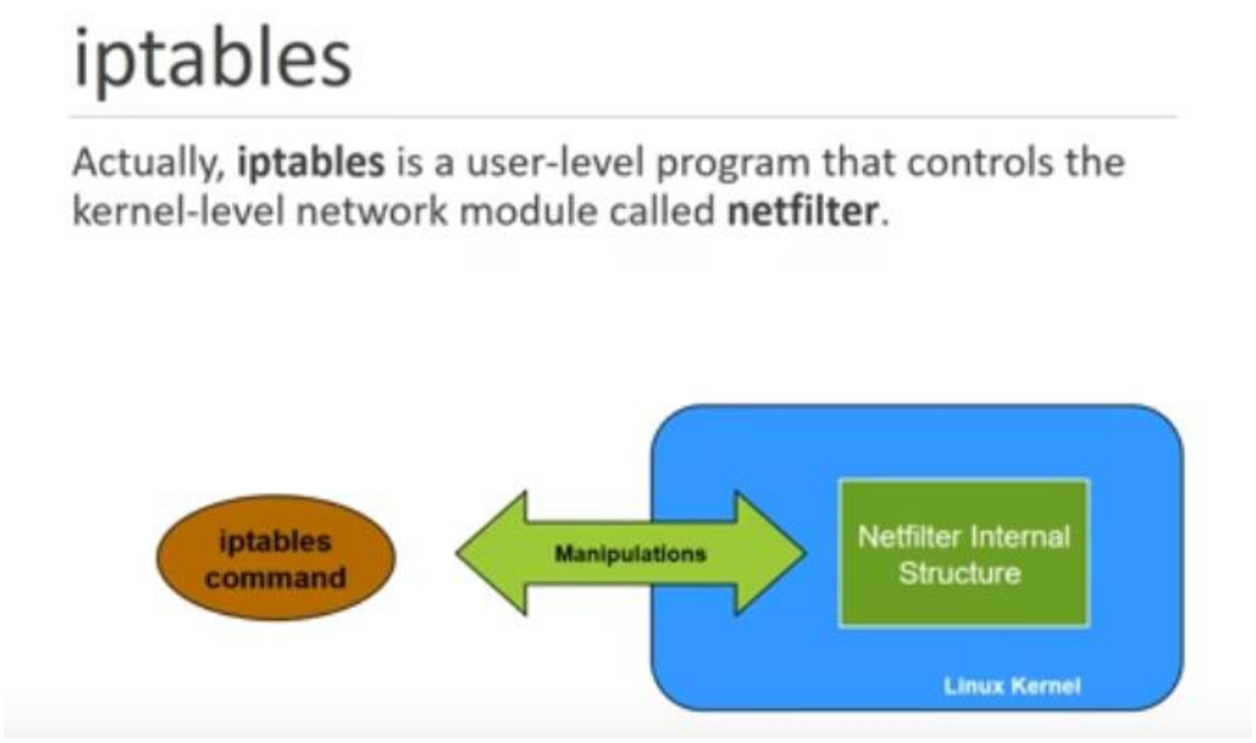
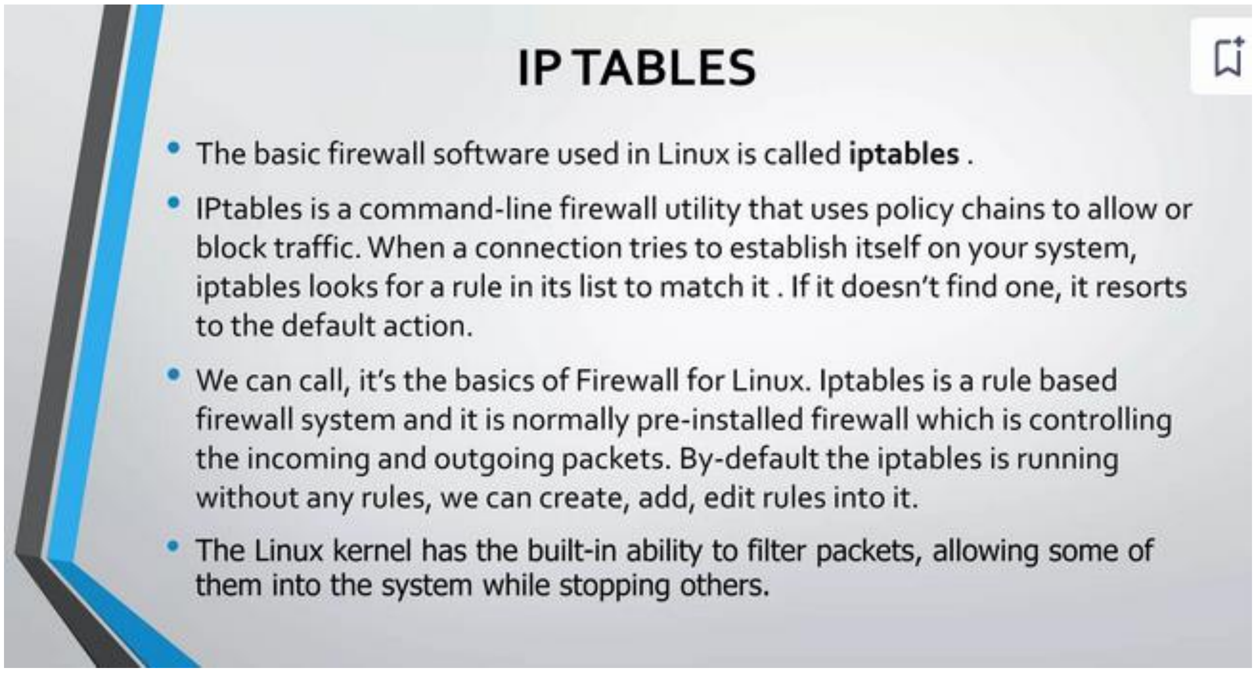


