Firewall Assignment Using Raw Sockets

CS6903 - Network Security

Submitted by Tahir Ahmed Shaik (CS20MTECH14007), Pratik Madhukar Lahase (CS20MTECH11003), Jaykishan Pipaliya (CS20MTECH11012)

01/05/2021

Abstract:

This assignment report illustrates the scenario of the working of a firewall system between a set of hosts using the rule sets that are fed to the system. Firstly we implement a simple firewall, which relies on the idea of pre-coded/hot coded rules using conditional statements. Then we try to implement a more advanced and sophisticated system that works on the dynamic set of rules and defines a complete firewall system that provides rule management (CRUD of Rule sets), Command line interface for user interaction. We then also benchmark the designed system across a set of parameters and determine the PPS (Packets Per Second) processing by the system by plotting a set of graphs. The system is further extended to employ the DoS attack detection mechanism.

Network Configuration and setup :

To simulate the system , we devise two virtual machines , where one machine acts as the firewall, while another machine is the internal host that we intend to protect from the external host. The external host in this scenario is the host system. The following configurations are done to the networking entities.

Firewall : **IP :** 10.0.0.1/24 **Hostname :** firewall	**Internal Host (Host1)** **IP :** 10.0.0.253 Hostname : host1	**External Host:** **IP :** 192.168.1.7

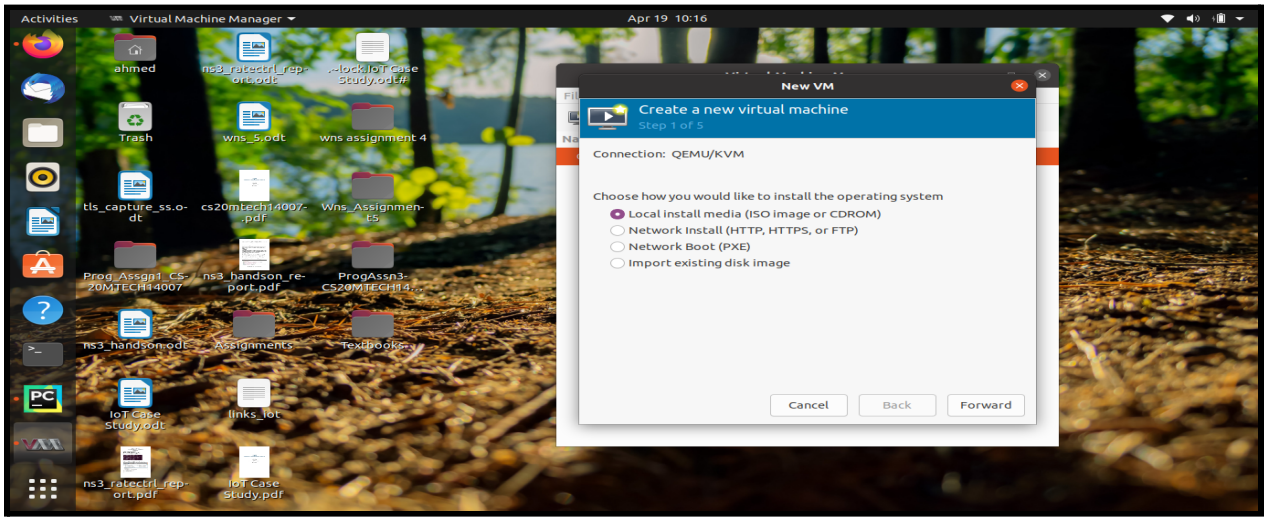


Fig 1 : Installing VM using virt-manager

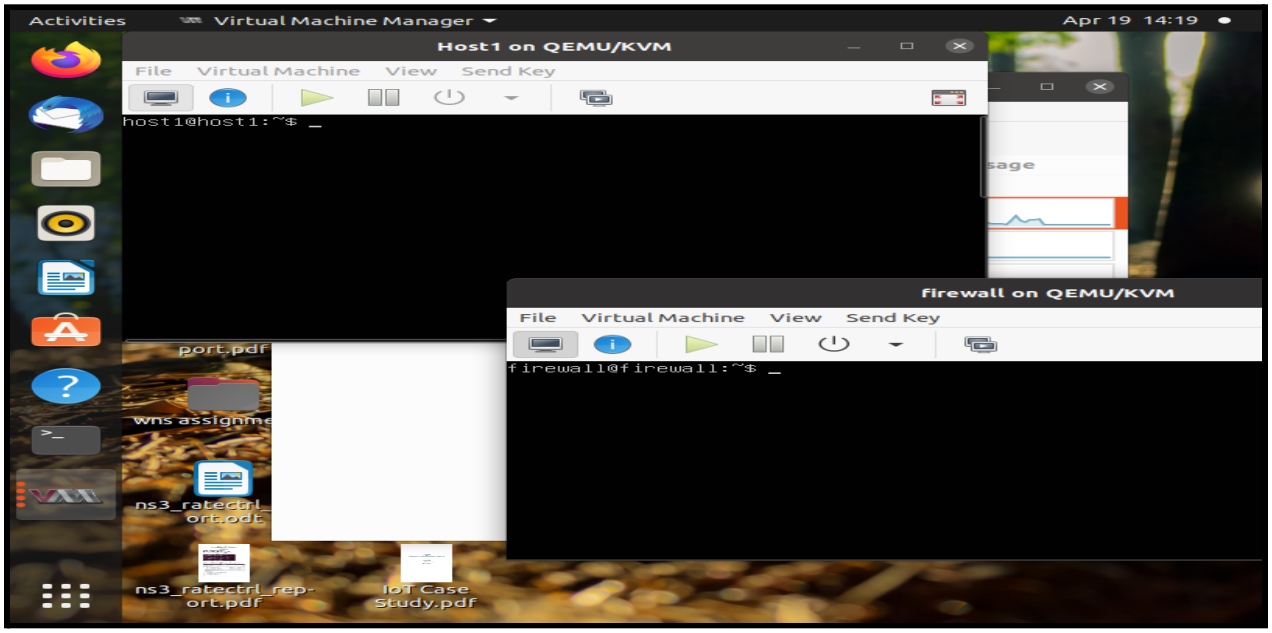
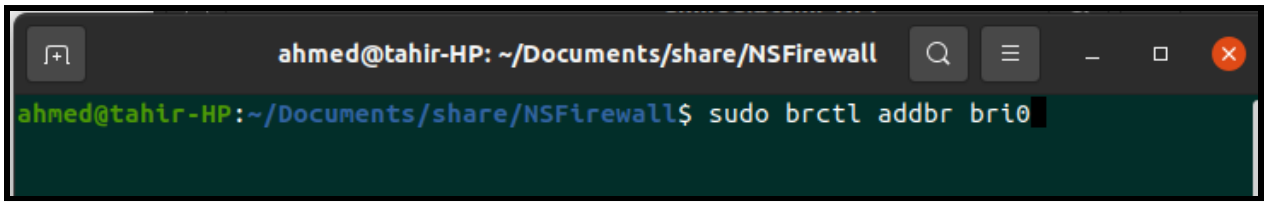


Fig 2 : Installed VM's Setup

The networking hosts are connected to the firewall using the set of bridges (bri0 and bri1). These bridges are created as shown below. **Brctl** tool is used for creating the bridges.



Similarly the bri1 bridge is created and configured with the IP addresses. The following figure shows the complete network configuration of the setup. The firewall has two interfaces to connect to the entities.

