NETFILTER

Introduction :

- It is a framework for packet mangling .
- Each protocol defines “hooks” which are well-defined points in a packet’s traversal of that protocol stack.
- At each of these points , the protocol will call the netfilter framework with the packet and the hook number .
- Kernel can register to these hooks and manage packets .
- ip_tables provide the rules_set which does the packet mangling .

Program :

```
import iptc
table = iptc.Table(iptc.Table.FILTER)
chain = iptc.Chain(table,'FORWARD')
#policy = iptc.Policy(iptc.Policy.ACCEPT)
rule = iptc.Rule()
#rule.in_interface = "wlo1"
rule.src="98.136.103.24/255.255.255.0"
rule.dst="98.136.103.24/255.255.255.0"
rule.protocol="tcp"
match=iptc.Match(rule,"state")
match.state="RELATED,ESTABLISHED"
rule.add_match(match)
rule.target=iptc.Target(rule,"DROP")
chain.insert_rule(rule)
table.commit()
#displays all the chains in that particular table
for  chain in table.chains:
    print(chain.name)
    for rule in chain.rules:
        print(rule.protocol,rule.src,rule.dst)
        for match in rule.matches:
            print(match.name)
        print(rule.target.name)
        #chain.delete_rule(rule)
```

NETFILTER

```
Activities  Terminal  Wed Sep 19, 2:50 PM
root@praveen: /home/praveen
File Edit View Search Terminal Help
praveen@praveen:~$ tcpdump -i any
tcpdump: any: You don't have permission to capture on that device
(socket: Operation not permitted)
praveen@praveen:~$ sudo su
[sudo] password for praveen:
root@praveen:/home/praveen# tcpdump -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
14:49:51.221998 IP 172.18.5.38.59449 > 255.255.255.255.1947: UDP, length 40
14:49:51.224631 ARP, Request who-has 169.254.146.37 tell 172.18.5.2, length 46
14:49:51.227118 IP localhost.44388 > localhost.domain: 24968+ PTR? 255.255.255.255.in-addr.arpa. (46)
14:49:51.227621 IP localhost.domain > localhost.44388: 24968 ServFail 0/0/0 (46)
14:49:51.372741 IP6 praveen.mdns > ff02::fb.mdns: 0 PTR (QM)? 37.146.254.169.in-addr.arpa. (45)
14:49:56.318451 IP localhost.35683 > localhost.domain: 45021+ PTR? 53.0.0.127.in-addr.arpa. (41)
14:49:56.318699 IP localhost.55249 > localhost.domain: 45706+ PTR? b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa. (9
0)
14:49:56.318960 IP praveen.41032 > google-public-dns-a.google.com.domain: 14246+ PTR? b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2
.0.f.f.ip6.arpa. (90)
14:49:56.362345 IP google-public-dns-a.google.com.domain > praveen.41032: 14246 NXDomain 0/1/0 (154)
14:49:56.363280 IP localhost.41469 > localhost.domain: 44965+ PTR? 8.8.8.8.in-addr.arpa. (38)
14:49:56.363400 IP praveen.53625 > google-public-dns-a.google.com.domain: 59950+ PTR? 8.8.8.8.in-addr.arpa. (38)
14:49:56.405069 IP google-public-dns-a.google.com.domain > praveen.53625: 59950 1/0/0 PTR google-public-dns-a.google.com. (82)
14:49:56.463677 ARP, Request who-has _gateway tell 172.18.5.19, length 46
14:49:56.463823 IP localhost.37004 > localhost.domain: 28653+ PTR? 1.0.18.172.in-addr.arpa. (41)
14:49:56.464014 IP praveen.41269 > google-public-dns-a.google.com.domain: 64882+ PTR? 1.0.18.172.in-addr.arpa. (41)
14:49:56.472523 ARP, Request who-has _gateway tell 172.18.5.19, length 46
14:49:56.506229 IP google-public-dns-a.google.com.domain > praveen.41269: 64882 NXDomain 0/0/0 (41)
14:49:56.872624 ARP, Request who-has 169.254.146.37 tell 172.18.5.2, length 46
14:49:57.723983 ARP, Request who-has 169.254.146.37 tell 172.18.5.2, length 46
14:49:57.791025 ARP, Request who-has 169.254.146.37 tell 172.18.5.30, length 46
14:49:57.791162 IP localhost.56762 > localhost.domain: 64263+ PTR? 30.5.18.172.in-addr.arpa. (42)
14:49:57.791371 IP praveen.54595 > google-public-dns-a.google.com.domain: 3753+ PTR? 30.5.18.172.in-addr.arpa. (42)
14:49:57.832143 IP google-public-dns-a.google.com.domain > praveen.54595: 3753 NXDomain 0/0/0 (42)
14:49:57.832346 IP localhost.domain > localhost.56762: 64263 NXDomain 0/0/0 (42)
14:49:58.383907 ARP, Request who-has 169.254.146.37 tell 172.18.5.30, length 46
14:49:58.724072 ARP, Request who-has 169.254.146.37 tell 172.18.5.2, length 46
14:49:59.316591 IP 172.18.5.58.60284 > 224.0.0.252.hostmon: UDP, length 38
14:49:59.316772 IP localhost.33401 > localhost.domain: 62049+ PTR? 252.0.0.224.in-addr.arpa. (42)
```

