Volume 3, Issue 12, December 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Distributed Denial of Service attacks and their Allevation using Iptables

Ravender Singh Dahiya*
Engineer (R&D Cell)
Samsung Research Institute, Noida, India

Ritu Dahiya
CSE, Department
GGGI, Dinarpur, Ambala, India

Abstract*— *The Internet consists of hundreds of millions of computers distributed all around the world. Millions of people use the Internet daily, taking full advantage of the available services at both personal and professional levels. The interconnectivity among computers on which the World Wide Web relies, however, renders its nodes an easy target for malicious users who at-tempt to exhaust their resources and launch Denial-of-Service (DoS) attacks against them. Denial-of-Service (DoS) is a network security problem that poses a se-rious challenge to trust-worthiness of services deployed on the servers. The aim of DoS attacks is to make services unavailable to legitimate users by flooding the victim with legitimate like requests and current net-work architectures allow easy-to-launch, hard-to-stop DoS attacks. It is also possible that a lot of malicious hosts coordinate to ood the victim with a abundance of attack packets, so that the attack takes place simultaneously from multiple points. This type of attack is called a Distributed DoS, or DDoS attack. We describe methods and techniques used in denial of service attacks, and we list possible defenses using iptables. In our study, we emulate a denial of service attack using a tool named Hping2. We examine how this tool works and carried out Denial of service attack on a web server and using iptables we prevented the server from getting exposed to the attack without vitiating the server resources to a remarkable extent.

Keywords*— *DoS attack, DDoS attack, ip-chains, ip-tables, Hping

I. Introduction

We studied distributed denial of service attacks in the Internet. Even though denial of service attacks have existed for some time, their recent distributed formats have made these attacks more difficult to prevent. In this paper we firrst summarize the methods involved in denial of service attacks, list possible defences using iptables. We used a tool named Hping2 to study denial of service attacks. Our emulation study examines how various iptables rules may alleviate the problem of denying bandwidth to legitimate users during the denial of service attack. Finally, we use emulation results to recommend certain more sophisticated rules that may protect users in cases of distributed denial of service attacks.

A. Characteristics of Distributed Denial of Service Attacks

DOS Attack: Information security aims to safe-guard confidentiality, integrity and availability of information and information systems. A Denial of Service (DoS) attack is a type of attack focused on disrupting availability. Such an attack can take many shapes, ranging from an attack on the physical IT environment, to the overloading of network connection capacity, or through exploiting application weaknesses[10]. One hundred percent availability of systems and networks is widely accepted to be unattainable, regardless of diligence or the amount of resources allocated to securing systems against attack. Internet facing and other networked infrastructure components are at risk of DoS for two primary reasons: 1. Resources such as bandwidth, processing power, and storage capacities are not unlimited and so DoS attacks target these resources in order to disrupt systems and networks. 2. Internet security is highly interdependent and the weakest link in the chain may be controlled by someone else thus taking away the ability to be self reliant.

DDoS Attack: Nowadays DoS attacks have been superseded by Distributed Denial of Service (DDoS) attacks, where attackers do not use a single host for their attacks but a cluster of several dozens or even hundreds of computers to do a coordinated strike. A Denial of Service (DoS) attack is an attack with the purpose of preventing legitimate users from using a specified network resource such as a website, web service, or computer system. A Distributed Denial of Service (DDoS) attack is a coordinated attack on the availability of services of a given target system or network that is launched indirectly through many compromised computing systems (zombies) [1]. Typically a DDoS master program is installed on one computer using a stolen account. The master program, at a designated time, then communicates to any number of "agent" programs, installed on computers anywhere on the Internet. The agents, when they receive the command, initiate the attack. Using client/server technology, the master program can initiate hundreds or even thousands of agent programs within seconds. The figure below explains DDoS attack mechanism using client/server technology. Master controls the zombies making the attack distributed. Zombies are used by master to attack the victim and these zombies are unknowingly participating in the attack on the victim.