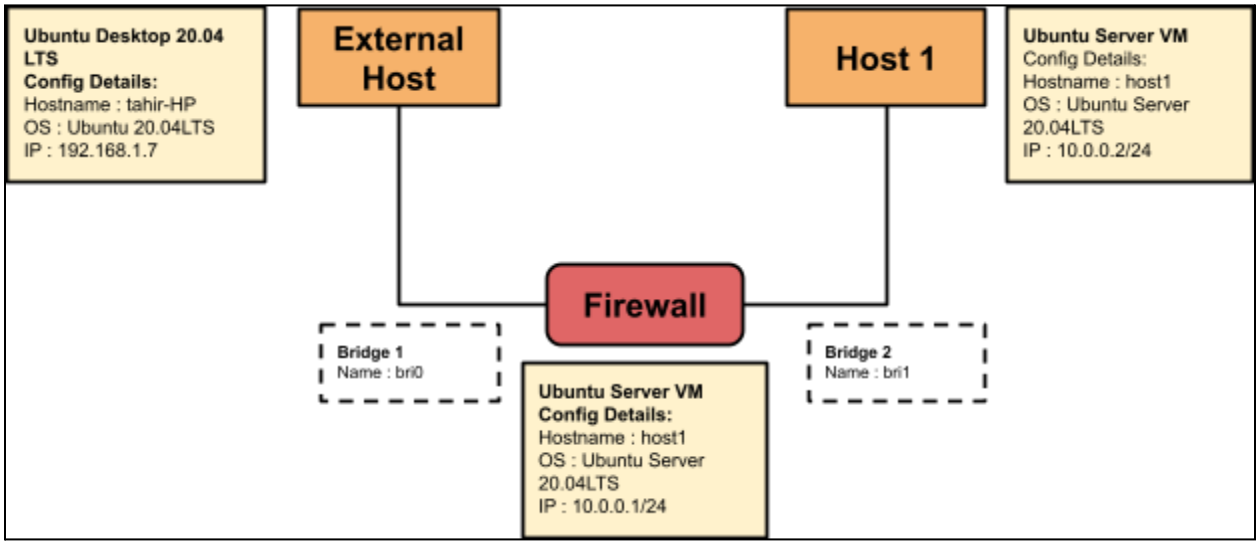


Fig 3: Network Setup

Task 1 : Running Simple Firewall at layer 2 level

In this task the firewall is designed at the layer 2 ethernet level that works filtering using the MAC addresses. The simple firewall class in the "firewall.py" code implements this design and system. This is executed as follows:

Command : python firewall.py simple_firewall <external_host_interface> <internal_host_interface>

This has pre-coded / hot coded logic to allow and deny the packets based on the MAC. We set to allow the IP packets for the external hosts MAC and the internal hosts MAC as shown below.

```
def decideRule(self,raw_data):
    eth = self.parseEtherHead(raw_data)
    if (eth[2] == socket.ntohs(0x0800) and eth[1] == "16:c0:f0:86:30:66"):   #Rule1 : Allow IP from external host
        allow = True
        packet_type = "External"
        self.host1sock.sendall(raw_data)

    elif(eth[2] == socket.ntohs(0x0800) and eth[1] == "6e:55:80:d6:df:9c"): #Rule2 : Allow Host1 Packets from host1
        allow = True
        packet_type = "Internal"
        self.host1sock.sendall(raw_data)

    else:                                           #Rule 5: Disallow all external packets
        allow = False
        packet_type = "External"

    return allow,packet_type
```

Fig 4 : Hard Coded logic filtering for simple firewall

The main interface of the simple firewall is shown below.

Fig 5: Simple Firewall CLI main screen

We run the firewall, and from the internal interface send a packet using the nping command “nping -e enp1s0 192.168.1.7”. This packet is captured by the firewall first and according to the rule it is allowed as shown below.

Fig 6: Packets allowed by firewall

Task 2: Advanced Firewall System (Filtering at layer 2[Ether], 3 [IP] and 4 [TCP,UDP]):

In this task we move ahead to design a more sophisticated system, that allows dynamic rule management (ADD, UPDATE and DELETE) , CLI interface, Saving and loading of the rules. The advanced firewall mode is executed in the “firewall.py” code as follows.

Command : python firewall.py adv_firewall <external_host_interface> <internal_host_interface>

The main screen of CLI interface of the system is as shown below in the following figure.

Fig 7 : Main Screen

The start firewall starts the firewall capturing and decides based on the rules defined. Print Firewall description prints a short firewall description. The Manage rules opens up the Rule manager, that allows to add, update and delete rules and print the rules present in the system. The statistics option prints out various statistical performance metrics of the system and also generates the plots. Save, and load rules are used to save and load the rules from the file in **json format.**

IP rule emulation:

Let us first emulate an IP rule first to filter the packets, this is as follows.

Fig 8 : Rule Manager(1 to add rule)

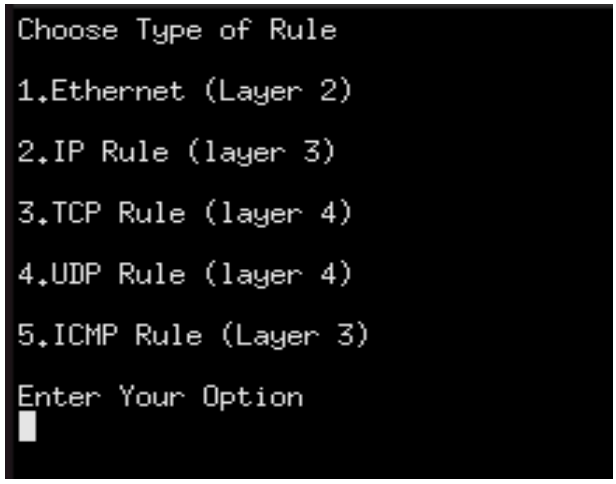


Fig 9 : Choosing IP rule

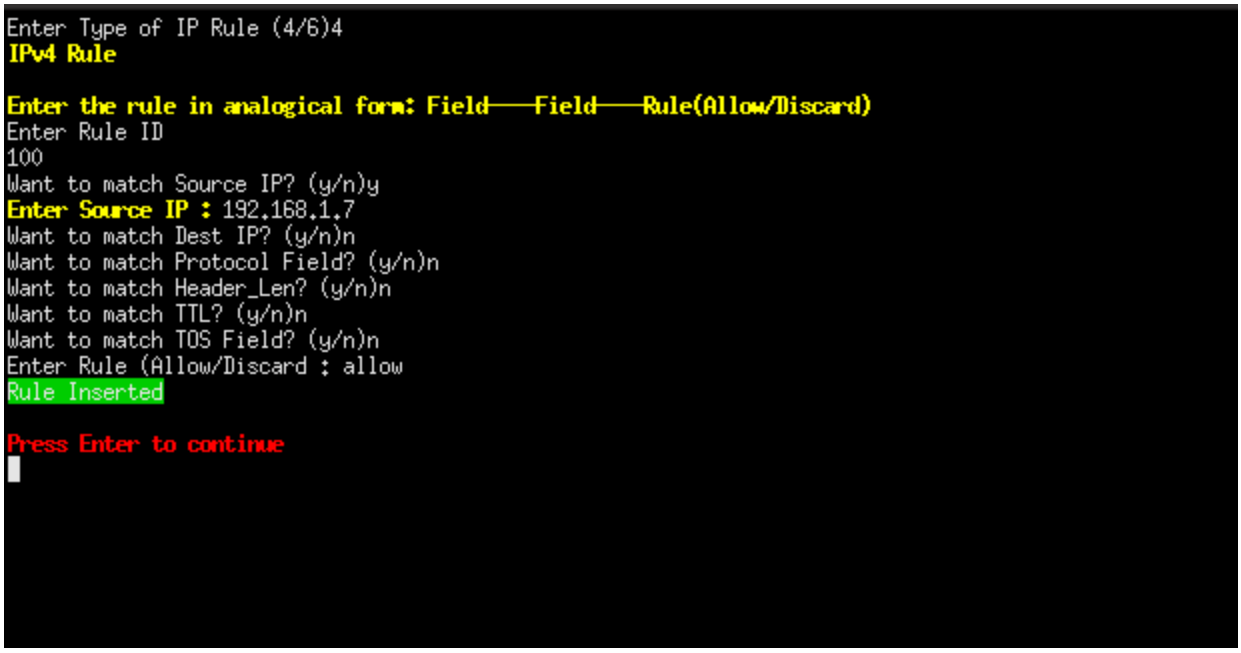


Fig 10 : Rule entered to the system

This rule allows the external host's IP packets having only IP 192.167.1.7, all other packets are discarded. We start the firewall from the main screen and can see the results as follows.

