic Research Network (ARN) to the BLIDA campus network, along with 1 GB/s connectivity with two thousands users with windows environment installed.

2. DDoS OVERVIEW

DoS attacks attempt to exhaust the victim's resources. These resources can be network bandwidth, computing power, or operating system data structures. To launch a DDoS attack, malicious users first build a network of computers that they will use to produce the volume of traffic needed to deny services to computer users. To create this attack network, attackers discover vulnerable sites or hosts on the network. Vulnerable hosts are usually those that are running either no antivirus software or out-of-date antivirus software, or those that have not been properly patched. Attackers who use their vulnerability to gain access to these hosts then exploit vulnerable hosts. The next step for the intruder is to install new programs (known as attack tools) on the compromised hosts of the attack network. The hosts that are running these attack tools are known as zombies, and they can carry out any attack under the control of the attacker. Many zombies together form what we call an army [3] (Fig. 1).

2.1. DDoS Attack Categories

There are three primary categories of DDoS attacks:

Volume based attacks: Include UDP, ICMP, and other spoofed-packet floods. The attack aims to saturate the bandwidth of the targeted resource. Magnitude is measured in bits per second.

Protocol attacks: Include SYN floods, fragmented packet attacks, Smurf DDoS and more. This type of attack consumes actual server resources, or resources on the intermediate equipment, such as firewalls and load balancers. Magnitude is measured in packets per second.

Application layer attacks: Include slow POST, HashDos, GET flood, clogging and more. This attack sends data according to specific features of well-known applications such as HTTP, DNS, SMTP, and SSL. Comprised of seemingly legitimate packets, the goal of these attacks is the depletion of certain resources in the application. Magnitude is measured in requests per second. The picture below (Fig. 2) shows the layering of DDoS attacks on the OSI network model [3, 4].

2.2. DDoS Detection Methods

Based on the locality of deployment, DDoS defense schemes can be divided into three classes: victim-end, source-end, and intermediate router defense mechanisms. All of these approaches have their own advantages and disadvantages. We discuss them one by one [5].

2.2.1. Victim-end defense mechanism. Victim-end detection approaches are generally employed in the routers of victim networks, i.e., networks providing critical Web services. The detection engine is used to detect intrusion either online or offline, using both misuse based intrusion detection or anomaly based intrusion detection. The reference data stores information about known intrusion signatures or profiles of

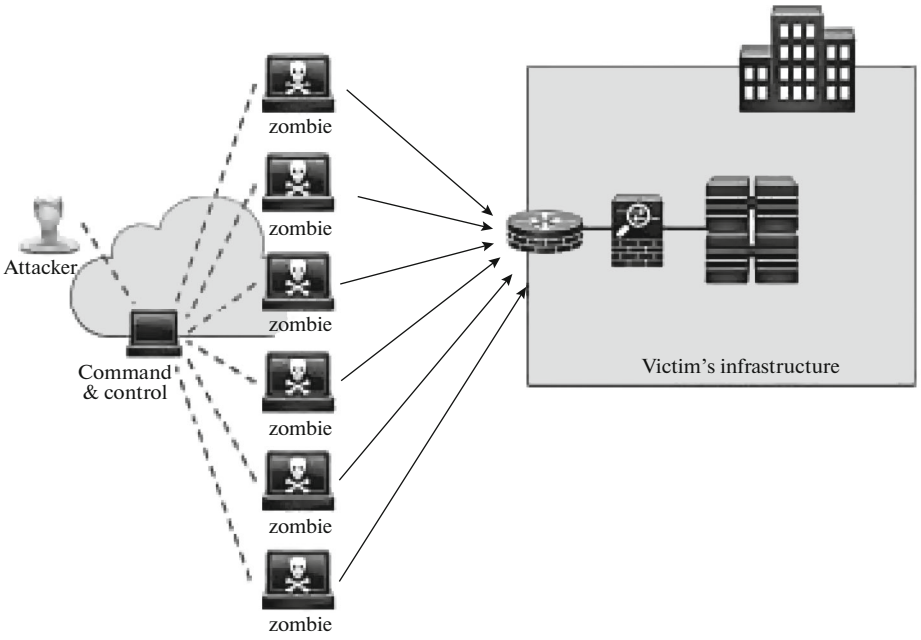

Fig. 1. A conceptual diagram of DDoS Attack.