II. METHODS OF DENIAL OF SERVICE ATTACKS

We described below some widely known basic denial of service attack methods that are employed by the attack daemons. Smurf-attack involves an attacker sending a large amount of Internet Control Message Protocol (ICMP) echo traffic to a set of Internet Protocol (IP) broadcast addresses. The ICMP echo packets are specified with a source address of the target victim (spoofed address) [2]. Most hosts on an IP network will accept ICMP echo requests [3] and reply to them with an echo reply to the source address, in this case, the target victim. This multiplies the traffic by the number of responding hosts. On a broadcast network, there could potentially be hundreds of machines to reply to each ICMP packet. The process of using a network to elicit many responses to a single packet has been labelled as an amplifier [7].

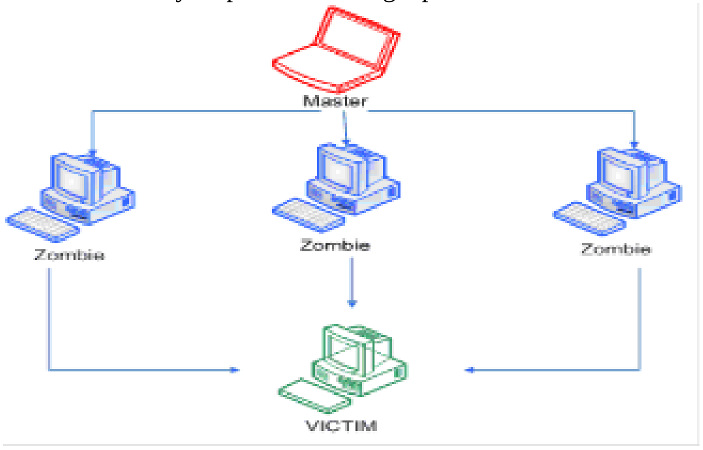


Fig.1: DDoS using Client/Server technology [4].

There are two parties who are hurt by this type of attack: the intermediate broadcast devices (amplifiers) and the spoofed source address target (the victim). The victim is the target of a large amount of traffic that the amplifiers generate. This attack has the potential to overload an entire network. SYN Flood attack is also known as the Transmission Control Protocol (TCP) SYN attack, and is based on exploiting the standard TCP three-way handshake. The TCP three-way handshake requires a three-packet exchange to be performed before a client can officially use the service. A server, upon receiving an initial SYN (synchronize/start) request from a client, sends back a SYN/ACK (synchronize/acknowledge) packet and waits for the client to send the final ACK (acknowledge). However, it is possible to send a barrage of initial SYNs without sending the corresponding ACKs, essentially leaving the server waiting for the non-existent ACKs [5]. Considering that the server only has a limited buffer queue for new connections, SYN Flood results in the server being unable to process other incoming connections as the queue gets overloaded [6]. UDP Flood attack is based on UDP echo and character generator services provided by most computers on a network. The attacker uses forged UDP packets to connect the echo service on one machine to the character generator (chargen) service on another machine. The result is that the two services consume all available network bandwidth between the machines as they exchange characters between themselves. A variation of this attack called ICMP Flood, floods a machine with ICMP packets instead of UDP packets.