iptables

Actually, **iptables** is a user-level program that controls the kernel-level network module called **netfilter**.

IP TABLES: is a commond -line user level program for configuring the linux kernel firewall , which is part of the Netfilter Project. It allows system administrator to define rules and policies for packet filtering , Network Address Translation (NAT), and packet mangling in the Linux kernel firewall.



IP TABLES

- The basic firewall software used in Linux is called **iptables** .
- IPtables is a command-line firewall utility that uses policy chains to allow or block traffic. When a connection tries to establish itself on your system, iptables looks for a rule in its list to match it . If it doesn't find one, it resorts to the default action.
- We can call, it's the basics of Firewall for Linux. Iptables is a rule based firewall system and it is normally pre-installed firewall which is controlling the incoming and outgoing packets. By-default the iptables is running without any rules, we can create, add, edit rules into it.
- The Linux kernel has the built-in ability to filter packets, allowing some of them into the system while stopping others.

IPtables Tutorial

IPtables is a packet filter-based implementation of the Linux kernel firewall (netfilter). It defines tables that contain a chain of rules that specify how packets should be treated. The hierarchy is iptables --> tables --> chains --> rules. There may be built-in tables and chains as well as user-defined ones.

There are three independent tables (the presence of a table depends on the kernel configuration options): ***filter***, ***nat*** and ***mangle***. We specify the table to be used through the **-t** option.

- The ***filter*** table is the default table (if no -t option is used) and it has three built-in chains:
  INPUT (for packets destined for the local sockets);
  FORWARD (for packets being routed through a machine) and
  OUTPUT (packets originating from local sockets).
- The ***nat*** table is used when a packet encountered by the router/firewall has to go through network address translation. The nat table consists of three built-in chains:
  PRE-ROUTING - used to change the destination IP address of the incoming packets
  POST-ROUTING - used to change the source IP address of the outgoing packets
  OUTPUT - used to alter and send out the locally generated packets