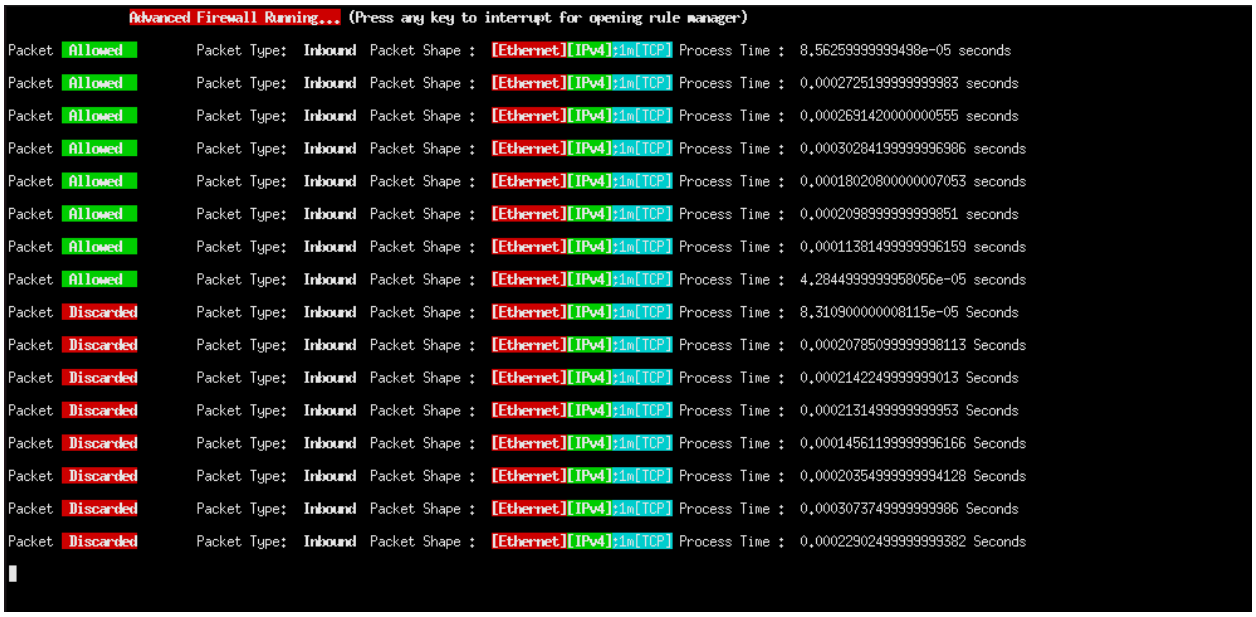
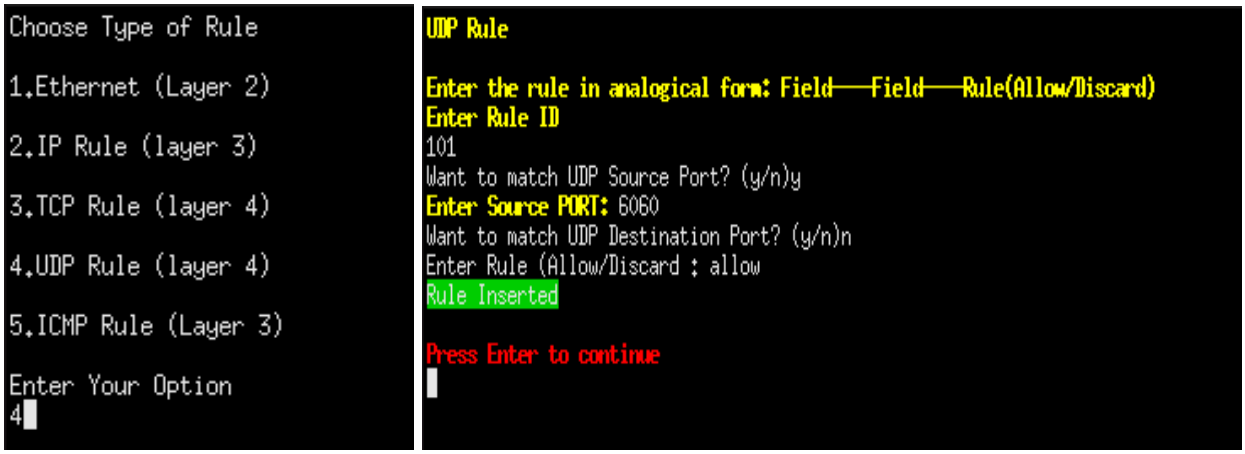


Fig 11 : Filtering Packets based on IP rule

More matching fields can be added such as TTL, TOS, HLEN etc for the IP rules. Both IPv4 and IPv6 rules can be filtered in the system.

UDP Filtering / UDP rule emulation :

We try to run the system on a UDP rule set as follows.



Running the nping command using the source port 6060 and trying with other ports is as follows:

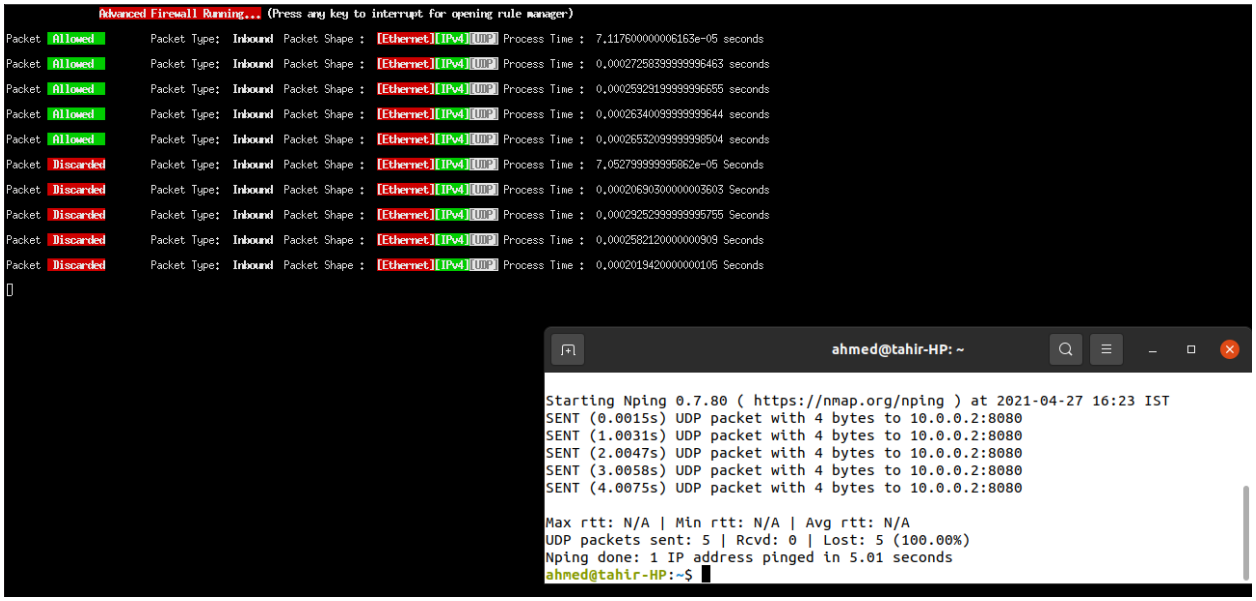


Fig 12 : UDP packets filtering

TCP Packet Filtering / TCP rule emulation :

We now try to observe the system against TCP packets by defining the TCP layer rules in the system. This is shown below.