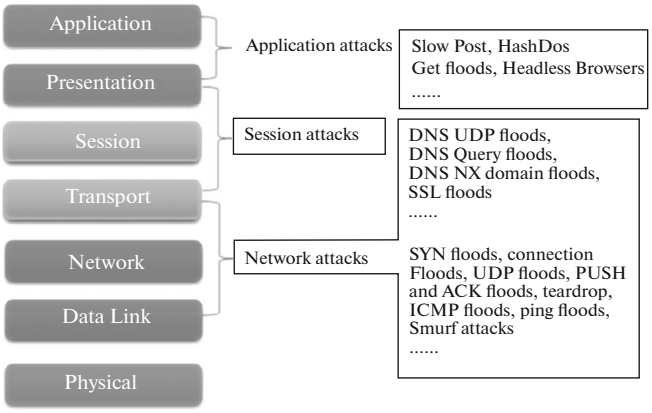

Fig. 2. The layering of DDoS attacks on the OSI network model.

normal behavior. The processing element frequently stores intermediate results in the configuration data. Detecting DDoS attacks in victim routers is relatively easy because of the high rate of resource consumption. It is also the most practically applicable type of defense scheme as Web servers providing critical services always try to secure their resources for legitimate users. But the problem with these approaches is that during DDoS attacks, victim resources, e.g., network bandwidth, often gets overwhelmed and these approaches cannot stop the flow beyond victim routers. Another important disadvantage is that these approaches detect the attack only after it reaches the victim and detecting an attack when legitimate clients have already been denied is not useful [6].

2.2.2. Source-end defense mechanism. This architecture is similar to the victim-end detection architecture. Here a throttling component is added to impose rate limit on outgoing connections. The observation engine compares both incoming and outgoing traffic statistics with some predefined normal profiles. Detecting and stopping a DDoS attack at the source is the best possible defense. It prevents the possibility of flooding not only on the victim side, but also in the whole intermediate network. The main difficulty with this approach is that, detecting DDoS attacks at source end is not easy. This is because in these attacks, sources are widely distributed and a single source behaves almost similarly as in normal trace. Another problem is the difficulty of deploying system at the source end.

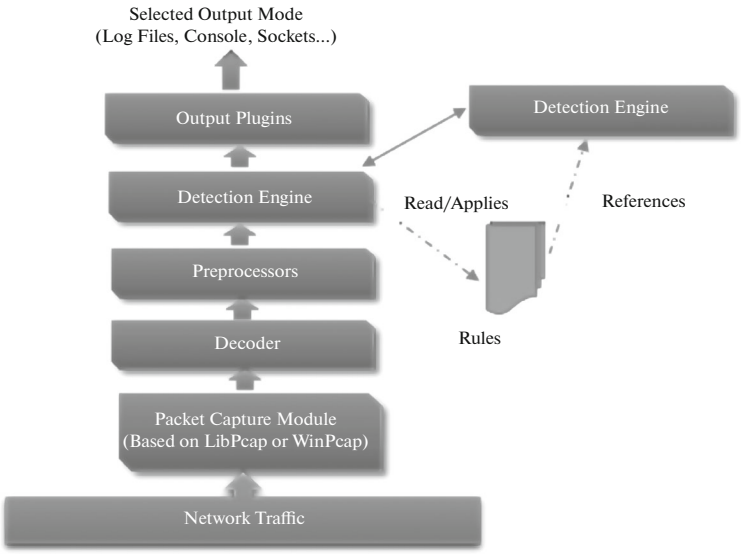

Fig. 3. Architecture of Snort.

2.2.3. Intermediate network defense mechanism. The intermediate network defense scheme balances the trade-offs between detection accuracy and attack bandwidth consumption, the main issues in source-end and victim-end detection approaches. Figure 8 shows a generic architecture of the intermediate network defense scheme, one that can be employed in any network router. Such a scheme is generally collaborative in nature and the routers share their observations with other routers. Like a source-end scheme, these schemes also impose rate limits on connections passing by the router after comparing with stored normal profiles. Detection and trace back of attack sources are easy in this approach due to collaborative operation. Routers can form an overlay mesh to share their observations [6]. The main difficulty with this approach is deployability. To achieve full detection accuracy, all routers on the Internet will have to employ this detection scheme, because unavailability of this scheme in only a few routers may cause failure to the detection and trace back process.

3. INTRUSION DETECTION AND PREVENTION

Intrusion detection and prevention systems (IDPSs) are composed of software that helps network administrator to monitor and analyze events occurring in their information systems and networks, and to identify and stop potentially harmful incidents. With the growing dependence of organizations on information systems to carry out essential activities and with the increasingly frequent and intense attacks on systems, IDPSs have become an essential component of the security infrastructure of nearly every organization [6].