OUTPUT :

before adding rules

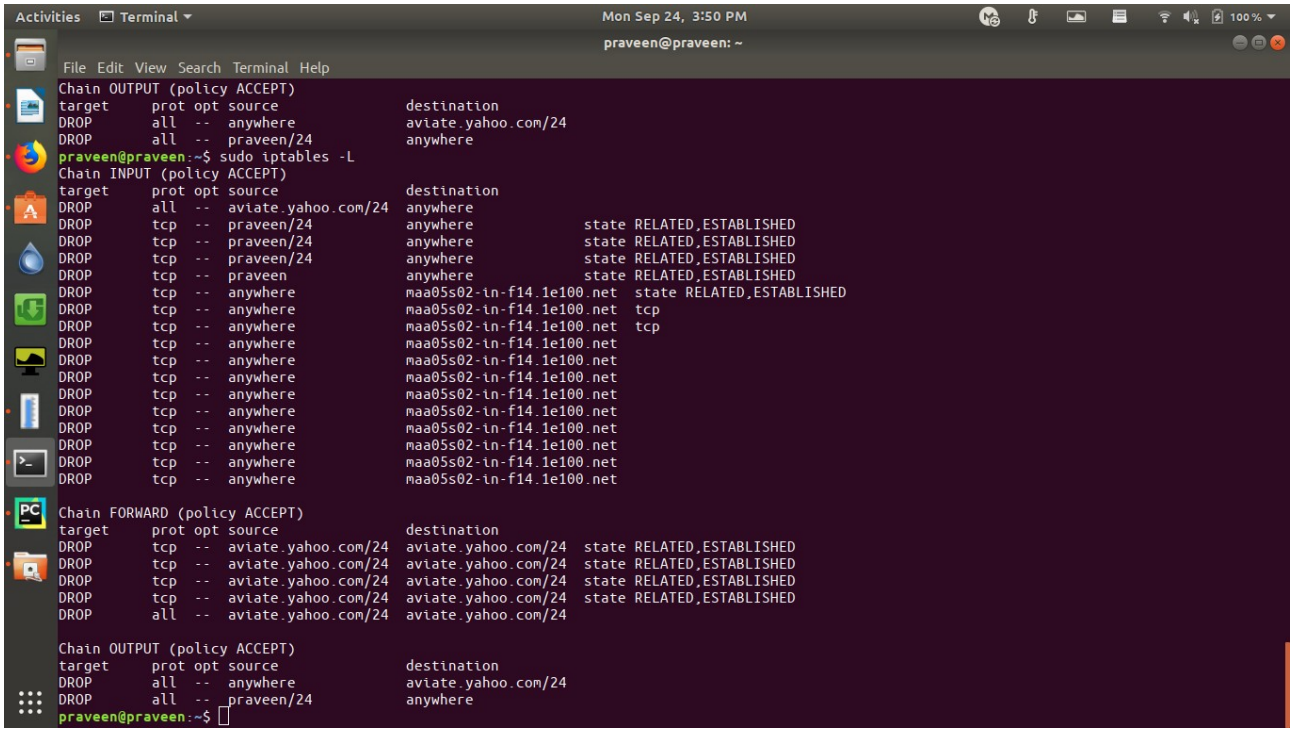
```
Activities  Terminal  Mon Sep 24, 3:49 PM
praveen@praveen: ~
File Edit View Search Terminal Help
64 bytes from aviate.yahoo.com (212.82.100.151): icmp_seq=3 ttl=46 time=186 ms
64 bytes from aviate.yahoo.com (212.82.100.151): icmp_seq=4 ttl=46 time=180 ms
64 bytes from aviate.yahoo.com (212.82.100.151): icmp_seq=5 ttl=46 time=180 ms
^Z
[4]+  Stopped                 ping yahoo.in
praveen@praveen:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  aviate.yahoo.com/24  anywhere
DROP       tcp  --  praveen/24           anywhere             state RELATED,ESTABLISHED
DROP       tcp  --  praveen/24           anywhere             state RELATED,ESTABLISHED
DROP       tcp  --  praveen/24           anywhere             state RELATED,ESTABLISHED
DROP       tcp  --  praveen              anywhere             state RELATED,ESTABLISHED
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net  state RELATED,ESTABLISHED
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net  tcp
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net  tcp
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net
DROP       tcp  --  anywhere             maa05s02-in-f14.1e100.net

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  aviate.yahoo.com/24  aviate.yahoo.com/24  state RELATED,ESTABLISHED
DROP       tcp  --  aviate.yahoo.com/24  aviate.yahoo.com/24  state RELATED,ESTABLISHED
DROP       tcp  --  aviate.yahoo.com/24  aviate.yahoo.com/24  state RELATED,ESTABLISHED
DROP       all  --  aviate.yahoo.com/24  aviate.yahoo.com/24

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             aviate.yahoo.com/24
DROP       all  --  praveen/24           anywhere
praveen@praveen:~$
```