Fig 13 : Choosing TCP rule

Fig 14: TCP rule defined

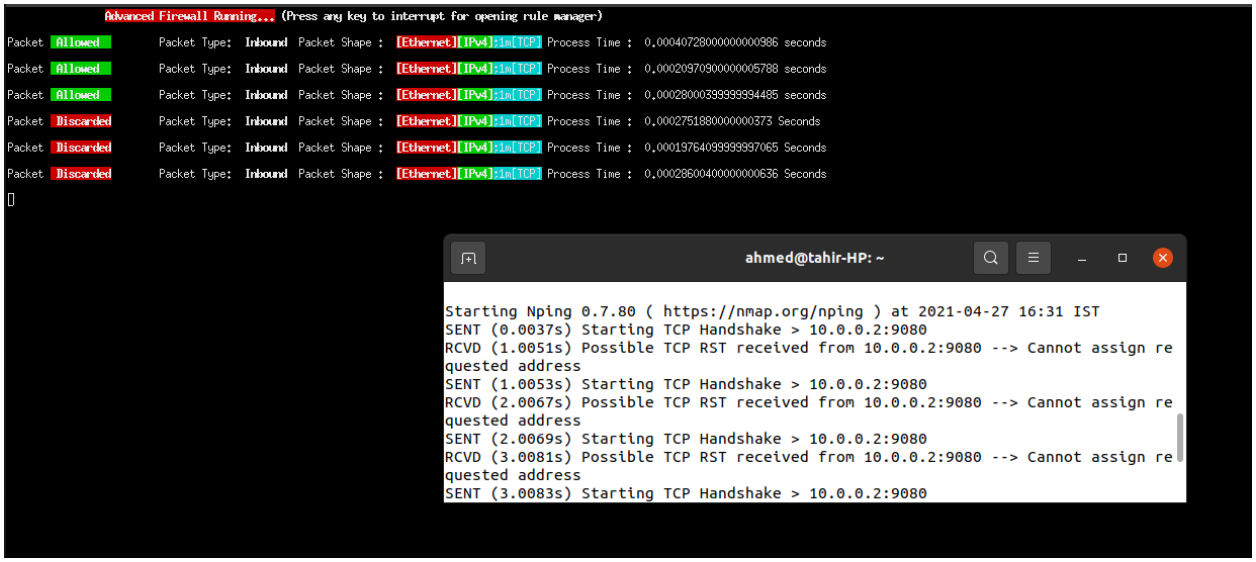


Fig 15: TCP packet filtering

ICMP Filtering/ ICMP rule emulation :

Defining the ICMP rules in the system to check filtering against ICMP protocol packers. The ICMP is identified from the IP packet using the protocol number **'1'.**

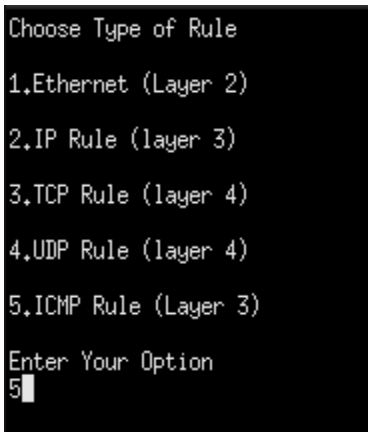


Fig 16 :ICMP rule selection

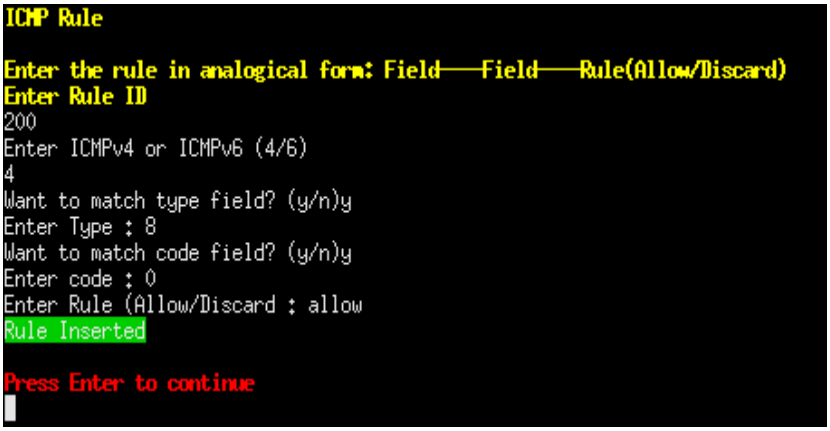


Fig 17: ICMPv4 Rule definition

Fig 18: ICMPv4 Filtering rules

Discarding ICMP requests from external host:

We define the following rule that doesn't allow the ICMP echo requests to the internal host.

Fig 19 : ICMP Discarding echo requests

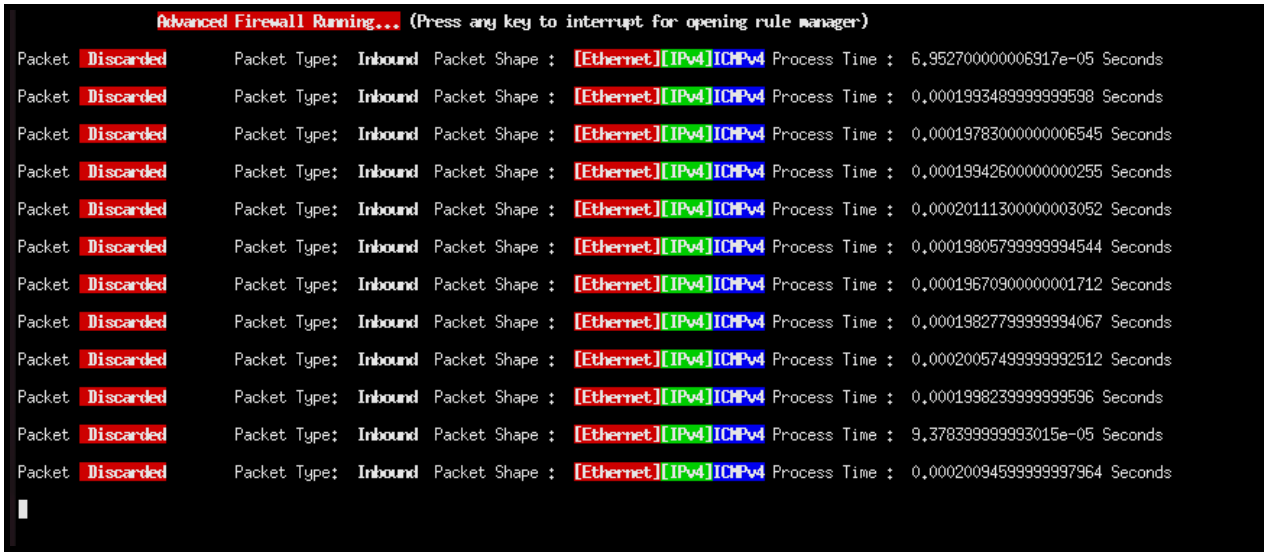


Fig 20: Discarding ICMP echo requests

Task 3 : Benchmarking and performance analysis of advanced firewall

In this task, we try to benchmark the system with respect to the packet processing powers, packet per second handling by the system, and also the performance of the system with respect to the number of rules and matching fields in the system.

