Pratibha Tambewagh et al | IJCSET(www.ijcset.net) | 2019 | Volume 9, 21-24

ISSN:2231-0711

Application Layer Packet Classifier

Pratibha Tambewagh[#1], Asmita Jagtap[#2]

[#1,] *Lecturer, BVIT, Kharghar, Navi Mumbai.*

[#2,] *Lecturer, BVIT, Kharghar,Navi Mumbai.*

Abstract-The use of Application layer packet classifier and optimization of bandwidth towards QoS in Linux using netfilter, iproute2 and layer-7 Filter.As seen in the statistics the huge amount of data flows through the network, so it is necessity to apply packet-filtering rules in order to control the traffic and add firewall rules. Some services are inherently insecure and impossible to secure on individual hosts. Packet filtering tools can help you segment and contain parts of your network to increase security. A packet filtering tools can help you enforce your network security policies by selectively allowing network services. Because a packet filtering tools must examine all inbound/outbound network traffic, it can help you log network activity. We are looking at packet filtering tools like Netfilters and iproute2, who examine the IP packets for filtering and using the queuing disciplines for traffic control.

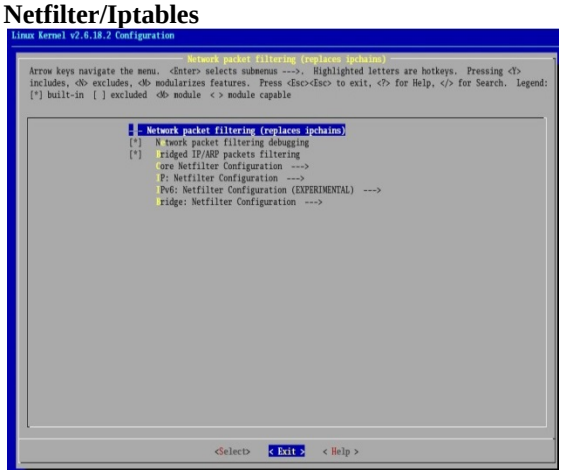
Keywords- Application layer packet, HTTP, FTP

1. INTRODUCTION

The most important things needed to build packet filtering rules using firewalls and Quality of Service (QoS) with Linux are two packages named netfilter and iproute2. While netfilter is a packet filtering framework included in the Linux kernels 2.5 and 2.6 which uses iptables as a frontend, netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack. Iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).Iproute is a package containing a few utilities like bandwidth provisioning called Traffic Control (tc) and ip that allow Linux users to do various method for classifying, prioritizing, sharing, and limiting both inbound and outbound traffic.

2. METHODOLOGY

2.1 Netfilter/Iptables

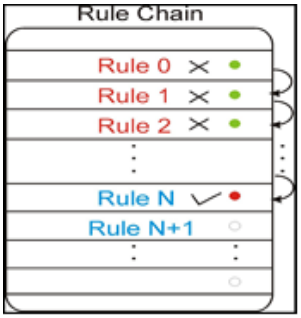


Netfilter is a very important part of the Linux kernel in terms of security, packet mangling, and manipulation. The front end for netfilter is iptables, which "tells" the kernel what the user wants to do with the IP packets arriving into, passing through, or leaving the Linux box, it's an main router to route all the packet. The most used features of netfilter are packet filtering and network address translation, but there are a lot of other things that we can do with netfilter, such as packet mangling. In addition we can do a application layer packet classification, putting some additional kernel hooks.

Advantages of Netfilter/Iptables

- •State matching - Connection tracking.
- •Automatic fragmentation reassembly - Connection tracking automatically reassembles fragmented packets for examination.
- •Improved matching - Advanced packet matching such as rate limit, string matching (packet data).
- •Improved logging - Customized logging levels and entries, also allows user space logging.
- •Allows packet mangling - Allows for the mangling of any information inside a packet.
- •Userspace queuing - Allows userspace programs access to packets.
- •Built-in support for port forwarding.

An rough explanation on how netfilter works is like this: The user instructs the kernel about what it needs to do with the IP packets that flow through the Linux box using the iptables tool. The Linux box then analyzes the IP headers on all packets flowing through it. If, when looking at the IP headers, the kernel finds matching rules, then the packet is manipulated according to the matching rule.



- If a rule does not match, try to match next rule.
- If a rule matches, take appropriate action.