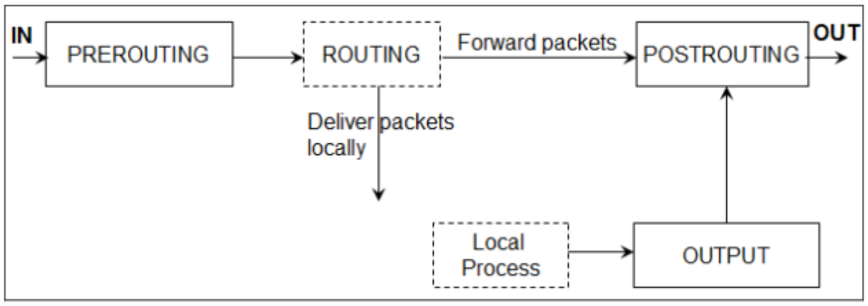
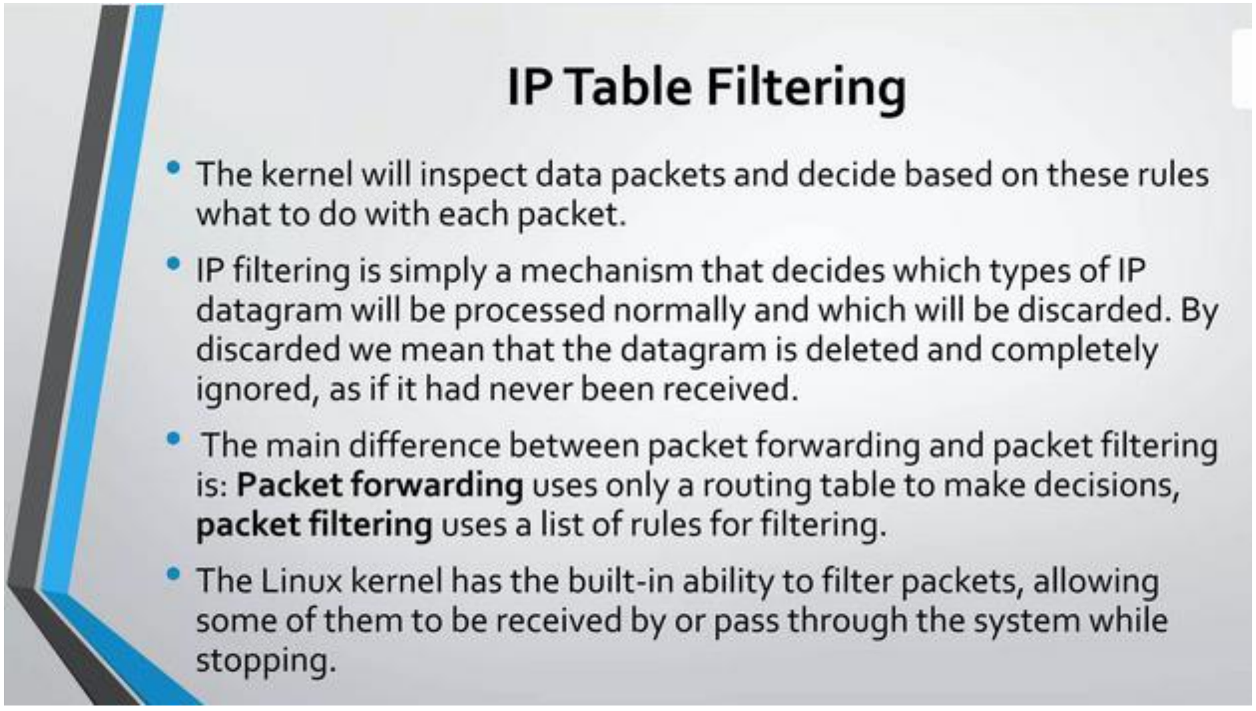
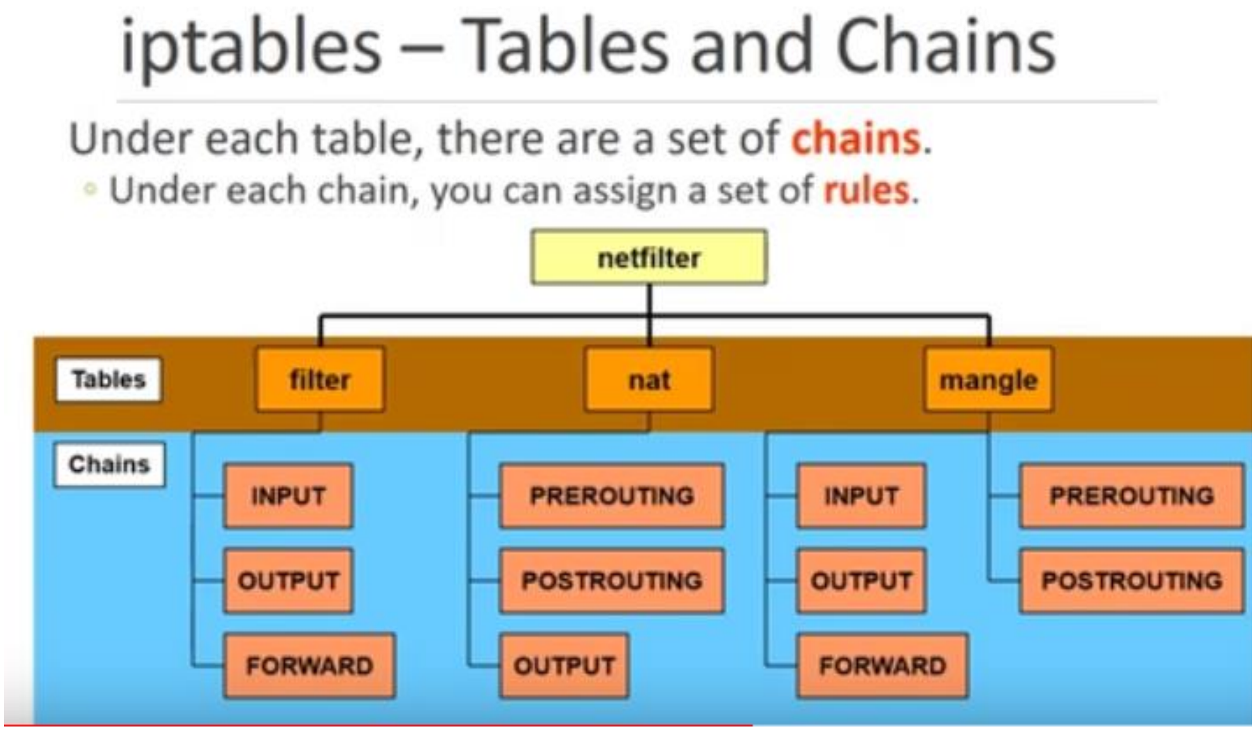


