Packet Filtering using IP Tables in Linux

Bhisham Sharma[1], Karan Bajaj[2]

[1] Computer Science and Engineering Department, Chitkara University
Baddi, Himachal Pradesh, India

[2] Computer Science and Engineering Department, Chitkara University
Baddi, Himachal Pradesh, India

Abstract

Firewalls are core elements in network security. However, managing firewall rules, especially for enterprise networks, has become complex and error-prone. Firewall filtering rules have to be carefully written and organized in order to correctly implement the security policy. In addition, inserting or modifying a filtering rule requires thorough analysis of the relationship between the rules in order to determine the proper order.

In this paper work has been done on creating the virtual network environment using Microsoft virtual PC(SP1) and Capturing and analyzing of network packets using the most popular open source network protocol analyzer Wireshark and on t he basis of analyzing the packet work has been done on writing the script to block/allow the network traffic using IPtables and after blocking traffic further capturing and analyzing of packets using Wireshark.

Keywords:*Firewall, Wireshark, IPtables,Linux, HTTP*

1. Introduction

With the global Internet connection, network security has gained significant attention in the research and industrial communities. Due to the increasing threat of network attacks, firewalls have become important integrated elements not only in enterprise networks but also in small-size and home networks. Firewalls have been the frontier defense for secure networks against attacks and unauthorized traffic by filtering out unwanted network traffic coming into or going from the secured network. The filtering decision is taken according to a set of ordered filtering rules written based on predefined security policy requirements. Different organizations don't have equal needs from the point of allowed traffic. What is allowed in one organization doesn't have to be allowed in another. According to different needs, different firewall policies must be created. Firewall policies don't only differentiate between organizations – they also change over time. Furthermore, different administrators would most likely create different rule lists even for the same policy. For each type of network traffic, there are one or more different rules. Every network packet, which arrives at firewall, must be ch ecked against defined r ules until first matching r ule is found. The packet w ill b e th en allowed or banned access to the network, depending on the action specified in the matching rule.

Packet filtering allows you to explicitly restrict or allow packets by machine, port, or machine and port. For instance, you can restrict all packets destined for port 80 (WWW) on all machines on your LAN except machine X and Y.

Packet filtering is most commonly used as a first line of defense against attacks from machines outside your LAN. Since most routing devices have built-in filtering capabilities, packet filtering has become a common and inexpensive method of security.

IPtables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

Linux IPtables is currently the default firewall package that comes from RedHat, CentOS, UBUNTU and Fedora, right after ipchains dominated them long time ago. IPtables supports different types of filters. To name a few, IPtables can do filters and firewall rules by usernames, by group IDs and user profiles, by source and destination ports, by source host and destination hosts, by URLs, by IP addresses, by packet ID flags, by protocols, and a lot more including filtering by MAC address.

2. IP Tables Background

IPtables provides comparatively higher speed and reliability than the other firewall tools. Being a Linux product, its integration with the OS is also more robust and reliable. It keeps a stateful track of each connection passing through it and tries to anticipate feature actions. Its capacity of filtering packets on the basis of MAC address makes it a formidable security system. It can filter out attacks using malformed packets and can restrict access from locally attached servers to other networks in spite of

their valid IP addresses. Network address translation (NAT) with masquerading capability of IPtables helps it to hide internal network IP sub networks behind one or a small poll of external IP addresses. NAT and masquerading enables the firewall system to open and close ports on the gateway. The rate-limiting feature of IPtables can block attacks even from some types of DoS (denial of service) attacks.

2.1. Firewall rules

Different firewalls usually provide different rule logic with different parameters. But some basic elements are common to all. They all allow an action to be defined allowing or banning specific network traffic. Also, all of them allow checking for most important elements in packets like IP addresses, ports and protocol. IP Tables is a command line tool for writing and executing of Firewall rules. One of the most important functionalities of IP Tables firewall is stateful inspection. S.I. automatically opens only the ports necessary for internal packets to access the Internet. It only allows transfer of packets which are defined in firewall rules and which are part of established connections.