3.1. Functions of Intrusion Detection and Prevention Systems

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. Incidents have many causes, such as malware (e.g., worms, spyware), attackers gaining unauthorized access to systems from the Internet, and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized. Although many incidents are malicious in nature, many others are not; for example, a user could enter an incorrect address of a system and accidentally attempt to connect to a different system without authorization [7].

3.2. Preventive Mechanisms

The goal of preventive mechanisms is either to eliminate the possibility of DDoS attacks altogether or to enable potential victims to endure the attack without denying services to legitimate clients. According

to these goals, we further divide preventive mechanisms into attack prevention and denial-of-service prevention mechanisms [7].

4. CONSTRUCTION OF SNORT NETWORK SECURITY IDS

Snort is a free Open Source, NIDS. Originally released in 1998 by Martin Roesch as a lightweight cross-platform network sniffing tool (around 1200 lines of code), it has evolved into a powerful and full-featured intrusion detection and prevention product [8]. Snort is a successful example of the Open Source development methodology, in which community members contribute to source code, bug reports, bug fixes, documentation and related tools. In our study, we downloaded the source code and install the executable files under Windows environments with the rules of file describing the intrusion detection features. As it can be seen in Fig. 3, the basic elements of its architecture are [9]:

—The capture module traffic that allows for the capture all the network packages. The decoder, which is responsible of creating data structures with the packages and identify the network protocols.

—The pre-processors that allows for extend the system functionalities; the detection engine that analyzes the packages according to the signatures; the file of signatures where the known attacks are defined for their detection.

—The detection plugins that allow to modify the functionality of the detection engine; and finally, the output plugins for determining what, how and where the alerts are saved (e.g. text file, database).

5. AIMS AND OBJECTIVES

In this particular section, we present the main contributions of the paper. The goal of the proposed system is to detect DoS/DDoS attacks on the campus server as well as store information concerning the IP addresses, ports and the number of times the attack has occurred as well as permit the traffic depending on that information. After that, it allows traffic inwards or outwards based on the rules.

For a period of two semesters, the network security students are used different DDoS attack tools in the laboratory environment after we provides the students with required network architecture setting and network devices (switch and routers). Different test required for generated DDOS attack, in order to be able to implement the appropriate security solution.

The attacker used any port scanner tool to identify the list of open TCP ports at the victim hosts (web server, mail server, ftp....). Then, the attacker can select one open port number for TCP/UDP/HTTP flood attack packets. It is used to perform DDOS attacks by simulating several zombie hosts.

This tool runs on Windows systems:

- Simulates several zombies in attack.
- Random IP addresses.
- TCP-connection-based attacks.
- Application-layer DDOS attacks.
- HTTP DDoS with valid requests.
- UDP connection flood on random port.

Otherwise, we have designate Snort IDS tool by adding a new approach in snort detection, new rules are written for the DoS/DDoS attacks to avoid intrusions, and finally we have compare basic Snort IDS rules with new ones.

6. OPERATING ENVIRONMENT

The development work is carried out on Window Server platform in order to comply with the SNORT program version 2.9.7.

Workstation. *Software Components*: Windows 2012 server XEON Processor with 128Go RAM; ADODB; MySQL; WinPCap; Snort; ACID; JPGraph.

All attacks tested with students used different DoS/DDoS attack tools. DoS/DDoS tools have evolved to support more powerful attack methods.

- **Low Orbit Ion Cannon (LOIC),** which can be used as a botnet controlled via IRC protocol and has the basic flooding ability using TCP, UDP and HTTP methods [10].