Figure 1: NAT Table



IP Table Filtering

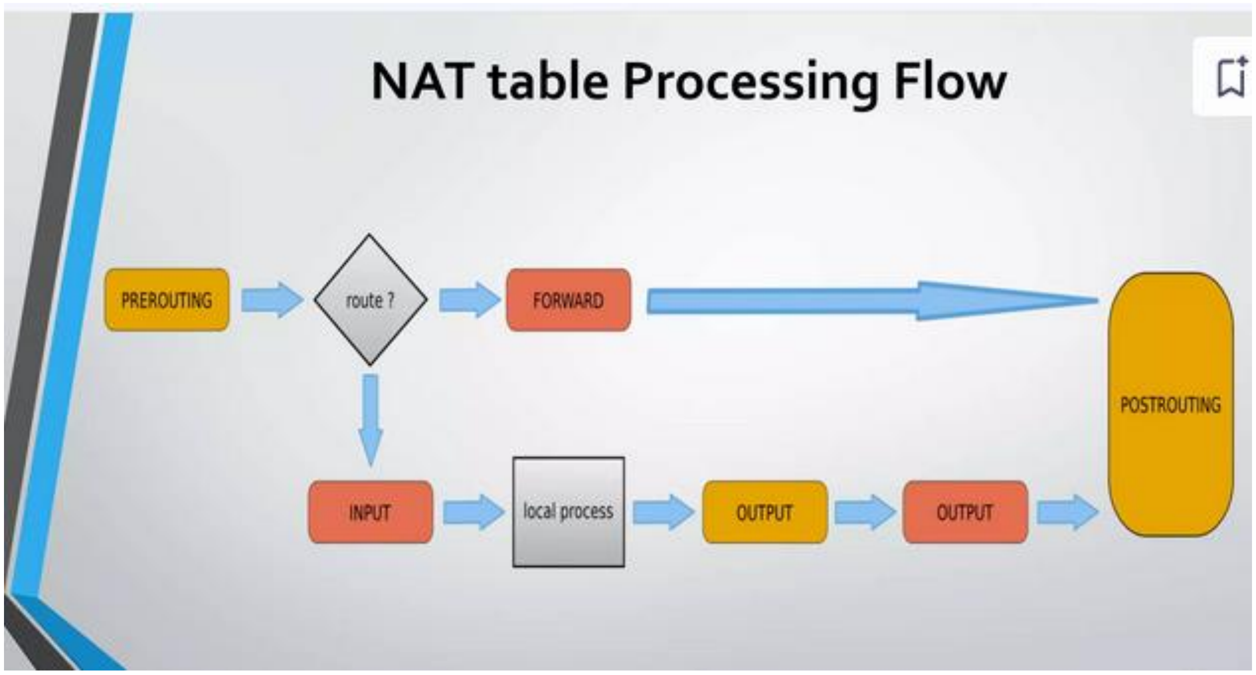
- The kernel will inspect data packets and decide based on these rules what to do with each packet.
- IP filtering is simply a mechanism that decides which types of IP datagram will be processed normally and which will be discarded. By discarded we mean that the datagram is deleted and completely ignored, as if it had never been received.
- The main difference between packet forwarding and packet filtering is: **Packet forwarding** uses only a routing table to make decisions, **packet filtering** uses a list of rules for filtering.
- The Linux kernel has the built-in ability to filter packets, allowing some of them to be received by or pass through the system while stopping.