III. **HPING**

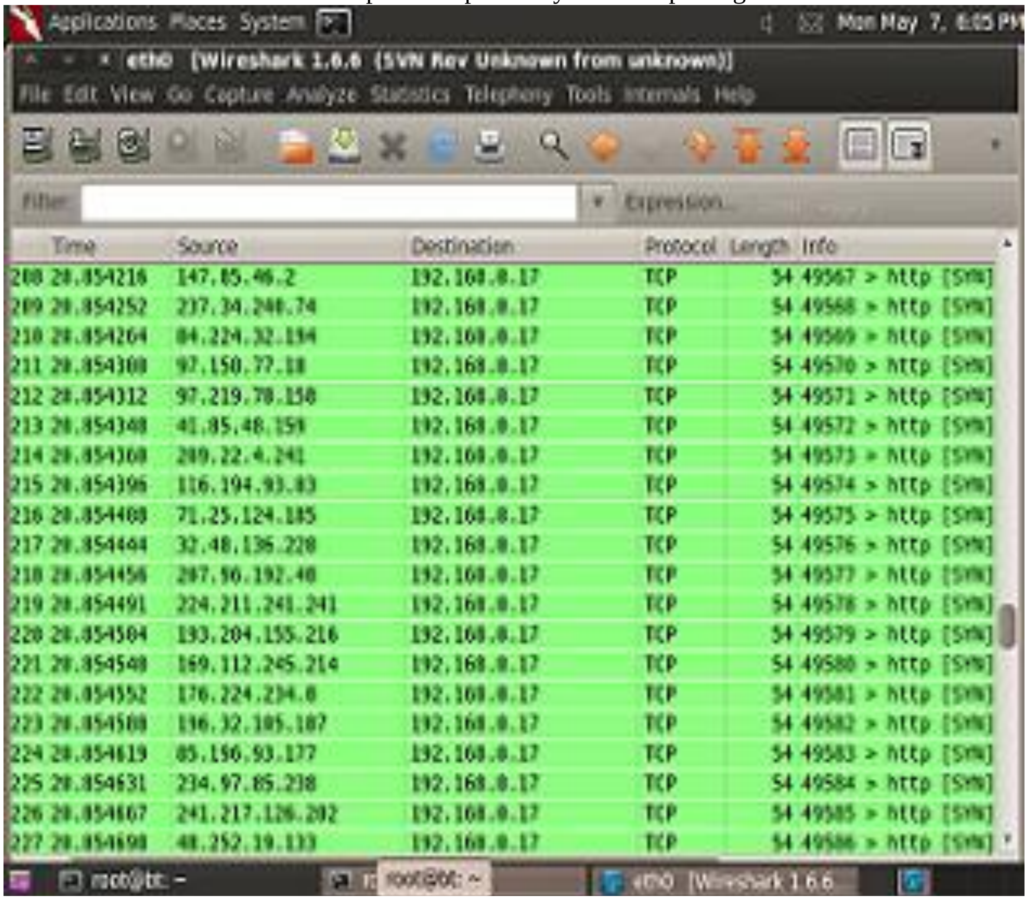
We emulate Denial of Service attack by using Network security tool Hping. Hping is a free packet generator and analyzer for the TCP/IP protocol. Hping is one of the de-facto tools for security auditing and testing of firewalls and networks, and was used to exploit the Idle Scan scanning technique now implemented in the Nmap port scanner. Like most tools used in computer security, hping is useful to security experts, but there are a lot of applications related to network testing and system administration. Hping is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired to the ping(8) unix command, but hping isn't only able to send ICMP echo requests. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features. While hping was mainly used as a security tool in the past, it can be used in many ways by people that don't care about security to test networks and hosts. A subset of the stuff you can do using hping [8]: i) Firewall testing. ii) Advanced port scanning. iii) Network testing, using different protocols, TOS, fragmentation. iv) Manual path MTU discovery. v) Advanced traceroute, under all the supported protocols. vi) Remote OS fingerprinting. vii) Remote uptime guessing. viii)TCP/IP stacks auditing. ix) hping can also be useful to students that are learning TCP/IP. Hping works on the following unix-like systems: Linux, FreeBSD,NetBSD, OpenBSD, Solaris, MacOs X, Windows. The versions of Hping Hping2 and Hping3 are now available. We have used Hping2 in our study.

IV. **IPTABLES**

Originally, the most popular firewall/NAT package running on Linux was ipchains, but it had a number of shortcomings. To rectify this, the Netfilter organization decided to create a new product called iptables, giving it such improvements as [9]: i) Better integration with the Linux kernel with the capability of loading iptables-specific kernel modules designed for improved speed and reliability. ii) Stateful packet inspection. This means that the firewall keeps track of each connection passing through it and in certain cases will view the contents of data flows in an attempt to anticipate the next action of certain protocols. This is an important feature in the support of active FTP and DNS, as well as many other network services. iii)Filtering packets based on a MAC address and the values of the flags in the TCP header. This is helpful in preventing attacks using malformed packets and in restricting access from locally attached servers to other networks in spite of their IP addresses. iv) System logging that provides the option of adjusting the level of detail of the reporting. v) Better network address translation. vi) Support for transparent integration with such Web proxy programs as Squid. vii) A rate limiting feature that helps iptables block some types of denial of service (DoS) attacks. Considered a faster and more secure alternative to ipchains, iptables has become the default firewall package installed under RedHat and Fedora Linux.