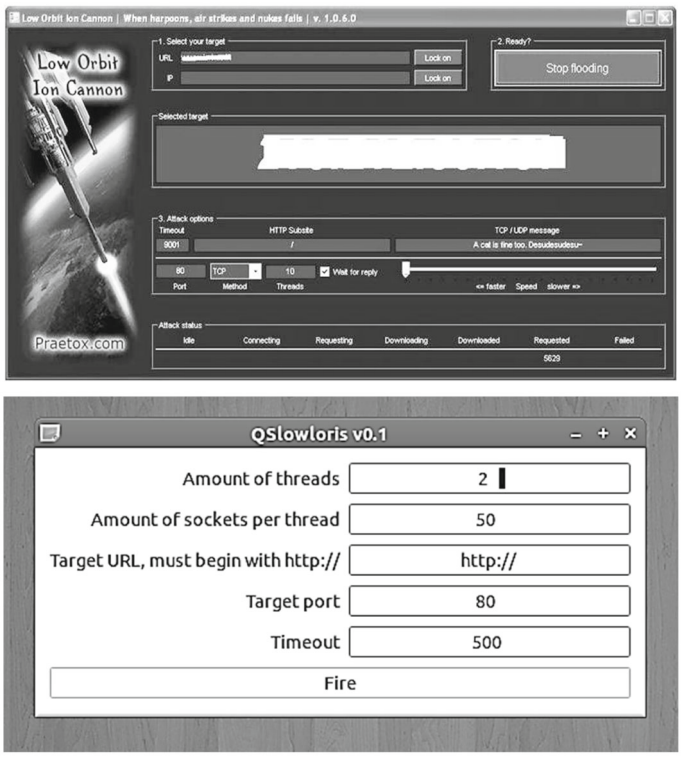

Fig. 4. LOIC and Slowloris in action.

• **HTTP POST DDOS**, Rate-limiting is enforced by classifying and keeping a watchful eye on the speed performance and size of each request, subsequently limiting the amount of extremely slow connections per CPU core and above the maximum allowable body.

• **HTTP GET DDOS**, IIS web servers or those having timeout limits for HTTP headers are usually not susceptible. Load balancers, reverse proxies, and Apache modules, such as mod_antiloris, might repel this kind of cyber-attack [11].

• **Slowloris**, DDoS attack software that enables a single computer to take down a web server. Due the simple yet elegant nature of this attack, it requires minimal bandwidth to implement and affects the target server's web server only, with almost no side effects on other services and ports, tested with web server of campus (MS IIS).

7. IMPLEMENTATION AND EXPERIMENTATION

7.1. DoS/DDoS Attacks

At first, the types of attacks that LOIC and Slowloris perform were the previous described attacks. It proposals its user an input field for IP address or domain of the target and several attack options containing protocol, port, number of threads and optional waiting for a reply.

To evaluate the output of LOIC (Fig. 4) three experiments same with Slowloris were setup:

UDP Attack: To perform the UDP attack, select the method of attack as UDP. It has port 80 as the default option selected, but you can change this according to your need. Change the message string or leave it as the default.

TCP Attack: This method is similar to UDP attack. Select the type of attack as TCP to use this.

HTTP Attack: In this attack, the tool sends HTTP requests to the target Web server.

The TCP (SYN flooding) option with 1000 and 10000 parallel connections, UDP flooding and the attack over the HTTP protocol with 1000 parallel connections are tested in these experiments. Throughout these experiments, the protocol messages were taken in a "winpcap" file. The firewall is not installed, which means without any traffic that does not belong to the attack. Whereas the attack was accomplished, the produced packets were captured to be analyzed in the real time.

LOIC/Slowloris experiments

Experiment	Method	Packets
1	TCP (1000)	210.421
	TCP(10.000)	225.942
2	UDP(75.000)	258.546
3	HTTP	8.445

Firstly, we tested the TCP method with 1000 parallel connections for each user. Analysis of the captured output reveals the following: Firstly starts sending the packets over the opened connection. This means it only opens one connection. After the initiation of the connection LOIC sends malformed TCP packets, the web server as HTTP manages these because LOIC sent the messages to port 80. After few seconds, the web server sent fault saying the request URL is too large and the TCP connection is closed. LOIC then started a new connection and restarted sending traffic. This procedure repeated a couple of times. In total 210.421 packets were captured, and again with same TCP, but with 10000 parallel connections. LOIC started sending packets right away without opening a connection. LOIC should have started a new connection but it possibly did not do due to a bug in the source code. The traffic looks similar to the traffic in the first experiment, where the web server also showed an error and restarted. However, we captured in the same interval only 225.942 packets in the first experiment, which means that the tool did not speed up the attack. Same experiment was with Slowloris tool.

In the second experiment, we tested the UDP method at the highest speed sends approximately 75.000 packets per second to the web server and approximately 250 at the lowest speed. In total 258.546 packets were captured.

The last experiment tested the HTTP attack, this attack showed a basic handshaking technique. After first a connection of HTTP GET request was sent, the Web server returned a HTML object and closed the connection. After that, the server did not respond for nearly a second. In total, there were 8.445 requests and the document was served 8.224 times.

The results of the experiments are presented in table.

7.1.1. CPU and memory load. This metric will evaluate the impact of attacking traffic on the physical resources of the web server system. The results for the CPU load experiment show clearly the impact of the attacking traffic on the consumption of system resources (Fig. 5). When we use TCP flooding, the CPU load increase when the rate of attacking traffic is 4000 packets per second rom 42% CPU load to 48%. In another end the UDP protocol, never stop its need of CPU load, UDP consume until 76% of CPU resources. Through the results that show an increased memory load as expected, the results are not significant enough because of the augmentation is in the order of 35 Mbytes for each protocol.