2.2 Netfilter tables

1.Packet Filtering (Filter Table)
2.Network Address Translation (Nat Table)
3.Packet Mangling (Mangle Table)

Each containing a default set of rules, which are called chains. The default table loaded into the kernel is the filter table. The filter table contains contains following chains:

- INPUT Chain
- FORWARD Chain
- OUTPUT Chain

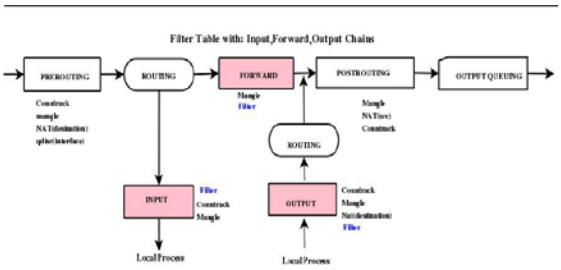


Fig.Filter Table

INPUT: used for traffic which is for our local machine

OUTPUT: used for traffic which originated on the local system, otherwise known as the firewall.

FORWARD: used for traffic which is being routed between two network interfaces on our firewall.

Immediately after a packet arrives at our Linux box, the mangle table PREROUTING chain is analyzed.At this point we can do all sorts of modifications on the IP packets supported by the mangle table (TOS byte modifications, TTL and Marking packets, and so on) before the routing process takes place.

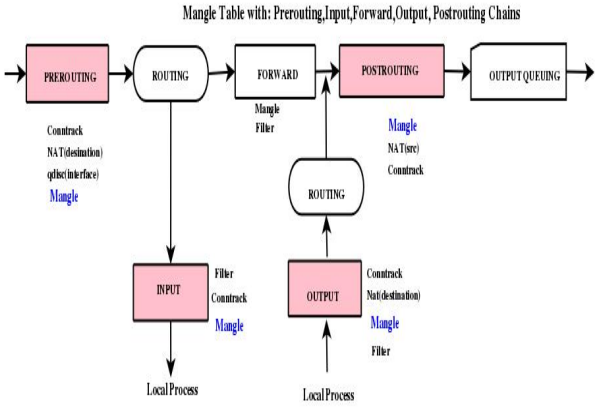


Fig.Mangle Table

PREROUTING Chain: The PREROUTING chain can be used to set netfilter, routing and SEC marks, both on a per packet basis and on a per connection basis.

INPUT Chain: The INPUT chain is could be used for mark handling.

FORWARD Chain: The FORWARD chain of the mangle table can be used for mark handling and for mangling packet headers of packets that are traveling across the firewall. Changing TTL and TOS.

OUTPUT Chain: The OUTPUT chain could be used to mangle the packets leaving the firewall or host itself, for example setting different marks or setting TTL or TOS values.

POSTROUTING Chain: This chain is basically used to setting values for all packets leaving both the host or firewall itself, and traffic traversing the machine. For example, it could be used to reset the MTU of packets, set TTL or TOS et cetera. Next, the packets flow through the pre-routing chain of the NAT table.

NAT tables allows a host or several hosts to share the same IP address in a way. NAT tables translates the source and destination addresses of packets as we already said to different addresses. The NAT receives the packet, rewrites the source and/or destination address and then recalculates the checksum of the packet. One of the most common usages of NAT is the SNAT (Source Network Address Translation) function. It also do the DNAT (Destination Network Address Translation) , DNAT is the process of translating one (usually public) IP address into another (usually private).

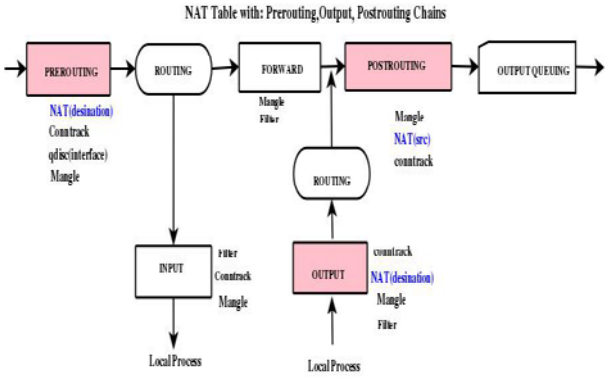


Fig.NAT Table

2.3 Iptables - Operations

The operations iptables can do with chains are:

- List the rules in a chain (iptables –L CHAIN).
- Change the policy of a chain (iptables –P CHAIN ACCEPT).
- reate a new chain (iptables –N CHAIN).
- Flush a chain; delete all rules (iptables –F CHAIN).
- Delete a chain (iptables –D CHAIN), only if the chain is empty.
- Zero counters in a chain (iptables –Z CHAIN). Every rule in every chain keeps a counter of the number of packets and bytes it matched. This command resets those counters.