NETFILTER

after adding rules



python-iptables :

Iptables is the tool that is used to manage **netfilter**, the standard packet filtering and manipulation framework under Linux.

- Iptables is used to set up, maintain, and inspect the tables of IPv4 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.
- Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a *target*, which may be a jump to a user-defined chain in the same table.

Installating via pip :

>pip install --upgrade python-iptables

NETFILTER

main classes in python-iptables :

1) Table :

A table is the most basic building block in iptables.

Different tables :

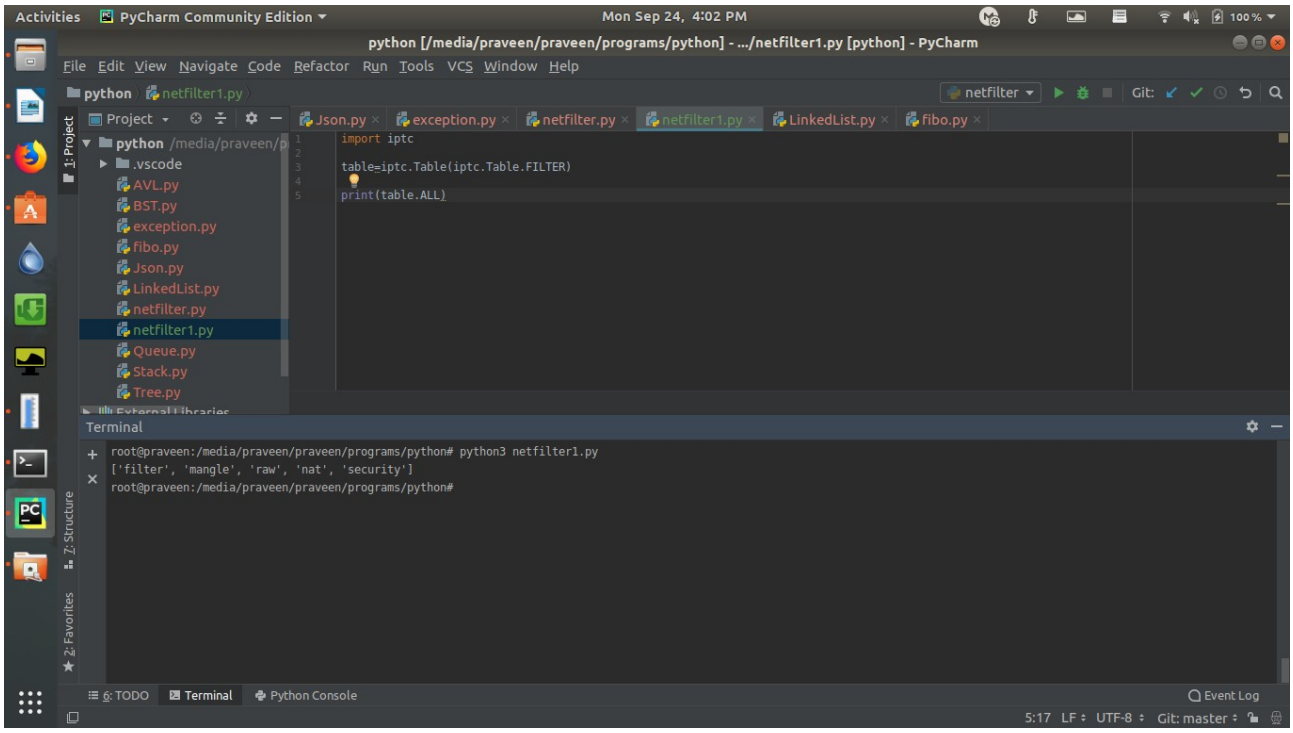
- Table.FILTER – filter table
- Table.NAT – NAT table
- Table.MANGLE – MANGLE table
- Table.RAW – RAW table