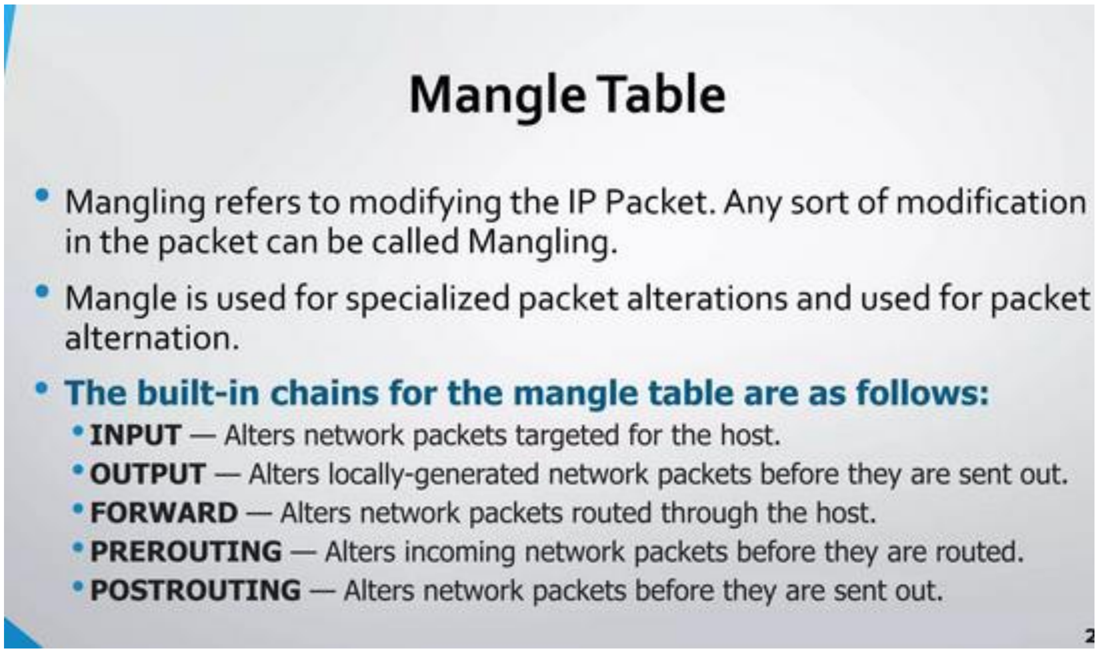
iptables – Tables and Chains

Under each table, there are a set of **chains**.

- Under each chain, you can assign a set of **rules**.