7.1.2. Campus network bandwidth. The figure below shows the traffic variation before and after the DDoS attack in the Main Protection System. We have simulated the DDoS attack incrementing the number of packets to the network to force the main protection system to reach the Black State. The figure shows how the traffic at the university campus network is increased rapidly after 2.1 s after executing the DDoS attack in PRTG graph (Fig. 6).

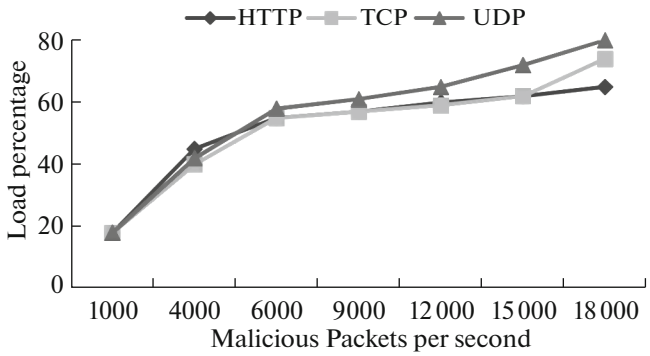

Fig. 5. CPU load results.

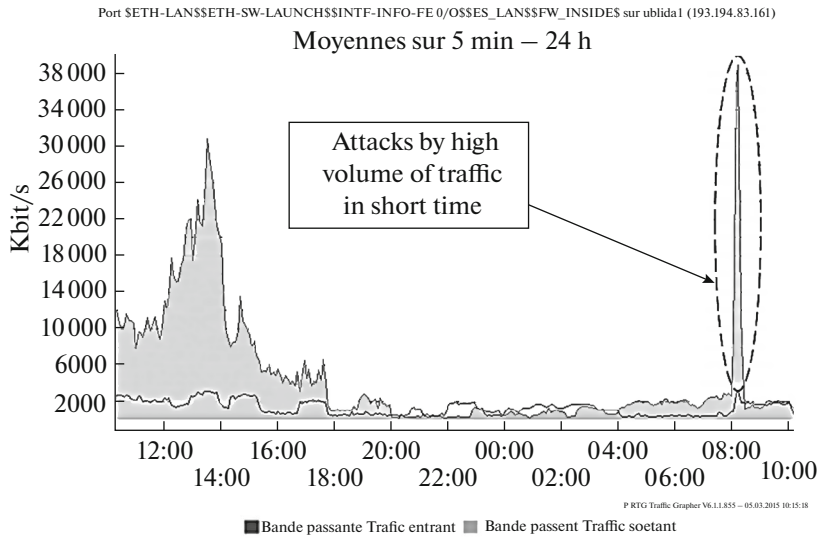

Fig. 6. Campus network traffic under attack.

7.2. DoS/DDoS Defense

All the components are combined together to make the application ready. Snort is configured to work in promiscuous mode to provide basis for intrusion detection. All packets captured by snort have to be logged. Modify the Snort configuration file, so that snort recognizes the path for rules (signatures), also, Snort should know local and external IP address. Signatures are written for the DoS/DDoS attacks to avoid intrusions. This alert is logged by default to var/log/snort. To make snort act as IDPS, it is configured in online mode, snort as that notifies the administrator of the intrusion and drops the packet entering the network. For this, configuring Iptables requires writing Iptables rules so that all captured packets are put on hold.

All the components of Snort are tested for their functioning. Snort is tested for packet capture. Signatures are tested with the captured packets. In case of intrusion, alerts are generated from snort that is then logged into output module. All these modules are tested and results are achieved.

The MySQL is used in the work to implement a relational database that stores information about captured packets and alarms is done if intrusion is detected. The database has four tables:

(1) TCPCapture table,

(2) UDPCapture,

(3) the table ICMPCapture,

(4) DDoS Alerts.

After all simulations attacks with DDoS tools (LOIC/Slowloris), we identify “DDoS Attack” using JavaScript embedded in a webpage Fig. 7.