2.2. Firewall chains

IP Tables group rules in chains. Different network packets are processed by different chains:

•Incoming traffic – packets for firewall (INPUT chain).
•Forwarding traffic – incoming packets for another machine (FORWARD chain).
•Outgoing traffic – packets generated by firewall (OUTPUT chain).

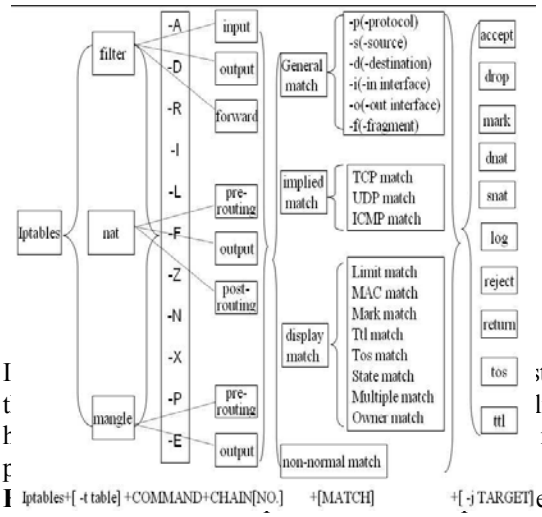
2.3. Rule parameters

Each rule identifies specific type of network traffic. In order to enable this identification parameter for identification of specific network packets must be set for each rule.

Different type of parameters:

•IP addresses – it can be destination or source IP address; also, it can be written as a single IP, network IP or IP range.
•Ports - it can be destination or source port; also, it can be written as a single port, port range or port array.
•Protocol – it can be referred to TCP, UDP, and ICMP or all together.
•Interface – it can be incoming or outgoing interface.
•TTL (Time To Live) field residing in the IP headers.
•ToS (Type of Service) field residing in the IP headers.
•Length of packet.
•MAC source address.
•Syn flag – identification of new connection.
•ICMP type.

2.4 Firewall using IP Tables



I t table is t ltering. It h rules are p ed by the firewall.

Input chain: Filters packets destined for the firewall.

Output chain: Filters packets originating from the firewall.

The second table is the **Nat** queue which is responsible for network address translation. It has two built-in chains; these are:

Pre-routing chain: NATs packets when the destination address of the packet needs to be changed.

Post-routing chain: NATs packets when the source address of the packet needs to be changed.

The third is the **mangle** table which is responsible for the alteration of quality of service bits in the TCP header.

It is necessary to specify the table and the chain for each firewall rule you create. There is an exception: Most rules are related to filtering, so IPtables assumes that any chain that's defined without an associated table will be a part of the filter table. The filter table is therefore the default.

2.5 Targets and jumps in IP Tables

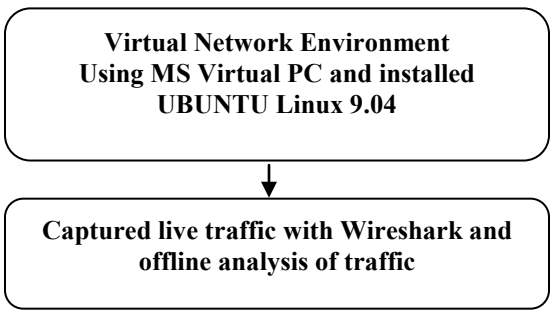
Each firewall rule inspects each IP packet and then tries to identify it as the target of some sort of operation. Once a target is identified, the packet needs to jump over to it for further processing.

Table1: Descriptions of the most commonly used targets

Target	*Description*

ACCEPT	IPtables stops further processing. The packet is handed over to the end application or the operating system for processing.
DROP	IPtables stops further processing. The packet is blocked.
LOG	The packet information is sent to the syslog daemon for logging. IPtables continues processing with the next rule in the table. As you can't log and drop at the same time, it is common to have two similar rules in sequence. The first will log the packet, the second will drop it.
REJECT	Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked.
DNAT	Used to do de stination network address translation. i.e. rewriting the destination IP address of the packet.
SNAT	Used to do s ource network address translation rewriting the source IP address of the packet. The source IP address is user defined.
MASQUERADE	Used to do Source Network Address Translation. By default the source IP address is the same as that used by the firewall's interface.