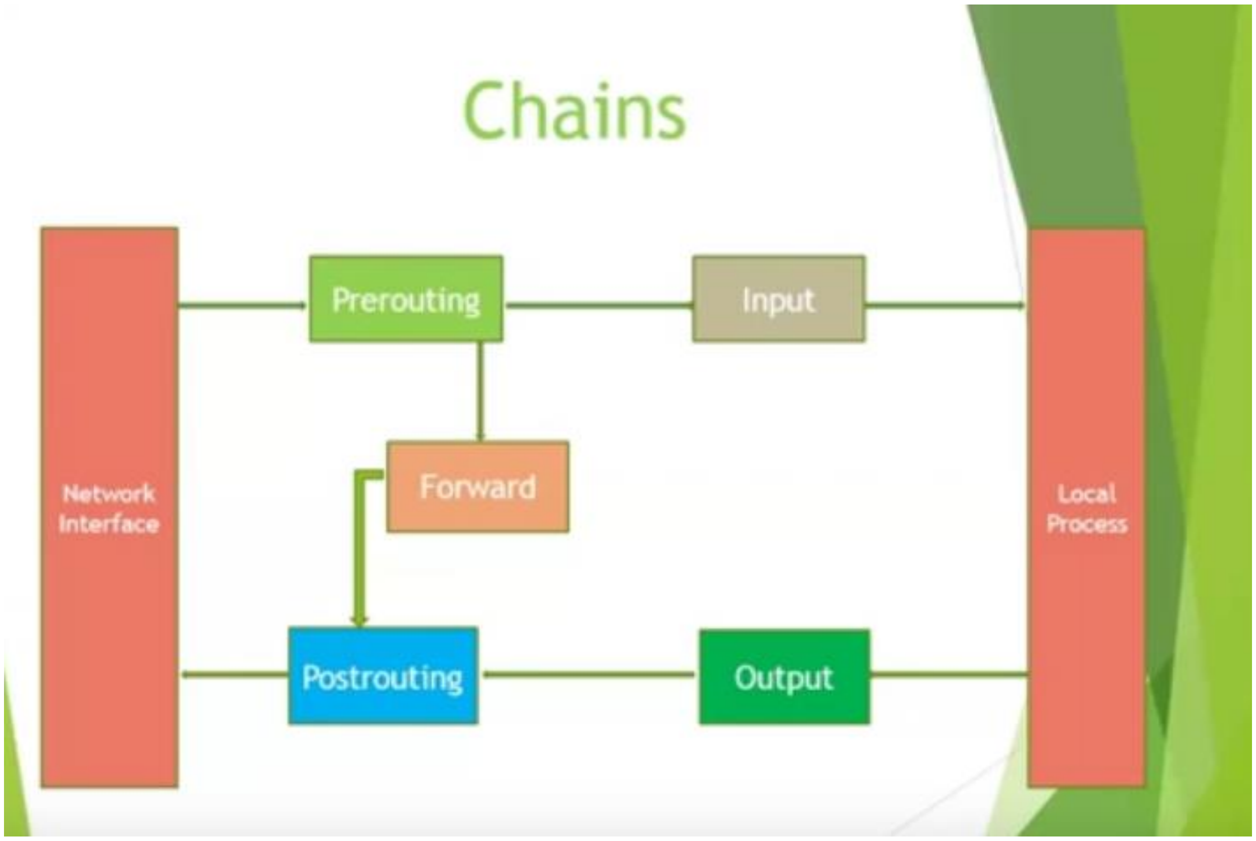


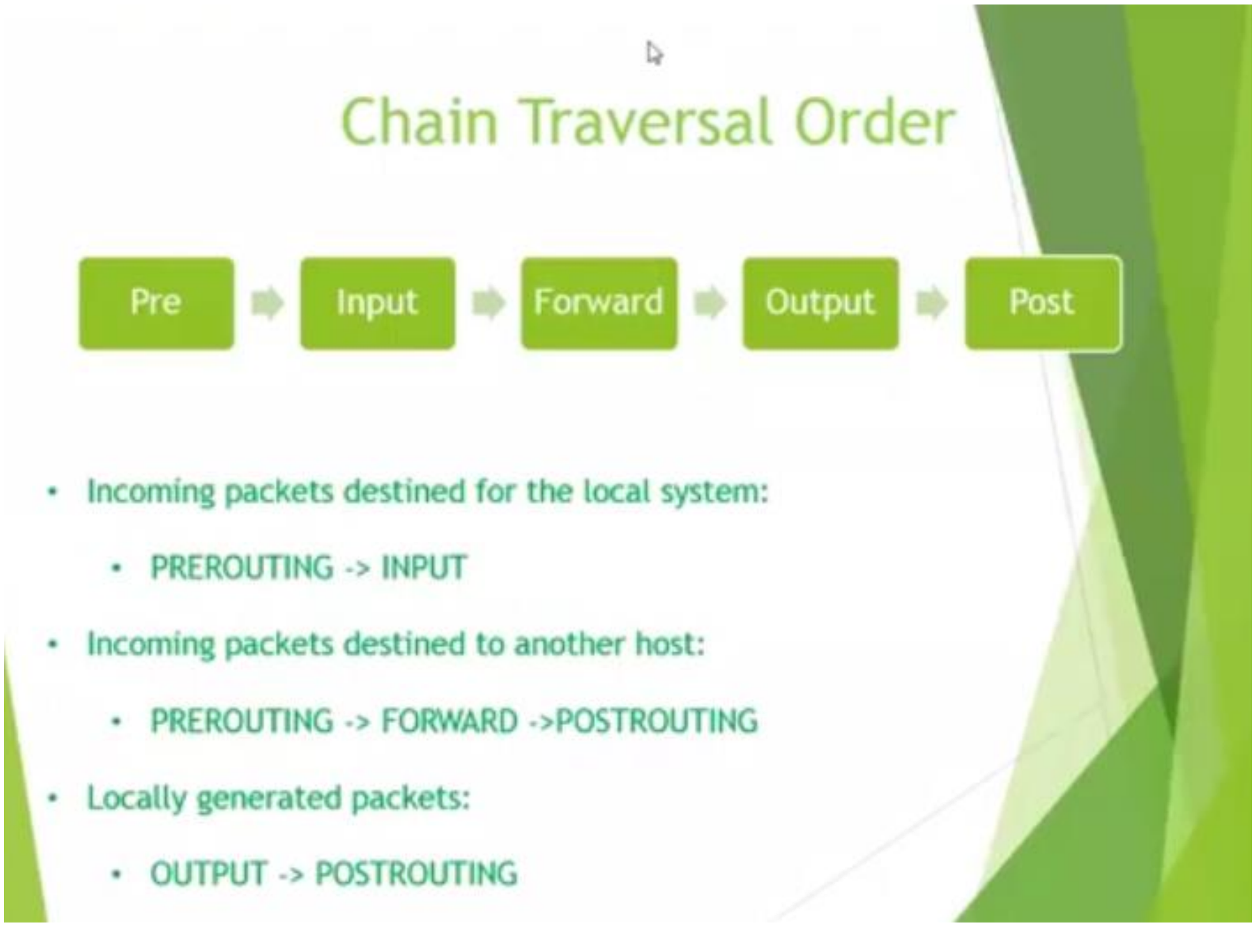
NAT table Processing Flow



Mangle Table

- Mangling refers to modifying the IP Packet. Any sort of modification in the packet can be called Mangling.
- Mangle is used for specialized packet alterations and used for packet alternation.
- **The built-in chains for the mangle table are as follows:**
  - **INPUT** — Alters network packets targeted for the host.
  - **OUTPUT** — Alters locally-generated network packets before they are sent out.
  - **FORWARD** — Alters network packets routed through the host.
  - **PREROUTING** — Alters incoming network packets before they are routed.
  - **POSTROUTING** — Alters network packets before they are sent out.