to get access to table :

table = iptc.Table(iptc.Table.FILTER)

options :

a)table.ALL – lists all the tables



NETFILTER

b)list all chains in the table

program :

```
import iptc
table=iptc.Table(iptc.Table.FILTER)
#print(table.ALL)
for chain in table.chains:
    print(chain.name)
```

output :

```
Terminal
root@praveen:/media/praveen/praveen/programs/python# python3 netfilter1.py
INPUT
FORWARD
OUTPUT
root@praveen:/media/praveen/praveen/programs/python#
```

2)Chain :

chain contains the rules .

a) insert_rule – insert the rule as first entry in the chain eg : first program

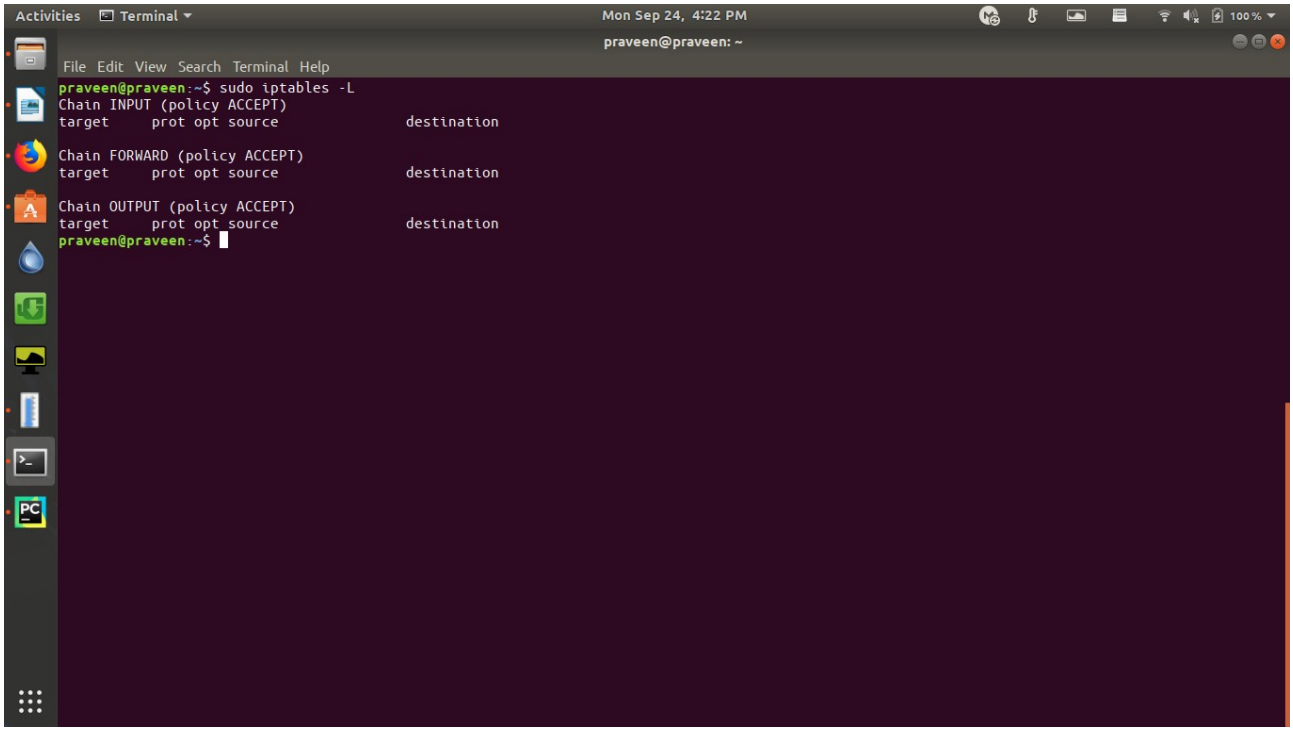
b)flush – flush all rules from chain

program :

```
import iptc
table=iptc.Table(iptc.Table.FILTER)
for chain in table.chains:
    chain.flush()
```

output :

NETFILTER



3) Target :

Targets specify what to do when rule is matched for a packet . It can drop the packet or accept the packet .

References :

- https://python-iptables.readthedocs.io/en/latest/
- https://netfilter.org/documentation/HOWTO//netfilter-hacking-HOWTO.html