3. Implementation Setup



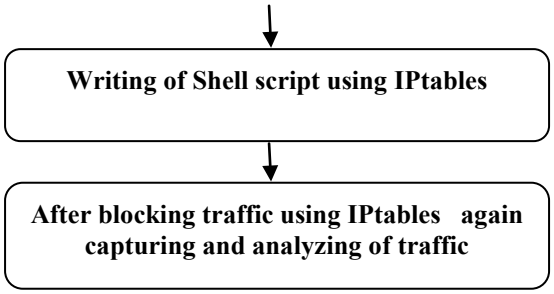


Fig.2 Implementation setup

3.1 Steps performed during Configuration and Implementation:

Step1: To establish a segregate network using virtualization. Microsoft Virtual PC SP1 is used to establish a s egregate network and UBUNTU 9.04 operating system is installed on it.

Step2: Configuring the Wireshark under root privileges Applications > Add/Remove applications > All open source application > wireshark.

Step3: Operating Wireshark (as root) and capturing network packets on eth0 interface.

Step4: IPtables was downloaded under the root privileges from www.netfilter.org and installed by implementing. /configure, make, and make install command.

Step5: Writing of IPtables script to deny/allow network traffic.

Step6: After blocking traffic using IPtables again capturing and analyzing of network data packets using Wireshark.

The following are the various tasks and milestones completed along with the results. All the results are checked on machine Intel® Core™ 2 Duo CPU T8100 @ 2.10 GHz with frequency 778 MHz along with 2 GB RAM space. The Operating System used was Ubuntu-9.04 Linux operating system which was installed using virtualization software.

In this research paper there are 5 different modules discussed below:

Module1. This module simply blocked HTTP traffic on TCP port no.80 on the Basis of specific URL's, Here we blocked social networking website that need to be banned in educational institution like www.facebook.com, www.myspace.com, and www.Bebo.com etc.

Facebook, Myspace, Bebo and Buzz allow students to contact people over the internet. They make 'friends' with people that they can't be sure are genuine and that can lead to problems. MySpace, Facebook and Bebo also encourage people (including children) to put photographs and other details about themselves into the system. Facebook is now

the most popular social networking site worldwide, with Myspace second.

Many schools and public libraries in the US and the UK have begun to restrict access to Facebook and Myspace because it has become "such a haven for student gossip and malicious comments".

#Blocking of www.Facebook.com accessed through port no 80

iptables -A OUTPUT -p tcp --dport 80 -d www.facebook.com -j DROP

#Blocking of www.Myspace.com accessed through port no 80

iptables -A OUTPUT -p tcp --dport 80 -d www.myspace.com -j DROP

#Blocking of www.bebo.com accessed through port no 80

iptables -A OUTPUT -p tcp --dport 80 -d www.bebo.com -j DROP

#Blocking of www.orkut.com accessed through port no 80

iptables -A OUTPUT -p tcp --dport 80 -d www.orkut.com -j DROP

Module2. Blocking of Spam mails coming from specific IP address to secure the network so that unauthorized user was unable to access the resources of the system. SPAM is one giant pain in the neck. There are usually fifty other things you would prefer to be doing than finding new ways to defeat spam. Here, we create new chain SPAMLIST in which all the rules are appended through which we block spam mails coming from bad IP's.

Blocked.ips is a file in which list of bad IP's is mentioned.