This tool generates the following script request.

```
GET/app/?id=1292337572944&msg=BOOM%2520HEADSHOT!HTTP/1.1Host:www.univ-blida.dz
comUser-Agent: Mozilla/5.0 (PC; U; Intel Windows; en-US; rv:1.9.2.12)
Gecko/20101026Firefox/3.6.12Accept:ext/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8Accept-Language: en-us, en; q=0.5 Accept-Encoding: gzip, deflateAccept-Charset: ISO-8859-1,utf 8;
q=0.7, *; q=0.7Keep-Alive: 80Connection: keep-alive
```

7.2.1. Solution for UDP attack. UDP attack allowed us to extract in the image above the following two actions:

- UDP traffic at port 80.
- Many connections for small time.

We created the following Snort rule:

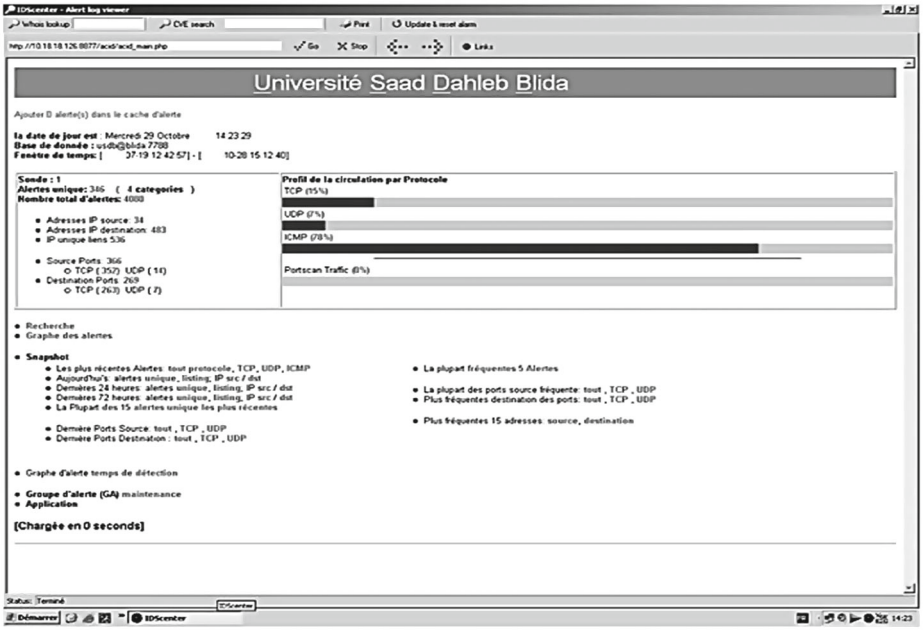

Fig. 7. Detection DoS and DDoS attack in the campus network.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:“SLR - LOIC DoS
Tool (HTTP Mode)”; flow: established,to_server; content:“|47 45 54 20 20 48 54 54  50 2f 31 2e
30 0d 0a 0d 0a 0d 0a|”; threshold: type threshold, track by_src, count 10, seconds 10; reference:
url, www.univ-blida.dz; classtype:misc-activity; sid:1234569; rev:1; )
```

snort -c snort-test.conf -A console -q -r /Blida1/LOIC/PCAP/LOIC-udp.pcap —o

```
alert udp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:“SLR - LOIC DoS Tool (UDP Mode)  - Behavior Rule
(tracking/threshold)”; threshold: type threshold, track by_src, count 100,
seconds 5; reference: url, www.uni-blida.dz; classtype:misc-activity;
sid:1234590; rev:1;)
01/24-14:58:38.849802  [**] [1:1234590:1] SLR - LOIC DoS Tool (UDP
Mode)  - Behavior Rule (tracking/threshold) [**] [Classification: Misc
activity] [Priority: 3] {UDP} 193.194.8.1:59022 -> 193.194.8.1:8001/24-
14:58:38.952511  [**] [1:1234590:1] SLR - LOIC DoS Tool (UDP Mode)
- Behavior Rule (tracking/threshold) [**] [Classification: Misc activity]
[Priority: 3] {UDP} 193.194.8.1:59022 -> 193.194.8.1:8001/24
-14:58:39.024253  [**] [1:1234590:1] SLR - LOIC DoS Tool (UDP Mode)
- Behavior Rule (tracking/threshold) [**] [Classification: Misc activity]
[Priority: 3] {UDP} 193.194.8.1:59022 -> 193.194.8.1:80
```

7.2.2. Solution for TCP flooding. Similar to the UDP attack. Because of this, detection based on a keyword is somewhat difficult.

The following behavior detected:

- ACK TCP flag set.
- Packet size of 1448.

The following screenshot shows this:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:“SLR - LOIC DoS
Tool (HTTP Mode)”; flow: established,to_server; content:“|47 45 54 20 20 48 54 54  50 2f 31
2e 30 0d 0a 0d 0a 0d 0a|”; threshold: type threshold, track by_src, count 10,
seconds 10; reference: url, www.univ-blida.dz ; classtype:misc-activity; sid:1234569; rev:1;)
```