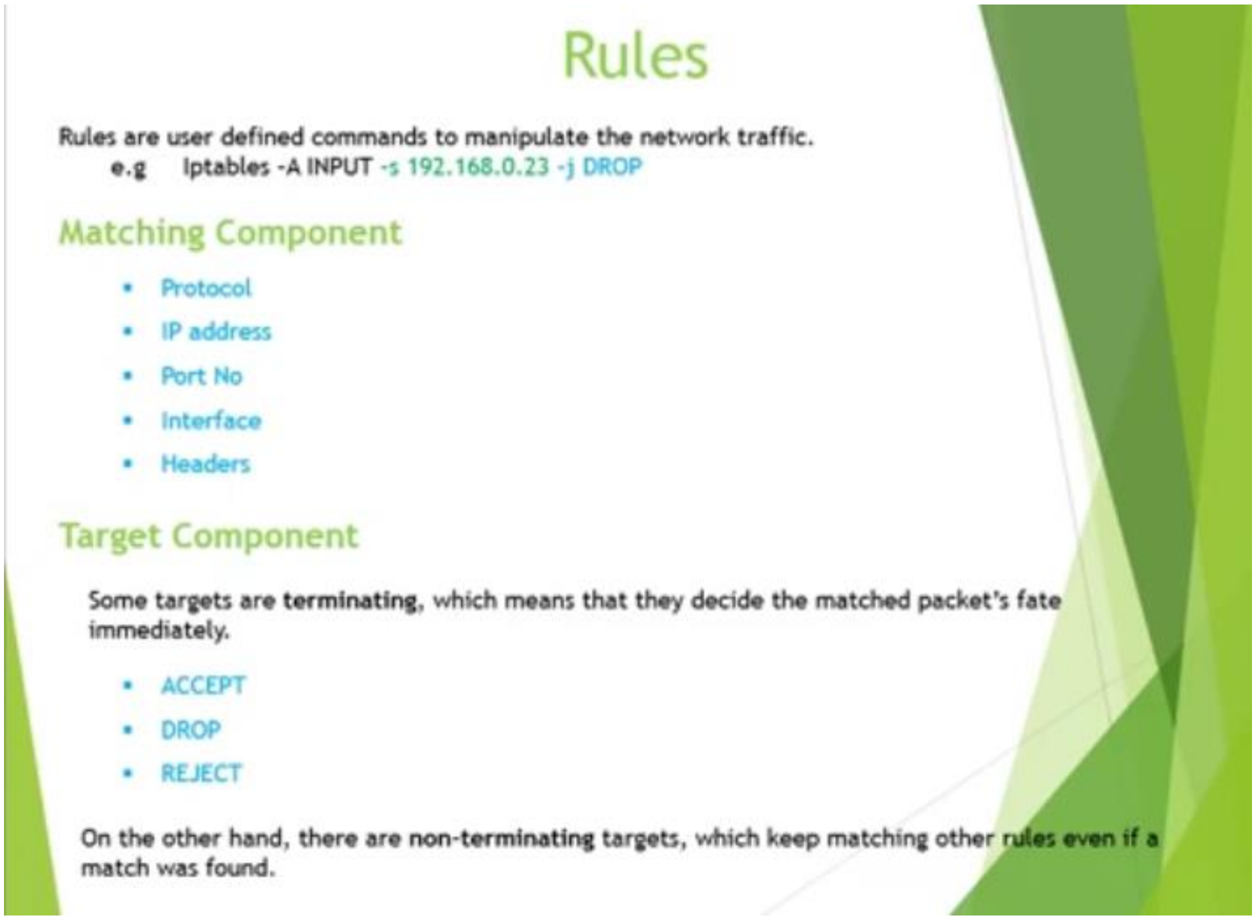




Chain Traversal Order

Pre
Input
Forward
Output
Post

- Incoming packets destined for the local system:
  - PREROUTING -> INPUT
- Incoming packets destined to another host:
  - PREROUTING -> FORWARD ->POSTROUTING
- Locally generated packets:
  - OUTPUT -> POSTROUTING



Rules

Rules are user defined commands to manipulate the network traffic.