 # IPTables IP/subnet block script

IPT=/sbin/iptables
SPAMLIST="spamlist"
SPAMDROPMSG="SPAM LIST DROP"
BADIPS=$(grep-Ev"^#|^$" root/blocked.ips")

 # create a new iptables list

$IPT -N $SPAMLIST
for ipblock in $BADIPS
do
$IPT -A $SPAMLIST -s $ipblock -j LOG --log-prefix "$SPAMDROPMSG"
$IPT -A $SPAMLIST -s $ipblock -j DROP
done
$IPT -I INPUT -j $SPAMLIST
$IPT -I OUTPUT -j $SPAMLIST
$IPT -I FORWARD -j $SPAMLIST

Blocked.ips

192.168.1.0/24
Vsnloard.vsnl.net.in
202.54.1.2
SPAM
202.5.1.2
Personal information stealers (Big Brother):
doubleclick.net
216.73.93.8
The SPAMMERs never stop to find new ways to SPAM
Messageaway.com PopUp SPAMMER
74.86.203.162
sofcom.com
98.124.199.1
mp3za.ru
203.110.240.22
free4all.com
64.95.64.198

Module3. Blocking of ICMP packet so that unauthorized user is unable to ping the system. Several Web sites block ICMP traffic due to DDoS attacks. A common example of "bad" ICMP is to allow any ICMP traffic from unknown sources onto your trusted networks. For example, if you allow ICMP redirects, you leave your Internet hosts susceptible to having their traffic inadvertently routed to the wrong location. This could result in a DoS is the best case (because the traffic never makes it to the hosts that are requesting data). To address this, it is generally a good idea to block ICMP traffic, in particular between authorized and unauthorized networks. ICMP messages themselves are also susceptible to manipulation. Perhaps the most well known of this kind of manipulation is known as the "ping of death," which transmitted a message that exceeded the 65,535-byte limit of the IP protocol, which would cause many target hosts to crash, resulting in a DoS.

 # set the default policies

iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
#

#In the mentioned method best thing is to drop the ICMP packets, by doing this we are not giving any clue to hacker whether the system is alive or not. Where as if we do reject definitely hacker will come to know that ICMP packets are blocked and the system is live.

#iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

Module4. In this module we simply stop incoming/outgoing SMTP traffic to protect the system from the various types of attacks like Phishing, Hoaxes, and Trojans. Phishing is a scam, where a stranger sends an email which appears as if it is from a trusted organization to a normal user to get his personal and financial information. For example, when you receive a mail from a bank to update your personal bank account information and when you click on the link to update the information a separate window opens which looks like an original bank site, where it asks for account information, password and other details. When you enter the information and press enter it will go to the hands of strangers and not to the bank site. Hoax is an attempt to make the person believe

something which is false as true. It also defined as an attempt to deliberately spread fear, doubt among the users.

 # stopping of reading and writing of Emails

```
iptables  -A INPUT -p tcp  --dport 25  -j DROP
iptables -A INPUT -p udp  --dport 25  -j DROP
iptables -A INPUT -p udp  --dport 110 -j DROP
iptables -A OUTPUT -p tcp  --sport 25  -j DROP
iptables -A OUTPUT -p udp  --sport 25  -j DROP
iptables -A OUTPUT -p udp  --sport 110 -j DROP
```

Module5. In this module we will work on blocking the P2P file sharing traffic. Peer-to-Peer (P2P) applications impede network traffic of businesses, governments, education, and the Internet infrastructure itself. These applications consume vast amounts of network resources, and prevent mission critical applications from accessing the network. In addition, hackers seek to exploit P2P applications to access and attack the large install base, presenting serious security vulnerabilities to systems and networks. These applications also pose a s erious legal issue as users download copyrighted material, placing access providers in a difficult legal situation. As a result, these applications create a l ogistic, security, and legal nightmare for network administrators on high-speed networks. To protect networks from excessive bandwidth consumption and malicious attacks, we will work on writing the IPtables script to control Peer-to-Peer applications and network traffic.