Operations that iptables can execute on rules

•Append rules to a chain (iptables –A)

•Insert rules in a chain (iptables –I)

•Replace a rule from a chain (iptables –R)

•Delete a rule from a chain (iptables –D)

The most used switches are –A and –D (append and delete rules). Usually, when designing firewalls, the rules are appended to chains. During run time, users use –I more than –A because often they need to insert temporary rules in the chain.

iptables –A places the rule at the end of the chain, while iptables –I places the rule on the top of the other rules in the chain. However, you can insert a rule anywhere in the chain by specifying the position where you want the rule to be in the chain with the –I switch: iptables –I CHAIN 5 will insert a rule at the fourth position of the specified chain.

iptables –D can be used by specifying the position of the rule you want to delete

The syntax for adding a rule to a chain is:

iptables –A <CHAIN_NAME> …<filtering specifications>… -j <TARGET>

Filtering specifications is a part of an iptables rule that is used by the kernel to identify IP packets for which the kernel does the action specified by TARGET.

4.5 Filtering Specifications

IP packets can be identified in a large number of ways by specifying interfaces, protocols, ports, etc., to iptables rules.

- Filtering specifications for Layer2: Interfaces can be specified as selectors with –i and –o switches. -i stands for "--in-interface", and -o for "--out-interface".

- Filtering specifications for Layer 5: Source IP address(es) can be specified using -s, --src, or --source, and destination IP address(es) with -d,--dst, or --destination. Sources or destinations can be IP addresses, subnets, or canonical names.
- Filtering specifications for Layer 5: Protocol can be specified using the -p switch, which stands for "--protocol". Protocols can be specified by their corresponding numbers or by their names—tcp, udp, or icmp (case insensitive).
- For the ICMP protocol, you can specify ICMP message types using "--icmp-type".
- For the UDP protocol, you can specify source or destination ports with "--source-port" or "--sport" and "--destination-port" and "--dport".

TCP, being the most complete Layer 5 protocol, has more options. You can specify, besides source or destination ports as for the UDP protocol, "--tcp-flags", "--syn" and "--tcp-option". TCP flags can be: SYN, ACK, FIN, RST, URG, PSH, ALL, NONE ."--syn" is used to identify the initiating connections and is equivalent to "--tcp-flags SYN, RST, ACK SYN". "--tcp-option" followed by a number matches TCP packets with the option set to that number. Filtering specifications can combine all of the features just mentioned; so we can have a combination of Layers 2, 5, and 5 specifications in the same rule. A new and "daring" extension to iptables is to extend its capabilities from the lower layers to the upper layer of the OSI model, Layer 7-application. is called layer7-filter and it will do the application layer packet filtering.

2.4 Target Specifications

Following are the targets:

ACCEPT, DENY, DROP, REJECT, SNAT, DNAT, MASQUERADE, LOG

For the filter table, the most used targets for firewall rules are DROP and ACCEPT. If a rule matches the filtering specifications and has a DROP target, the packet will simply be discarded. If a packet matches a rule with a DROP target, the Linux kernel will drop the packet without consulting other rules in the firewall. If the target is ACCEPT, then the packet is accepted without further consultation of other firewall rules. An alternative to DROP is the REJECT target, which drops the packet but sends an ICMP packet to the source IP of the packet.

By default, the REJECT target will send an ICMP 'port unreachable' message to the sender, but that can be overwritten using the "--reject-with" switch. Another useful target is LOG, which can be used to log packets matching a filtering specification in the kernel log, which can be read with dmesg or syslogd.LOG target options are:

--log-level: The level of logging can be a name or a number. The valid names are debug, info, notice, warning, err, crit, alert, and emerg with corresponding numbers from 7 to 0.

--log-prefix prefix: Log prefix is followed by a string of up to 29 characters, placed at the beginning of the log message

--log-tcp-sequence: Logs TCP sequence numbers.

--log-tcp-options: Logs the option field of TCP packet headers.

--log-ip-options: Logs the option field of the IP packet headers.

--log-uid: Logs the user ID of the process that generated the packet.

The LOG target is not a terminating target like ACCEPT, DROP, and REJECT. This means that if a packet matches a rule that has the LOG target, the kernel looks up the rules that follow to also match this packet. A limit match for rules with LOG targets would be a good idea to prevent flooding the log files.

iproute2

iproute2 is a software package that provides various tools for advanced routing, tunnels, and traffic control. iproute2 is well known for implementing QoS in Linux kernels. The most important tools that iproute2 provides are ip and tc.

ip Tool

The ip tool provides most of the networking configuration a Linux box needs. You can configure interfaces, ARP, policy routing, tunnels, etc. Now, with IPv5 and Ipv6.