V. EMULATION SCENARIO AND RESULTS

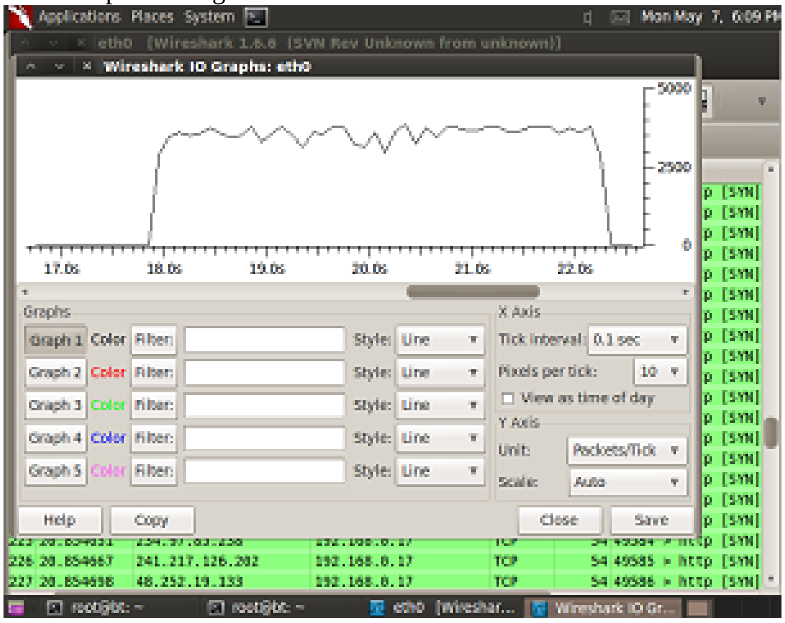
Using the tool Hping2, we were able to perform SYN flood and ICMP flood DDoS attacks. Hping2 used random source addresses and the attack was performed successfully. Hping2 command for SYN flood attack: hping2 -i u1 -S -p 80 - -rand-source 192.168.0.17 Here are also the snapshots captured by traffic capturing tool wireshark.


Applications Places System
Mon May 7, 6:05 PM
eth0 [Wireshark 1.6.6 (SVN Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression...
root@bt: ~
eth0 [Wireshark 1.6.6

```
    Time       Source            Destination     Protocol Length Info
208 20.854216  147.85.46.2       192.168.0.17    TCP      54 49567 > http [SYN]
209 20.854252  237.34.240.74     192.168.0.17    TCP      54 49568 > http [SYN]
210 20.854264  84.224.32.194     192.168.0.17    TCP      54 49569 > http [SYN]
211 20.854308  97.150.77.18      192.168.0.17    TCP      54 49570 > http [SYN]
212 20.854312  97.219.78.158     192.168.0.17    TCP      54 49571 > http [SYN]
213 20.854348  41.85.48.159      192.168.0.17    TCP      54 49572 > http [SYN]
214 20.854368  209.22.4.241      192.168.0.17    TCP      54 49573 > http [SYN]
215 20.854396  116.194.93.83     192.168.0.17    TCP      54 49574 > http [SYN]
216 20.854408  71.25.124.185     192.168.0.17    TCP      54 49575 > http [SYN]
217 20.854444  32.48.136.228     192.168.0.17    TCP      54 49576 > http [SYN]
218 20.854456  207.96.192.48     192.168.0.17    TCP      54 49577 > http [SYN]
219 20.854491  224.211.241.241   192.168.0.17    TCP      54 49578 > http [SYN]
220 20.854504  193.204.155.216   192.168.0.17    TCP      54 49579 > http [SYN]
221 20.854548  169.112.245.214   192.168.0.17    TCP      54 49580 > http [SYN]
222 20.854552  176.224.234.8     192.168.0.17    TCP      54 49581 > http [SYN]
223 20.854588  196.32.105.187    192.168.0.17    TCP      54 49582 > http [SYN]
224 20.854619  85.156.93.177     192.168.0.17    TCP      54 49583 > http [SYN]
225 20.854631  234.97.85.238     192.168.0.17    TCP      54 49584 > http [SYN]
226 20.854667  241.217.126.202   192.168.0.17    TCP      54 49585 > http [SYN]
227 20.854698  48.252.19.133     192.168.0.17    TCP      54 49586 > http [SYN]
```