Tool used for benchmarking : nping from nmap repository to generate various packets.

```
ahmed@tahir-HP:~$ nping --version

Nping version 0.7.80 ( https://nmap.org/nping )
```

Test 1 :

Metric	Value
Packet probe delay	500ms
Number of Packets	1000
Running Time	50 Seconds

The following results are observed for the above test.

```
**********************************************

                STATISTICS

**********************************************

The Statistics of the system are as follows

Average Time Taken to process packet :  0.00020567615895372045

No of packets allowed :  496

No of packets dropped :  498

No of rules in system :  1
Maximum Matching Fields in Rules :  1

Press Enter to continue
```

Fig 21: Statistics Display

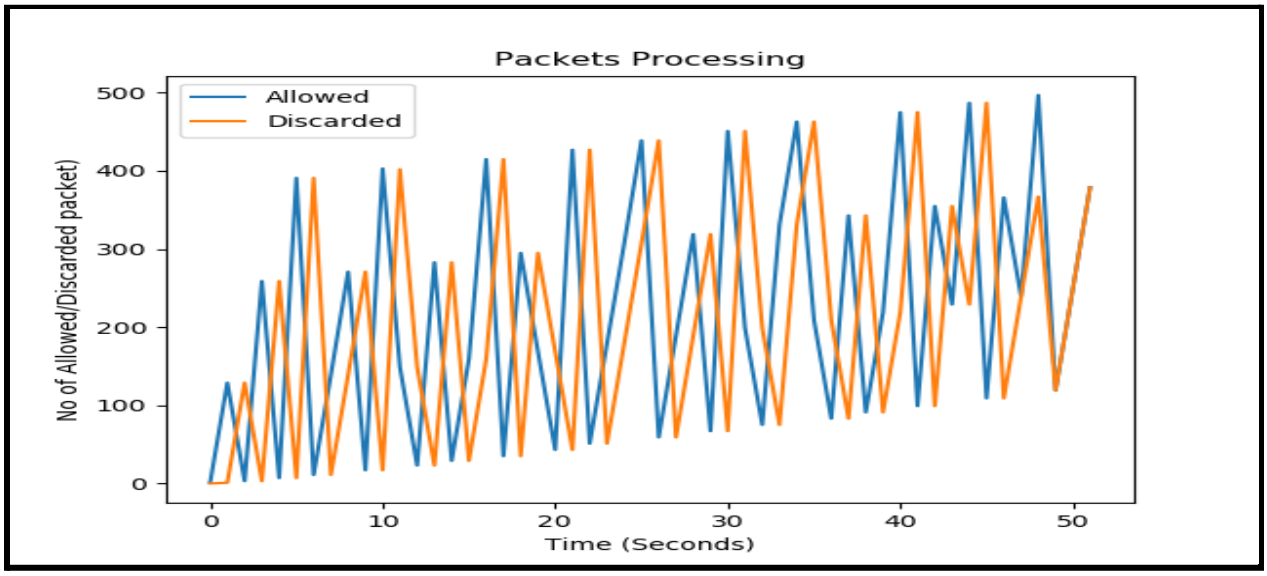


Fig 21 : Packet Processing Plot w.r.t Allowed and Discarded packets

The PPS for this case is around : **> 100 packets**

Test 2:

Metric	Value
Packet probe delay	100ms
Number of Cycles	1000
Running Time	50 Seconds

Statistics Observed :

```
**********************************************

                STATISTICS

**********************************************

The Statistics of the system are as follows

Average Time Taken to process packet :  0.00019327437830633605

No of packets allowed :  2437

No of packets dropped :  2440

No of rules in system :  1
Maximum Matching Fields in Rules :  1

Press Enter to continue
```

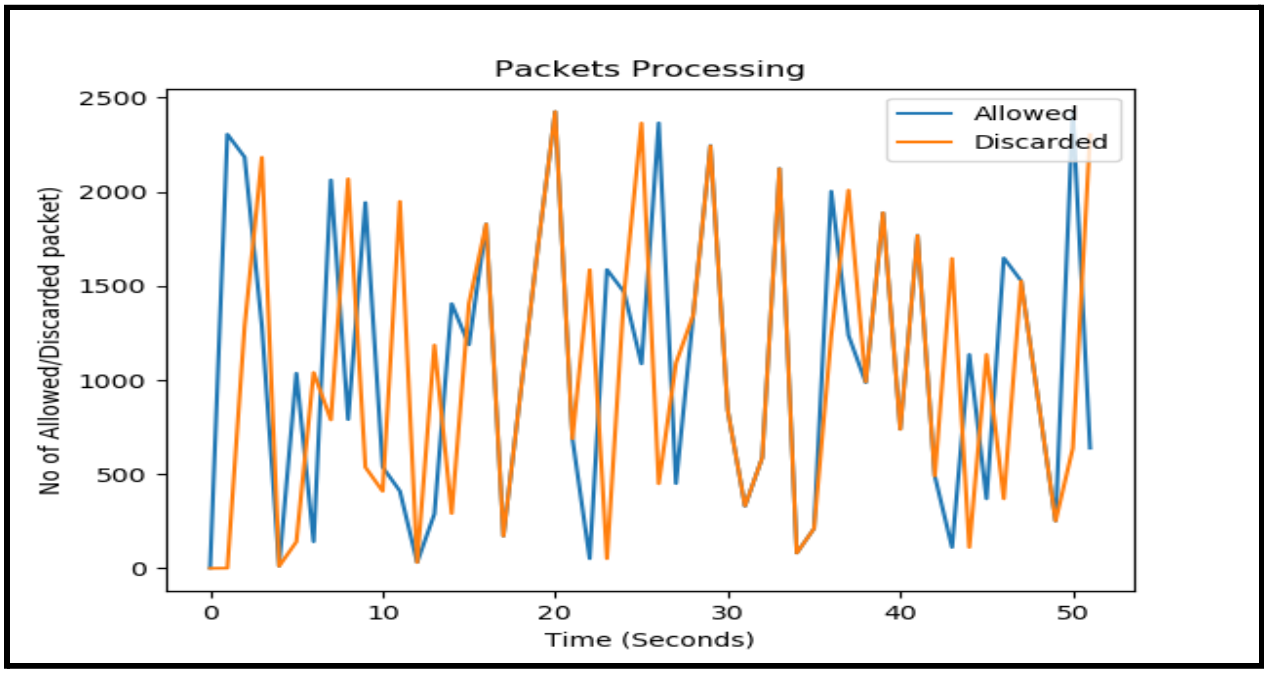


Fig 22 : Packets Processing

The PPS observed for this test case is : **>500 packets**

Test 3:

Metric	Value
Packet probe delay	10ms
Number of Cycles	1000
Running Time	20 Seconds

```
************************************************

                STATISTICS

************************************************

The Statistics of the system are as follows

Average Time Taken to process packet :  0.000161552705032617

No of packets allowed :  3218

No of packets dropped :  3220

No of rules in system :  1
Maximum Matching Fields in Rules :  1

Press Enter to continue
```