e.g Iptables -A INPUT -s 192.168.0.23 -j DROP

Matching Component

- Protocol
- IP address
- Port No
- Interface
- Headers

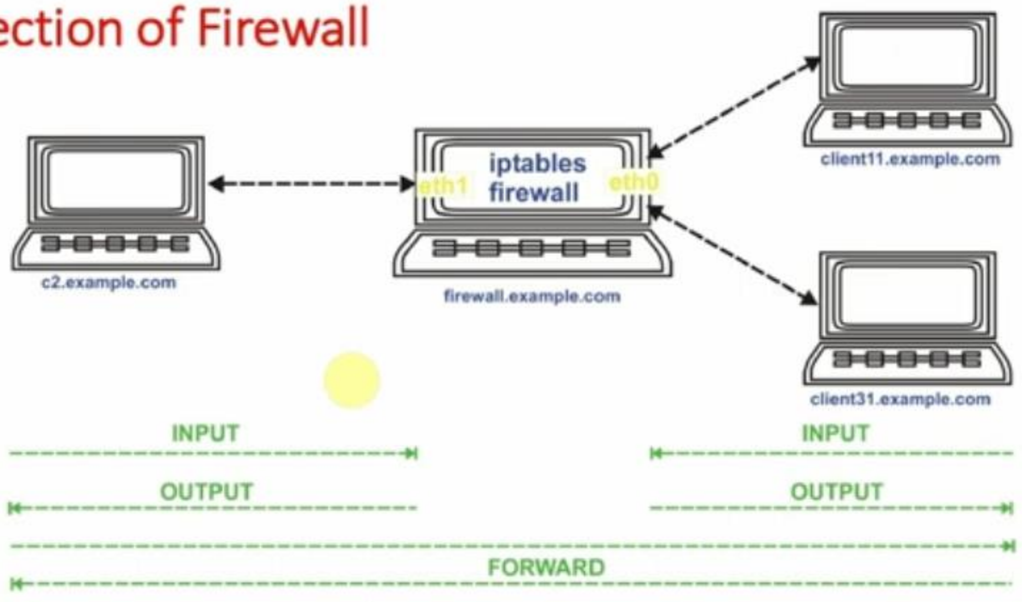
Target Component

Some targets are **terminating**, which means that they decide the matched packet's fate immediately.

- ACCEPT
- DROP
- REJECT

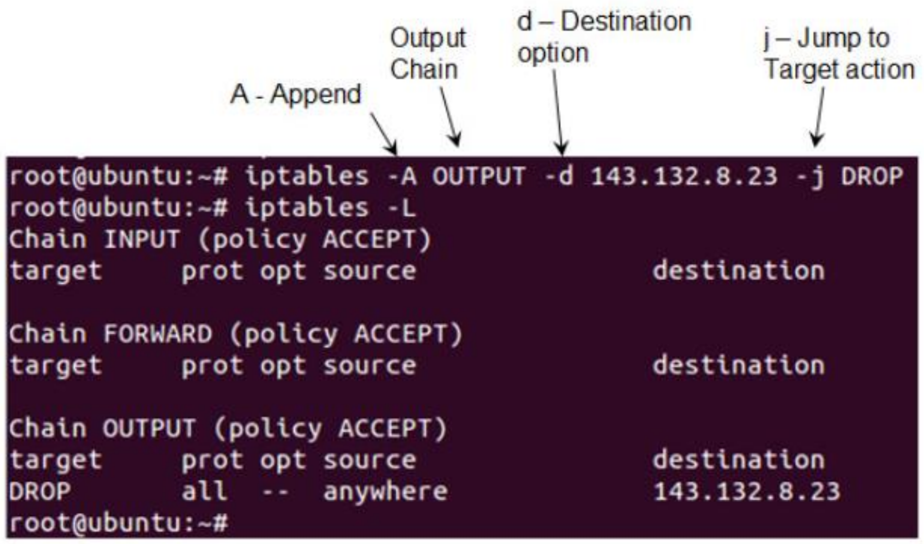
On the other hand, there are **non-terminating** targets, which keep matching other rules even if a match was found.

Direction of Firewall



INPUT means traffic coming towards/terminating at firewall.
OUTPUT means traffic originating from firewall and going towards c2, client11 or client31
FORWARD means traffic going through firewall (from c2 to client11 or client31 and vice versa)

<u>Command:</u> iptables -A OUTPUT -d 143.132.8.23 -j DROP



```
root@ubuntu:~# iptables -A OUTPUT -d 143.132.8.23 -j DROP
root@ubuntu:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             143.132.8.23
root@ubuntu:~#
```

We could open a web browser and try to visit www.google.com; we could visit without any problem. On the other hand, try to visit www.jsums.edu; you will only see a message on the browser telling "connecting to...," but it could not connect eventually.

Individual commands method

If you prefer to configure the software firewall by using discrete steps instead of by using the one-line command, perform the following steps:

1. Run the following command to allow traffic on port 80:

   ```
   sudo iptables -I INPUT -p tcp -m tcp --dport 80 -j ACCEPT
   ```

2. Run the following command to allow traffic on port 443:

   ```
   sudo iptables -I INPUT -p tcp -m tcp --dport 443 -j ACCEPT
   ```

3. Run the following command to save the `iptables` rules:

   ```
   sudo service iptables save
   ```

iptables -A (or **--append**): Appends a rule to the end of a chain.

    iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT

iptables -I (or **--insert**): Inserts a rule at a specific position in a chain.

iptables -I INPUT 2 -s 10.0.0.0/24 -j DROP

iptables -D (or **--delete**): Deletes a rule from a chain by specifying its rule number or matching

```
iptables -D INPUT -s 192.168.1.0/24 -j ACCEPT
```

iptables -L (or **--list**): Lists all rules in a chain or all chains if none specified.

```
iptables -L INPUT
```