Fig. 2: SYN Flood attack on 192.168.0.17 (Random source addresses)

Given below is the IO graph showing the duration for which the attack was performed. As shown in the figure 3 that the attack was performed for less than 5 secs and hping2 sent more than 1 million packets in this small interval of time. So hping is a very good emulator for performing such an attack.



Tick interval: 0.1 sec
Pixels per tick: 10
View as time of day
Unit: Packets/Tick
Scale: Auto

```
226 20.854667  241.217.126.202  192.168.0.17  TCP  54 49585 > http [SYN]
227 20.854698  48.252.19.133    192.168.0.17  TCP  54 49586 > http [SYN]
```

Fig.3: IO graph of SYN flood(attack was performed from 17.8 sec to 22.3 sec)

Given below in figure 4 is the flow graph for the SYN flood performed.

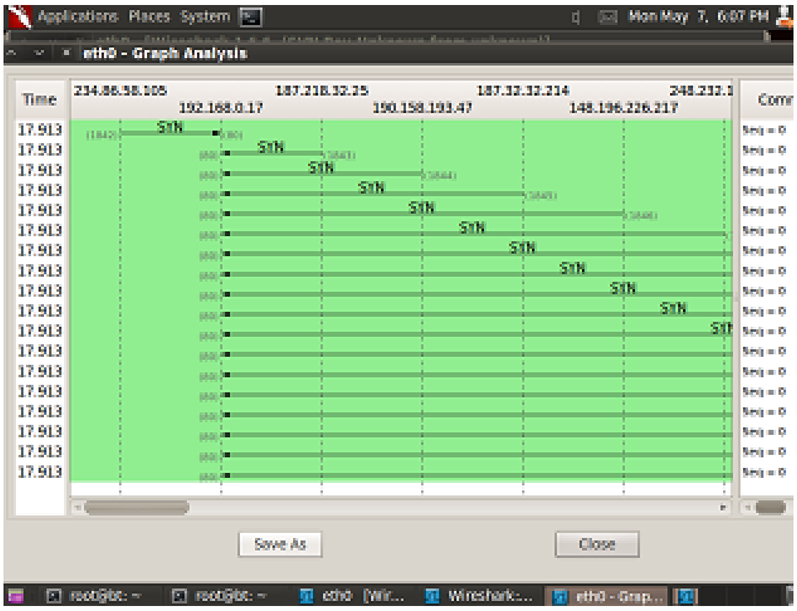


Fig.4: Flow graph of SYN flood

Similarly, ICMP flood attack was performed : hping2 -i u1 -1 - -rand-source 192.168.0.17 Below in figure 5 is the traffic captured during the attack. Mitigation through iptables rules: After performing the attacks some of the iptables rules were applied to remarkably mitigate these attacks.

First rule : Limit NEW traffic on port 80: iptables -A INPUT -p tcp - -dport 80 -m state - -state NEW -m limit - -limit 50/minute - -limit-burst 200 -j
ACCEPT