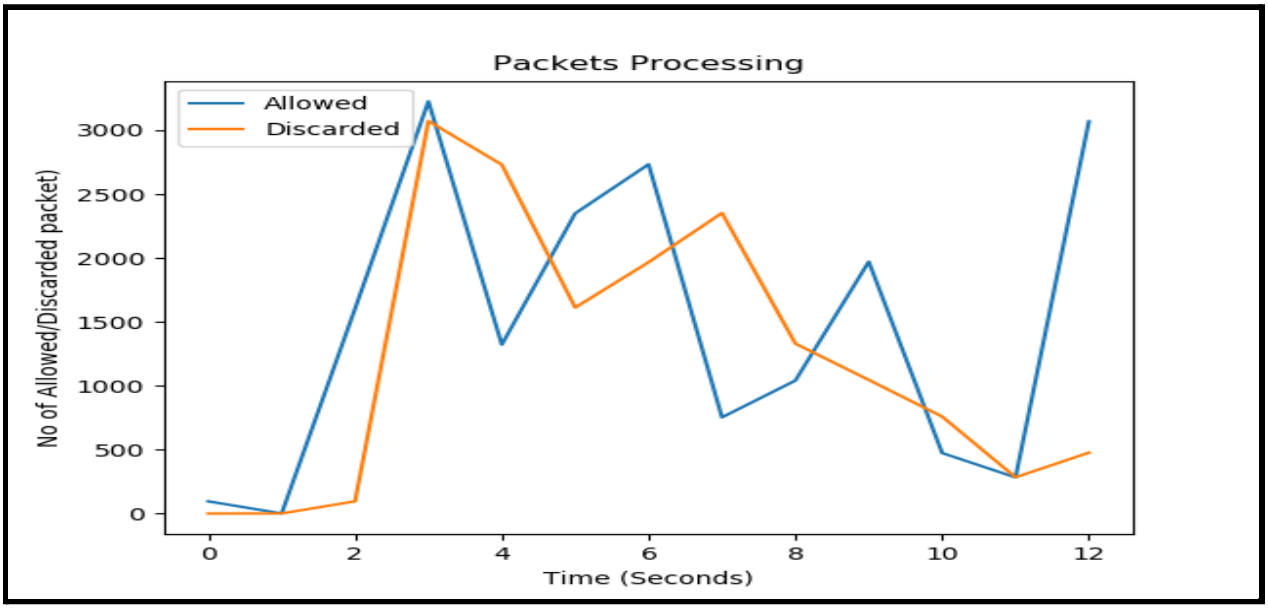


Fig 23 : Packet Processing with 10ms delay

The observed PPS for this case is : **> 1000 packets**

Performance of system w.r.t number of rules :

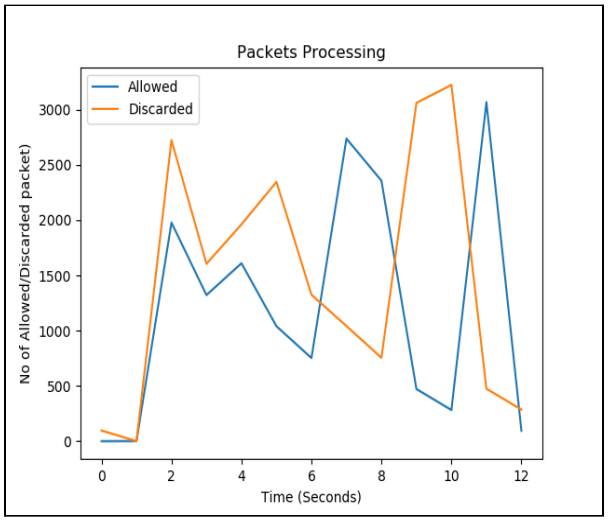


Fig 24: No of rules : 5, PPS observed : >2000

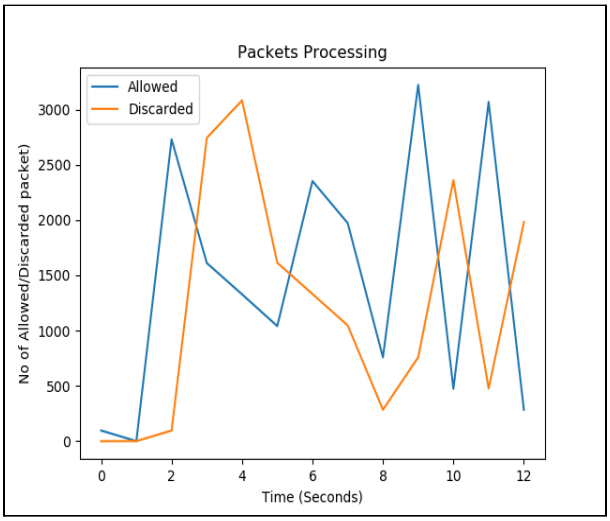


Fig 25: No of rules : 10, PPS observed : >1000

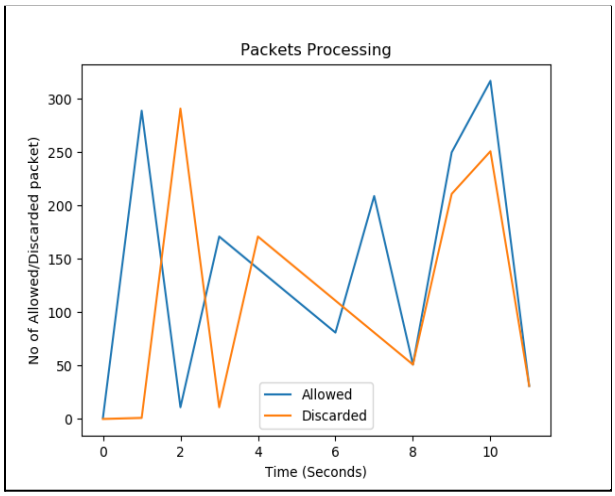


Fig 26: No of rules : 15, PPS observed : >250

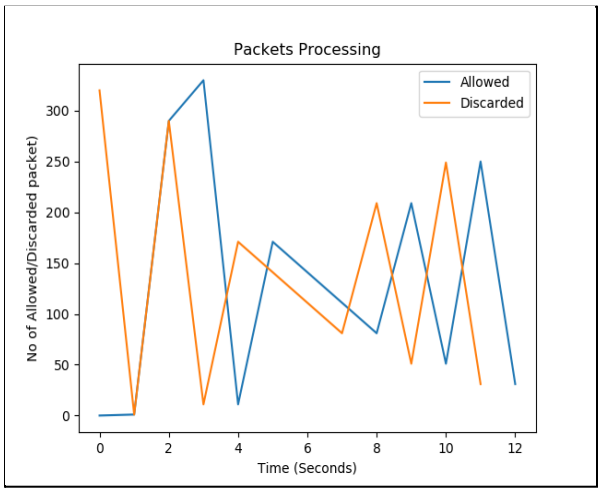


Fig 27: No of rules : 20, PPS observed : >150

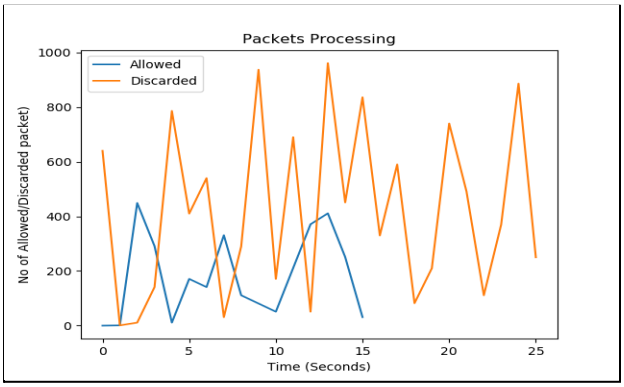


Fig 28: No of rules : 50, PPS observed : >100

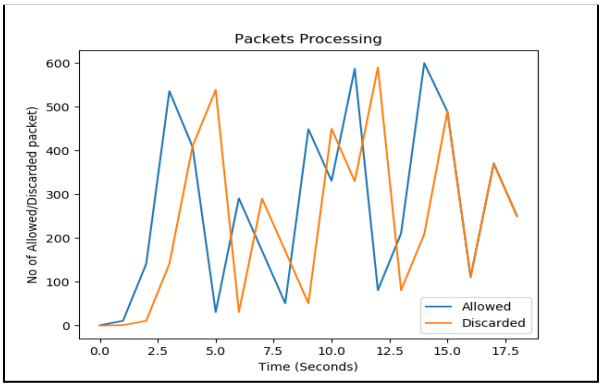


Fig 29 : No of rules : 100, PPS observed : >50

Observation :

From the above performance plots, we clearly observe that as the number of rules tend to increase the ***PPS decreases*** since the matching fields increase and a greater amount of time is spent in comparing the fields and making decisions against the rules defined by the system.