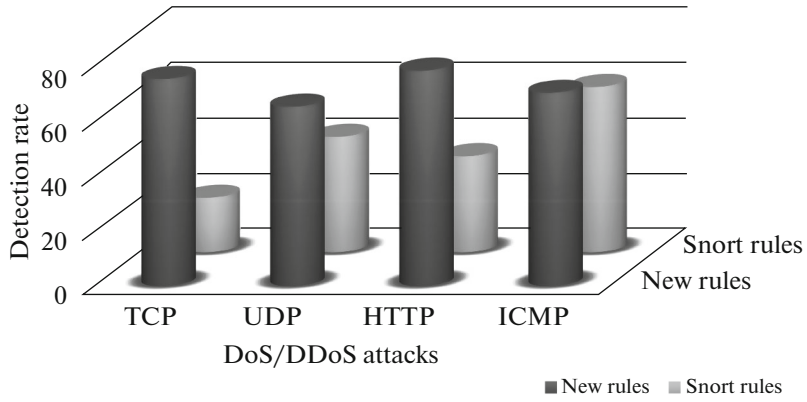

Fig. 8. Comparison between Snort rules and new ones.

Based on this criterion, we created the following Snort rule:# snort -c snort-test.conf -A console -q -r /Blida1/LOIC/PCAP/LOIC-tcp.pcap –o

```
1/24-14:59:11.405063  [**] [1:123456:1] SLR - LOIC DoS Tool (TCP Mode)  - Behavior Rule
(tracking/threshold) [**] [Classification: Misc activity] [Priority: 3] {TCP} 193.194.8.1:55331 ->
193.194.8.1:8001/24-14:59:11.996198  [**] [1:123456:1] SLR - LOIC DoS Tool (TCP Mode)  - Behavior
Rule (tracking/threshold) [**] [Classification: Misc activity] [Priority: 3] {TCP} 193.194.8.1:55331 ->
193.194.8.1:8001/24-14:59:12.318804  [**] [1:123456:1] SLR - LOIC DoS Tool (TCP Mode)  - Behavior
Rule (tracking/threshold) [**] [Classification: Misc activity] [Priority: 3] {TCP} 193.194.8.1:55332 ->
193.194.8.1:80
```

7.2.3. Http Filter. We created the follow snort rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:“SLR - LOIC DoS Tool (UDP
Mode)  - Behavior Rule (tracking/threshold)”; threshold: type threshold, track by_src, count 100, seconds 5;
reference: url, www.univ-blida.dz; classtype:misc-activity;  sid:1234590; rev:1;)
```

snort -c snort-test.conf -A console -q –r /Blida1/LOIC/PCAP/LOIC-http.pcap –o

```
01/27-11:57:52.977537  [**] [1:1234569:1] SLR - LOIC DoS Tool (HTTP Mode) [**] [Classification:
Misc activity] [Priority: 3] {TCP} xxx.xxx.xxx.xxx:55178 -> xxx.xxx.xxx.xxx:8001/27-11:57:54.184679
[**] [1:1234569:1] SLR - LOIC DoS Tool (HTTP Mode) [**] [Classification: Misc activity] [Priority: 3]
{TCP} xxx.xxx.xxx.xxx:55188 -> xxx.xxx.xxx.xxx:8001/27-11:57:55.111591  [**] [1:1234569:1] SLR -
LOIC DoS Tool (HTTP Mode) [**] [Classification: Misc activity] [Priority: 3] {TCP}
xxx.xxx.xxx.xxx:55198 -> xxx.xxx.xxx.xxx:80
```

The host running Snort should be installed on the port of a switch to be able to monitor and analyze the network traffic exchanged. Snort uses a database of our new rules to detect this attack.

For any DDoS tools, we generate this following Snort rule, it used to detect DDoS attack traffic:

```
alert tcp any any -> any any (msg:“TCP SYN flood attack detected”; flow: stateless; flags:S,12;
threshold: type threshold, track by_src, count 3, second 1; classtype: attempted-recon; sid:10002;rev1;).
alert tcp any any -> any any (msg:“SLR - LOIC DoS Tool HTTP Mode)”; flags:PA; content:“GET /
HTTP/1.0”; sid:1234569; rev:1;)
alert tcp any any -> any any (msg:“SLR - LOIC DoS Tool TCP Mode)”; flags:PA; content:“|65 73 75 64
65 73 75 64 65 73 75 7e|”; sid:1234570; rev:1;)
alert udp any any -> any any (msg:“SLR - LOIC DoS Tool UDP Mode)”; content:“|65 73 75 64 65 73 75
64 65 73 75 7e|”; threshold: type threshold, track by_src, count 100 , seconds 5; sid:1234571; rev:1;)
```

In the first three experiments, the TCP/UDP attack was selected and the TCP/UDP filter was enabled. In the last experiment, the HTTP method was selected and the HTTP filter of SNORT was enabled. After capturing and analyzing the baseline attack (without any protection). Snort new rules were

setup and tested. Figure 8 shows that the new snort rules generated were efficient in finding ICMP attack, SYN flood attack, UDP attack and HTTP get, this new rule sets available to protect against DDoS attacks executed using LOIC and gives the better results.

Based on the results, the effectiveness of the proposed mitigation rules to mitigate DDoS attack within the tested detection rate has improved approximately 43.95%. As opposed to the classic rules DDoS defense mechanism snort, the proposed new rules can accurately detect TCP, UDP, Http attacks and our scheme in early mitigating ICMP attacks.

8. CONCLUSION

Denial of service attacks and specially Distributed Denial of service attack are hazardous for the internet and web services. According to the surveys, the percentage of attacks is at exponential rise with new and sophisticated techniques. This is a problem when security students are exposed to several DoS and DDoS tools on offensive techniques, it is necessary that students know how to attack and anatomize offensive techniques to truly understand how to defend networks and computer systems, and strengthen their security skills. This research aimed at evaluating some defense methods against DoS and DDoS attacks executed using LOIC and Slowloris, pointing out which one is the most effective. This tool supports three different types of attack. The only difference between them is the protocol used to send messages to the targets, where TCP, UDP or HTTP can be selected. The second sub question was answered by showing that SNORT, which is currently one of the most used network intrusion detection systems, has already rule sets available to protect against DDoS attacks executed using LOIC or Slowloris but with our new rules, the detection rate has improved approximately 43.95%. Nevertheless, in this research only SNORT was tested. There are several other intrusion detection systems, which could be used against LOIC or Slowloris. Their effectiveness has to be ascertained as well.

REFERENCES

1. Meena, D. and Jadon, R.S., Distributed denial of service attacks and their suggested defense remedial approaches, *Int. J. Adv. Res. Comput. Sci. Manage. Stud.,* 2014, vol. 2, no. 4, pp. 183–197.
2. Patrikakis, C., Masikos, M., and Zouraraki, O., Distributed denial of service attacks, *Int. Protoc. J.,* 2004, vol. 7, no. 4, pp. 183–197.
3. Computer Emergency Response Team CERT. *DDoS Overview and Incident Response Guide*, Computer Emergency Response Team, 2014.
4. Ling, Y., Gu, Y., and Wei, G., Detect SYN Flooding Attack in Edge Routers, *Int. J. Secur. Appl.,* 2009, vol. 3, no. 1, pp. 31–45.
5. Choi, Y. seo., Kim, I.-K., Oh, J.-T., and Jang, J.-S., Aigg threshold based http get flooding attack detection, *Proc. of the 13th International Workshop on Information Security Applications,* Jeju Island, 2012, vol. 7, pp. 270–284.
6. Peng, T., Leckie, C., and Ramamohanarao, K., Survey of network-based defense mechanisms countering the DoS and DDoS problems, *J. ACM Comput. Surv.,* 2007, vol. 39, no. 1, pp. 1–42.
7. Mirkovic, J., Martin, J., and Reihe, P., *A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. Technical Report #020018,* Computer Science Department University of California, Los Angeles, 2014.
8. Lanke, N.M. and Raja Jacob, C.H., Detection of DDOS attacks using Snort Detection International, *J. Emerging Eng. Res. Technol.,* 2014, vol. 2, no. 9, pp. 13–17.
9. Zeng, X., Peng, X., Li, M., Xu, H., and Jin, S., Research on an effective approach against DDoS attacks, *Int. Conf. Res. Challenges Comput. Sci.,* 2009, pp. 21–23.
10. Alomari, E., Gupta, B.B., and Karuppayah, S., Botnet-based distributed denial of service (DDoS) attacks on web servers: Classification and Art, *Int. J. Comput. Appl.,* 2012, vol. 49, no. 7, pp. 24–32.
11. Zargar, S.T., Joshi, J., and Tipper, D., A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, *IEEE Commun. Surv. Tutorials,* 2013, vol. 5, no. 4, pp. 2046–2069.