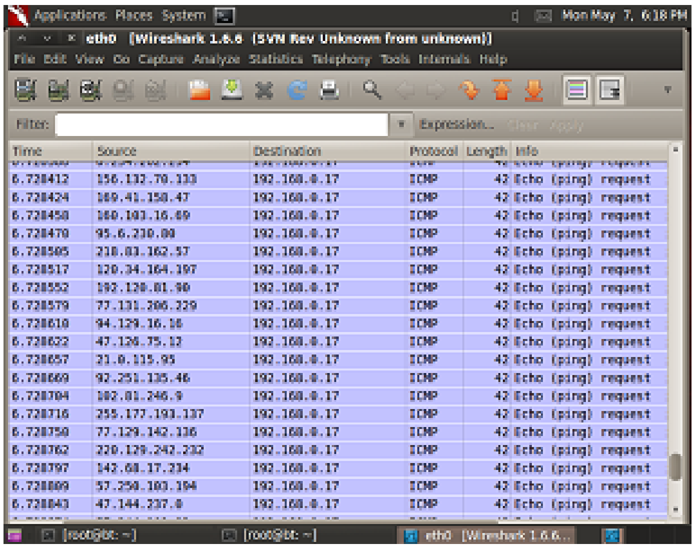
Lets break that rule down into intelligible chunks**.**

-p tcp - -dport 80 : Specifies traffic on port 80.

-m state NEW : This rule applies to NEW connection.

-m limit - -limit 50/minute - -limit-burst 200 -j ACCEPT : This is the essence of preventing DOS.- -limit-burst is a bit confusing, but in a nutshell 200 new connections (packets really) are allowed before the limit of 50 NEW connections (packets) per minute is applied. Similarly, limit can be imposed on ICMP requests.

Second rule: Limit established traffic: iptables -A INPUT -m state - -state RELATED,ESTABLISHED -m limit - -limit 50/second -
-limit-burst 50 -j ACCEPT.


Applications Places System
Mon May 7, 6:18 PM
eth0 [Wireshark 1.6.6 (SVN Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
[root@bt: ~]
[root@bt: ~]
eth0 [Wireshark 1.6.6...

```
Time        Source            Destination    Protocol Length Info
6.728412    156.132.70.133    192.168.0.17   ICMP     42 Echo (ping) request
6.728424    169.41.158.47     192.168.0.17   ICMP     42 Echo (ping) request
6.728458    160.103.16.69     192.168.0.17   ICMP     42 Echo (ping) request
6.728470    95.6.230.80       192.168.0.17   ICMP     42 Echo (ping) request
6.728505    218.83.162.57     192.168.0.17   ICMP     42 Echo (ping) request
6.728517    120.34.164.197    192.168.0.17   ICMP     42 Echo (ping) request
6.728552    192.120.81.90     192.168.0.17   ICMP     42 Echo (ping) request
6.728579    77.131.206.229    192.168.0.17   ICMP     42 Echo (ping) request
6.728610    94.129.16.18      192.168.0.17   ICMP     42 Echo (ping) request
6.728622    47.126.75.12      192.168.0.17   ICMP     42 Echo (ping) request
6.728657    21.0.115.95       192.168.0.17   ICMP     42 Echo (ping) request
6.728669    92.251.135.46     192.168.0.17   ICMP     42 Echo (ping) request
6.728704    182.81.246.9      192.168.0.17   ICMP     42 Echo (ping) request
6.728716    255.177.193.137   192.168.0.17   ICMP     42 Echo (ping) request
6.728750    77.129.142.136    192.168.0.17   ICMP     42 Echo (ping) request
6.728762    220.129.242.232   192.168.0.17   ICMP     42 Echo (ping) request
6.728797    142.68.17.234     192.168.0.17   ICMP     42 Echo (ping) request
6.728809    57.250.103.194    192.168.0.17   ICMP     42 Echo (ping) request
6.728843    47.144.237.0      192.168.0.17   ICMP     42 Echo (ping) request
```

Fig.5: ICMP flood on 192.168.0.17(Random source addresses)

This rule applies to RELATED and ESTABLISHED all traffic on all ports, but is very liberal (and thus should not affect traffic on port 22 or DNS). If you understood the 1st rule, you should understand this one as well. In summary, 50 ESTABLISHED (and/or RELATED) connections (packets really) are allowed before the limit of 50 ESTABLISHED (and/or RELATED) connections (packets) per second is applied.

Third rule: Log the input traffic:

iptables -A INPUT -j LOG {log-level 4.