Task 4-b : Attack Detection (CHOSEN ATTACK : DoS)

In this task, the advanced firewall system is extended to detect certain attacks that exist in the networking communication. We choose to employ the DoS attack detection in the firewall. We first understand the DoS attack as follows:

DoS (Denial of Service Attack)

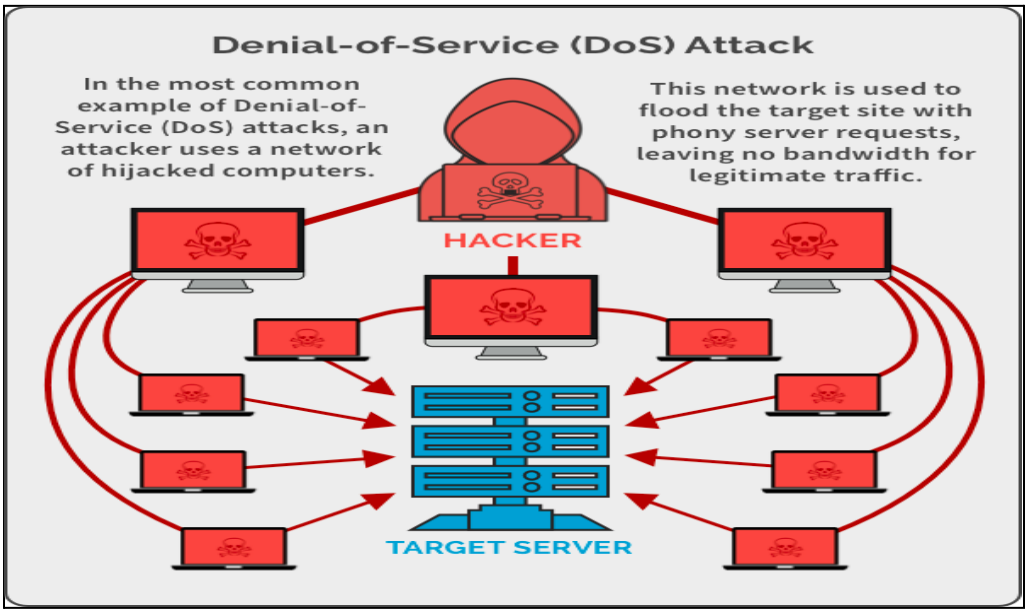


Fig 30 : DoS attack (Source Via https://spanning.com/)

The DoS attack or Denial of Service attack is a type of attack where an attacker overpowers or generates a heavy amount of requests to a server, which causes the server to exhaust all its resources to prepare responses to the heavy requests, thereby making the server unserviceable to further incoming genuine requests.

Design and Implementation of the detection mechanism:

Now to make our system be able to detect the DoS attack and mitigate it, we first create a dos_track map which is a dictionary that basically holds the count of each source IP that enters the system. Then a certain dos_threshold is also defined in the system initially from the options of Firewall. During the packet decision process, after the rules checking is completed, the system further checks if the IP count is still lesser than the threshold limit. If the threshold limit is exceeded a DoS error is generated and the packet is dropped.

Entity	Value
Tool Used	Nping <https://nmap.org/nping/>
Probe Delay b/w packets	200ms
Threshold Limit	20

The following workflow shows the DoS detection by the system.

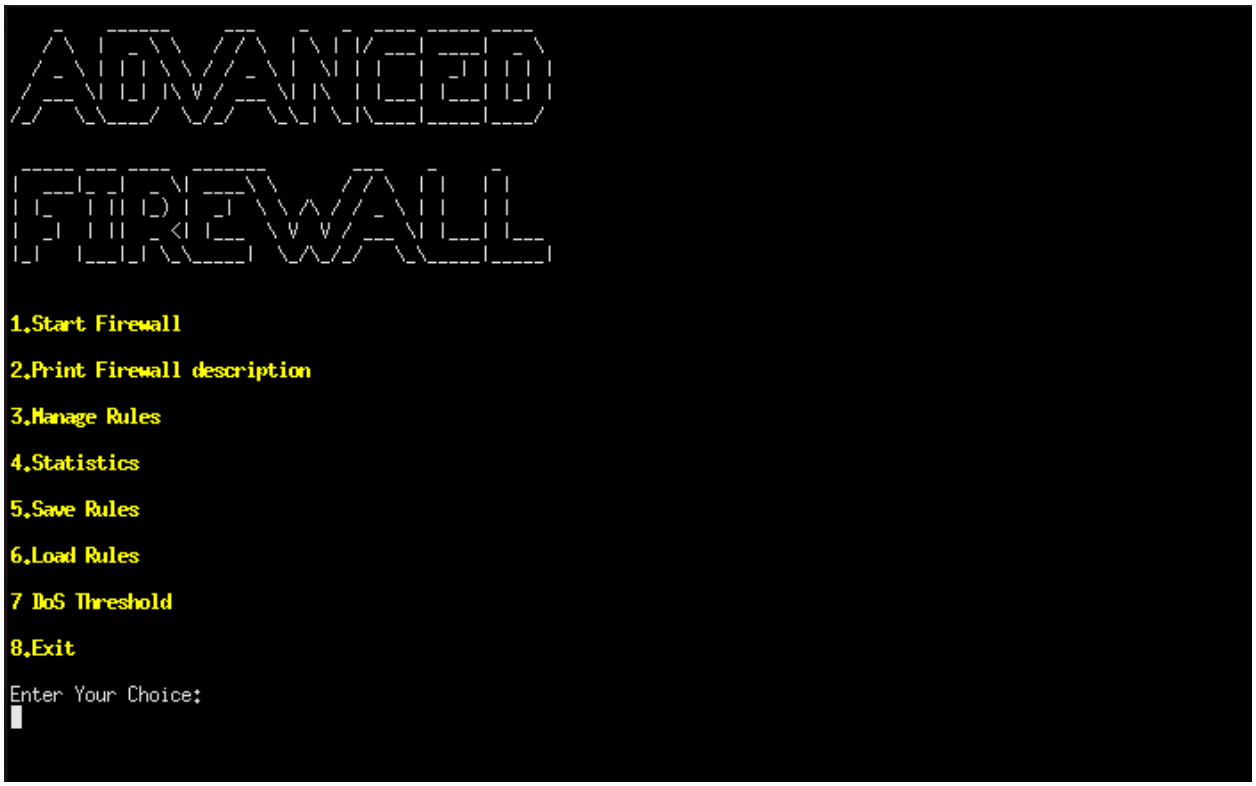


Fig 31 : Setting DoS threshold

Fig 32 : DoS threshold

Next we send a continuous stream of packets from the external host with a delay of 200ms using the ***nping utility by Nmap.*** The firewall after a threshold limit of 20 discards the packet as shown below.

Fig 33 : DoS detected by Firewall

Technical Specifications:

Configuration Specifications	
Host Operating System	Ubuntu Desktop 20.04 LTS
VM Configurations	**Firewall:** **OS :** Ubuntu Server 20.04 LTS --- **Host1:** **OS :** Ubuntu Server 20.04 LTS
Programming Language	Python 3.8
Tools Used	
Performance Benchmarking	Nping Tool Version 0.7.08 **(Nmap)**
DoS simulation	Nping Tool Version 0.7.08 **(Nmap)**
Libraries Used	
Matplotlib.pyplot (Graph Plots), socket (Raw Socket Programming), pyfiglet (For Banner on CLI), json (Rule storing)	

References :

1. https://nmap.org/
2. https://nmap.org/nping/ [Nping Utility]
3. https://virt-manager.org/ [Virt manager]
4. https://en.wikipedia.org/wiki/Denial-of-service_attack

Conclusion :

Through this assignment and experimentation we understand the working and design of firewall systems, to filter network packets across various domains. We analyse the performance metrics of the firewall system and understand the causes behind the observances that we come across. We also get to know the attacks that are possible and identify the mitigation strategies to such attacks.