 # Block P2P Traffic

```
iptables -A FORWARD -p tcp -m ipp2p --edk -j DROP
iptables -A FORWARD -p udp -m ipp2p --edk -j DROP
iptables -A FORWARD -p tcp -m ipp2p --dc -j DROP
iptables -A FORWARD -p tcp -m ipp2p --kazaa -j DROP
iptables -A FORWARD -p udp -m ipp2p --kazaa -j DROP
iptables -A FORWARD -p tcp -m ipp2p --gnu -j DROP
iptables -A FORWARD -p udp -m ipp2p --gnu -j DROP
iptables -A FORWARD -p tcp -m ipp2p --bit -j DROP
iptables -A FORWARD -p udp -m ipp2p --bit -j DROP
iptables -A FORWARD -p tcp -m ipp2p --apple -j DROP
iptables -A FORWARD -p tcp -m ipp2p --winmx -j DROP
iptables -A FORWARD -p tcp -m ipp2p --soul -j DROP
iptables -A FORWARD -p tcp -m ipp2p --ares -j DROP
```

4. Conclusions

In this research paper, work has been done on capturing the live traffic using the network protocol analyzer Wireshark and on the basics of analyzed data packets further explored and designed the script using IPtables to allow/deny the network traffic on the basics of the IP address of the computer sending the packets, the IP address of the computer receiving the packets, the type of packet (TCP, UDP, etc.), The port number, and URL's etc. This enables us to protect our system from a wide variety of hazards, including service attacks and hack attempts.

The script discussed here can be used for the purpose of network Security.

- Web traffic sent on HTTP can be analyzed.
- Denying of ICMP, SMTP data packets.
- Configuring of host based packet filtering firewall to deny various type of attacks like spoofing, Stop bad packets, Stop Xmas Tree type scanning, null scanning, syn flood and, ping flood attack etc.
- Deny P2P file sharing traffic.

References

[1]Gunter Schafer, "Network Security Tutorial", May 2003, Anchorage, Alaska.
[2] Network Security policy and objectives, http://publib.boulder.ibm.com/infocentr/iseries/securitypolco.htm
[3]Packtet Filtering Process, http://www.ibm.com/developerworks/linux/library/s-netip/
[4]Packet filtering using IPtables, http://netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-7.html
[5]eSoft, "Modern Network Security: The Migration to Deep Packet Inspection", White Paper, 2006.
[6]David W Chadwick, "Network Firewall Technologies", IS Institute, University of Salford, Salford, M5 4WT, England.
[7]Internet Firewall Tutorial, A White Paper July 2002.
[8]A. Hari, S. Suri, and G. M. Parulkar, "Detecting and resolving packet filter conflicts", In Proc. of IEEE Infocom, pages 1203-1212, 2000.
[9]Christoph Ludwig Schuba, "On the modeling, design, and implementation of firewall technology", Purdue University.
[10]Elizabeth D. Zwicky, Simon Cooper, D Brent Chapman,"Building Internet Firewalls", 2-Ed, Oreilly, 2000.
[11]Ch14 : Linux Firewalls using iptables, http://www.linuxhomenetworking.com/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables/
[12]Oskar Andreasson, "Iptables Tutorial", http://www.frozentux.net/documents/iptables-tutorial/
[13]IPtables HowTo, https://help.ubuntu.com/community/IptablesHowTo
[14]IPtables Home, http://www.netfilter.org/
[15]IPtables Scripting, http://www.linuxdoc.org/
[16]Iptables command, http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/s1-iptables-options.html
[17]IPtables Parameters, http://www.centos.org/docs/2/rhl-rg-en-7.2/s1-iptables-options.html
[18]IPP2P, http://www.ipp2p.org/docu_en.html

First Author: *Bhisham Sharma, M.E(Computer Science & Engineering) from Thapar University, Patiala. He is presently working as Assistant Professor in Department of Computer Science & Engineering at Chitkara University. He has over 2 year of industry and*

teaching experience to his credit. He has developed various software and websites during his job tenure. He has guided various projects at B. Tech. Level. His areas of interest include Design of Computer networks, Computer and Network Security, Integration of Computer Networks and Communication Systems, Data Structure, Artificial Intelligence, Data base management system and HCI. He has attended various workshops and short-term courses in different domains. He has presented several papers in national and international conferences.

Second Author *Karan Bajaj, Btech(Computer Science & Technology) from Himachal Pradesh University. He is presently working as Assistant Lecturer in Department of Computer Science & Engineering at Chitkara University. He has 2 year of teaching experience to his credit. He has attended various workshops and short-term courses in different domains.*