Similarly we can apply these rules on output chain also.We can also log these ips as well(experiment was performed on Redhat based, Centos). First of all set your Log Daemon to log the IPTABLES:

```
vi /etc/syslog.conf
```

Add the following line at the end of the file :

```
kern.warning/var/log/iptables.log
touch/var/log/iptables.log
```

Restart the System Log Service :

```
/etc/init.d/syslog restart (On Redhat based,Centos)
```

The next stage is to add these logged ips to TCP-WRAPPER (/etc/hosts.deny)

```
/bin/cat /var/log/iptables.log |awk 'fprint $9g' | cut -f2 -d "=" >>/root/rd.txt
```

The above line will grep the SOURCE ip from the log and append to rd.txt.Next Run this command as a frequent interval with the help of CRON

```
vi/etc/crontab
*/1 * * * * root /bin/cat /var/log/iptables.log |awk 'fprint $9g' | cut -f2 -d "=" >>/root/rd.txt
```

Here the script will run in every minutes. The file will be grow up rapidly to heavy size if your server have heavy traffic. So clean up the file in frequent intervals. Better setup another CRON for it. Next to add these IPs in the hosts.deny file.

vi/etc/hosts.deny
SSHD:/root/rd.txt

Now the ips can be denied by the system.

VI. CONCLUSION

In this paper, we discussed distributed denial of service attacks on the Internet. We described how distributed attacks are conducted, we reviewed some well known distributed denial of service techniques, and discussed a defence mechanism that could be employed using iptables on hosts. We used network emulation tool hping2 and traffic capturing tool wireshark to examine the various DDoS attacks. We also used iptables and found out some rules that were helpful in alleviating the distributed denial of service attacks and in providing desired service to-the users. It was found that although the iptables rules we used seems very open, it does put some limits on our connections. So one should be very wise while adopting these rules and the second rule we discussed one should try to use the first rule with or without the second one and accordingly decide to go for the second rule. We applied the rules on the input chain of iptables, similarly they can be applied on the outbound traffic that is the output chain of the iptables. Lastly we found out that exploring log files early detection is possible and we were also able to deny hosts and provided our system with prevention and mitigation. In summary, our emulation and its results indicated how DDoS attack occurs, how the victim fell prey to the attacker and also what we can do using iptables and provide the desired solution in protecting users in cases of distributed denial of service attacks.

REFERENCES

[1] David Karig and Ruby Lee, \Remote Denial of Service Attacks and Countermeasures," Princeton University Department of Electrical Engineering Technical Report CEL2001-002, Oct 2001.
[2] Daemon9, Infinity, and Routte, \IP-spoofing demystified: trustrelationship exploitation,"Phruck Mug., June 1996,http://www.fc.net/phrack/files/p48/p48-14.html.
[3] S. Bellovin, Ed., \The ICMP traceback message,"Network Working Group Intemet Draft, Mar.2000, http://www.research.att.com/-smb/papers/draft-bellovin-itrace-00.txt.
[4] Lincoln Stein & John Stewart, \WWW Security FAQ: Securing Against Denial of Service," www.w3.org/Security/Faq/wwwsf6.html,Feb 02, 2003.
[5] S. Bellovin, \Security problems in the TCP/IP protocol suite," Comput. Commun. Rev., vol. 19,no. 2, pp. 32-48, Apr. 1989.
[6] Cisco Systems, Inc., \De_ning strategies to protect against TCP SYN denial of service attacks,"July 1999,http://www.cisco.com/warp/public/707/4.html.
[7] C. A. Huegen, \The latest in denial of service attacks: 'Smurfing' description and information to minimize effects," Feb. 2000,http://users.quadrunner.coin/chuegen/smurf.cgi.
[8] "HPING,"http://Hping-ActiveNetworkSecurityTool.htm.
[9] "QuickHOWTO,"http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO [4.1.12 PM 09:29:32].
[10] Cisco Systems, \Distributed Denial of Service Attacks," The Internet Protocol Journal, Volume 7, Number 4, 2005.