Appendix:

More Options in CLI of Advanced Firewall

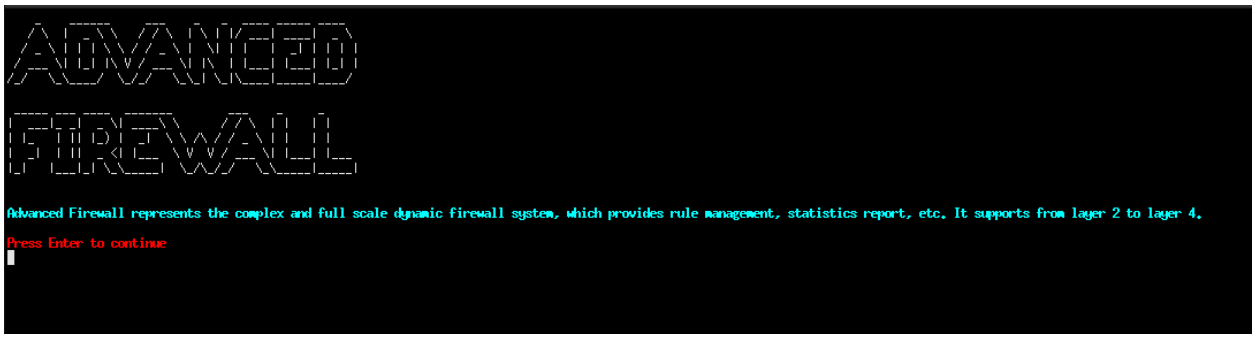


Fig a : Firewall Description

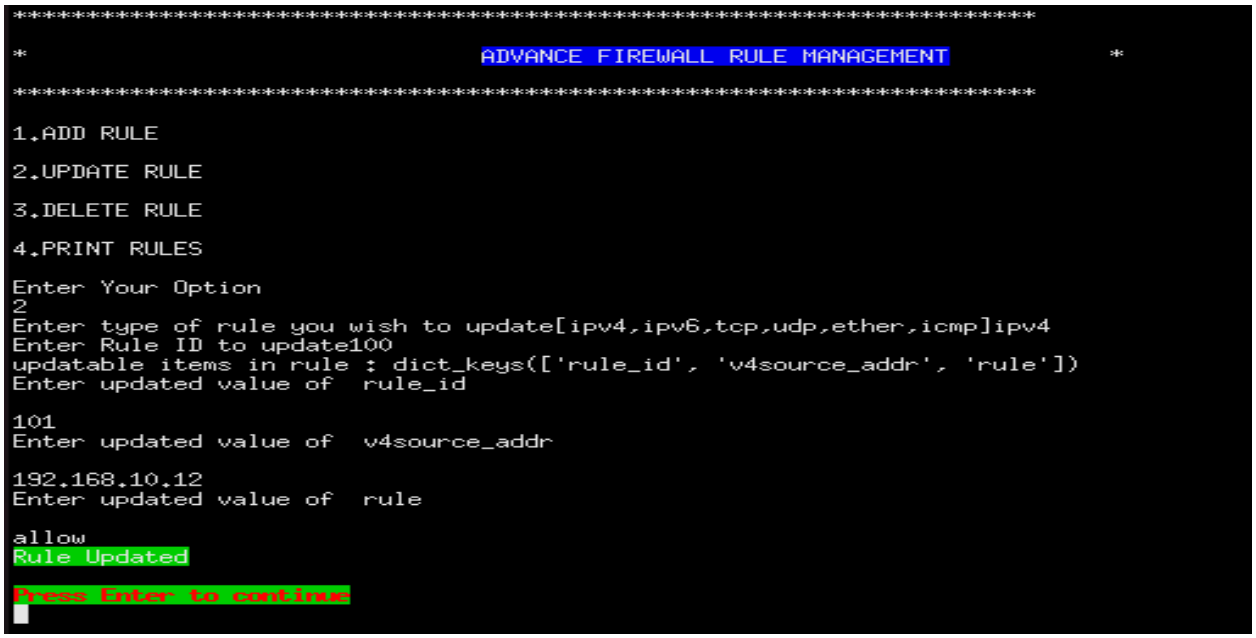


Fig b : Updating Rules

Fig c : Deleting Rules

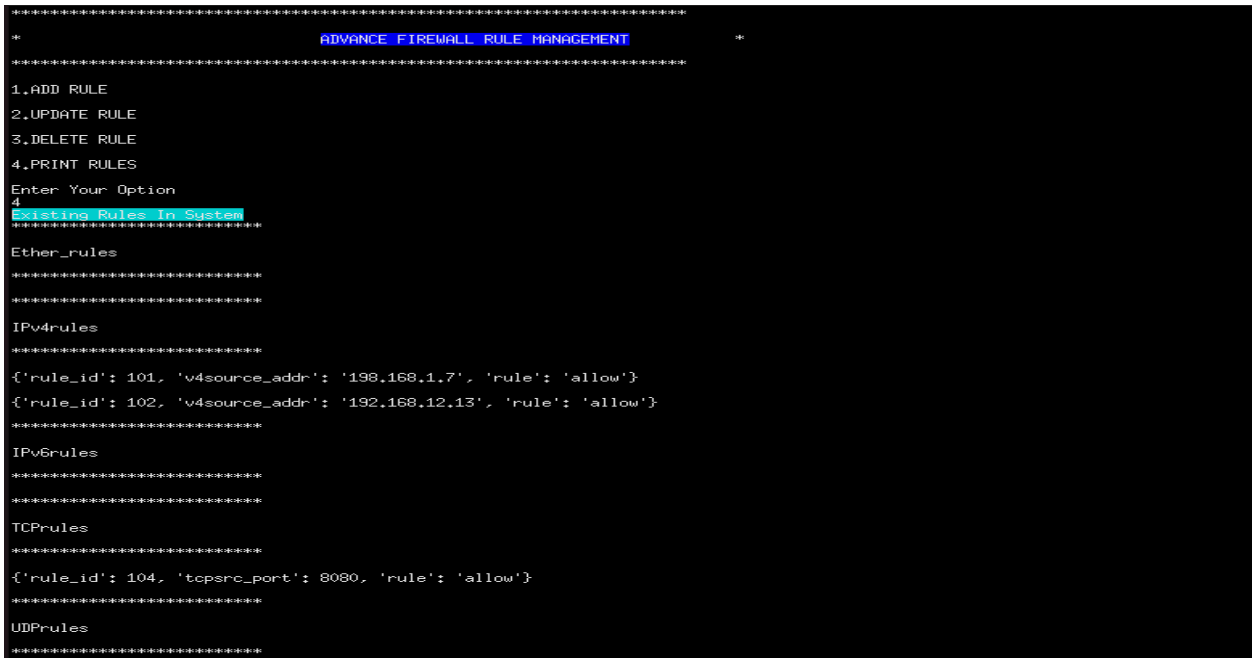


Fig d : Printing Existing Rules in System

Fig e : Saving Rules

Fig f : Loading Rules

Deliverables enclosed:

1. firewall.py - Complete source code for firewall including the simple firewall, advanced firewall for tasks 1, 2 , 3 and 4.

2. README.txt - The readme file enlisting the commands to run the program.

3. report.pdf - Assignment and program documentation report

PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes,and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.

Name: Tahir Ahmed Shaik, Pratik M. Lahase, Jaykishan Pipaliya
Date: 01/05/2021
Signature: Tahir, Pratik, Jaykishan