To see what ip knows:

[root@igarbo ~]# ip help

Usage: ip [OPTIONS] OBJECT { COMMAND | help }

where OBJECT := { link | addr | route | rule | neigh | tunnel | maddr | mroute | monitor | xfrm } OPTIONS := { -V[ersion] | -s[tatistics] | -r[esolve] | -f[amily] { inet | inet6 | ipx | dnet | link }| -o[neline] }

The ip link command shows the network device's configurations that can be changed with ip link set. This command is used to modify the network device's proprieties and not the IP(IPV5/6) address.The IP addresses can be configured using the ip addr command. This command can be used to add a primary or secondary (alias) IP address to a network device (ip addr add), to display the IP addresses for each network device (ip addr show), or to delete IP addresses from interfaces (ip addr del).IP addresses can also be flushed using different criteria, e.g. ip addr flush dynamic will flush all routes added to the kernel by a dynamic routing protocol. Neighbor/Arp table management is done using ip neighbor, which has a few

commands expressively named add, change, replace, delete, and flush.
One very important and probably the most used object of the ip tool is ip route, which can do any operations on the kernel routing table. It has commands to add, change, replace, delete, show,flush, and get routes.

Traffic Control (tc) tool
The tc command allows us to build different QoS policies in their networks using Linux instead of very expensive dedicated QoS hardware boxes.
The tc tool performs all of the configuration of the kernel structures required to support traffic control. The utility takes as its first non-option argument one of three Linux traffic control components, qdisc, clsss or filter.
In order to match Layer 7 data, netfilter looks deeper into an IP packet than just at its header. However, the actual data contained in the packet doesn't just say "I'm a P2P packet; filter me!"; so
the data is matched against a set of regular expressions that are common to different applications. This set of regular expressions is probably the most important part of this project, and is called "protocol definitions".
The L7-filter project contains three important parts:
1.A kernel patch, which provides a way for the kernel to look into the IP packets
2.An iptables patch, which provides the match option for iptables.
Netfilter/iptables:
netfilter is a very important part of the Linux kernel in terms of security, packet mangling, and manipulation. The front end for netfilter is iptables, which "tells" the kernel what the user wants to do with the IP packets arriving into, passing through, or leaving the Linux box.
The most used features of netfilter are packet filtering and network address translation, but there are a lot of other things that we can do with netfilter, such as packet mangling Layer 7 filtering.
A rough explanation on how netfilter works is like this:
1.The user instructs the kernel about what it needs to do with the IP packets that flow through the Linux box using the iptables tool.
2.The Linux box then analyzes the IP headers on all packets flowing through it.
3.If, when looking at the IP headers, the kernel finds matching rules, then the packet is manipulated according to the matching rule.

3.CONCLUSION

It might look very simple at the beginning, but actually is a lot more complicated process. netfilter has a few tables, each containing a default set of rules, which are called chains. The default table loaded into the kernel is the filter table, which contains three chains, that contains rules for packets destined to the Linux machine itself, for packets that the Linux machine routes to another IP address, rules for packets generated by the Linux machine. The advantage is the low cost and works on nonstandard protocol.

REFERENCES

1. Lucian Gheorghe "*Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and LFilter*" Packt Publishing, October 2006
2. Application Layer Packet Classifier for Linux website "http://l7filter.sourceforge.net/"
3. Netfilter, firewalling, nat and packet mangling for linux website http://www.netfilter.org
4. Nigel Kukard "*Bandwidth Management and Optimization*" International Network INASP,Opensource Bandwidth Solutions March 2006
5. Lukas Kencl, Christian Schwarzer, "*TrafficAdaptive Packet Filtering of Denial of ServiceAttacks*" Intel Research laboratories, World of Wireless, Mobile and Multimedia Networks,2006. WoWMoM 2006.
6. J. McCann and Satish Chandra, "*Packet Types:Abstract Specification of Network ProtocolMessages"* Bell Laboratories, ACM SIGCOMM Computer Communication ReviewVolume 30 , Issue 4 October 2000
7. Jeffrey C. Mogul, " *The Packet Filter An Efficient Mechanism for UserlevelNetworkCode*"Digital Equipment Corporation Western Research Laboratory, ACM Operating SystemsReview, SIGOPS
8. HolgerDreger, AnjaFeldmann, Michael Mai, Vern Paxson, Robin Sommer "*DynamicApplicationLayerProtocol Analysis for Network Intrusion Detection*" USENIX Security
9. Pankaj Gupta ,Nick McKeown "*Packet Classification on Multiple Fields*", Proc. Sigcomm,Computer Communication Review, vol. 29, no. 4, pp 14760,September 1999, HarvardUniversity.
10. Florin Baboescu George Varghese "*Scalable Packet Classification* " , University of California,San Diego Proceedings of ACM Sigcomm, pages 199210,August, 2001.