What is iptables?

IPTables is the default command-line-based stateless firewall for many Linux machines. In short, it is the thing that decides what happens to all incoming and outgoing packets. When a connection tries to establish itself to a system, IPTables matches it to a rule in its list(if none match default action is taken). Important note here is that the order of the rules in IPTables matters because it starts checking from the first rule and goes down the chain.

General IPTables format :

iptables → Tables → Chains → Rules

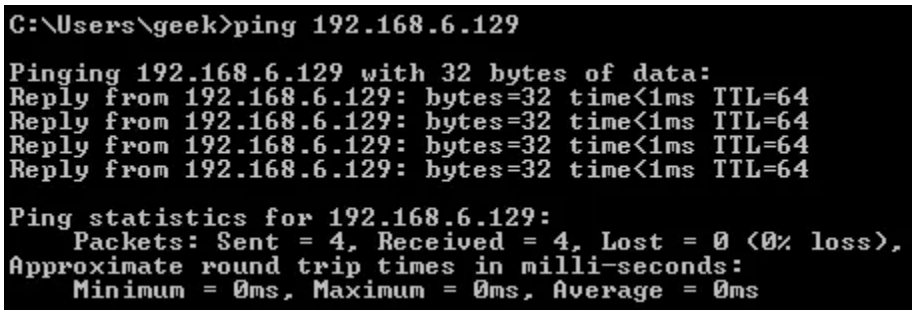
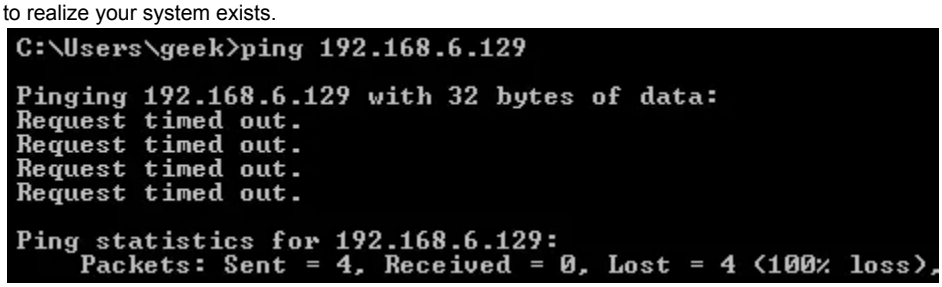
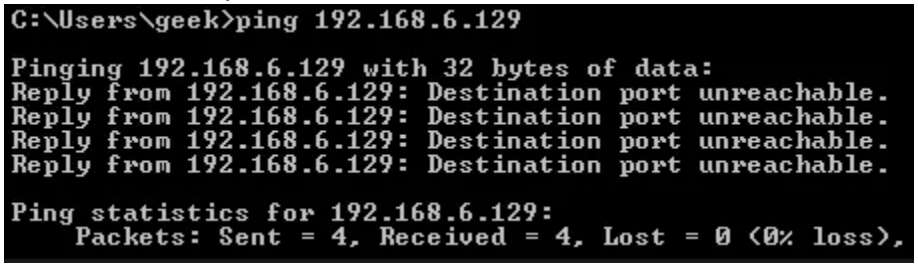
iptables [-t table] command [match pattern] [action]

*IMPORTANT : if you are using ufw, its settings will override any iptables rules!

iptables tables	
filter	Used for normal filtering of traffic based on rules defined by the user(accept, reject, etc). Most used table in the iptables firewall. It is helpful in carrying out normal day-to-day blocking and filtering.
nat	Used for Network Address Translation purposes. NAT is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.
mangle	Rules in this table can be used to modify the packers based on the user given criteria. It can be used to modify the TTL, MSS value, TOS(traffic priority), etc. Used to alter QOS bits in the TCP header.
raw	Primarily used to add No Connection Tracking rules.
security	Used for Mandatory Access Control networking rules.

iptables chains		
INPUT Chain used to control the behavior of incoming connections Ex. a user attempting to SSH into your server: iptables will match the user's IP address and port to a rule in the input chain.	**FORWARD** Chain used for incoming connections that aren't actually being delivered locally. Primarily used on pass-through devices like routers. Ex. NAT, routing, etc.	**OUTPUT** Chain used for outgoing connections Ex. you attempt to ping google.com : iptables will check what rules are regarding ping and google.com.
POSTROUTING(NAT) Alter packets before routing; packet translation happens immediately after the packet comes to the system(and before routing). An example use case is translating the destination IP address of the packet to something that matches the routing on the local machine; used for destination NAT(DNAT).	**PREROUTING(NAT)** Alter packets after routing; packet translation happens when the packers are leaving the system. Used for source NAT(SNAT).	**OUTPUT(NAT)** NAT for locally generated packets on the firewall

iptables target values	
ACCEPT	Firewall will accept the packet. Example of pinging INPUT ACCEPT :

	
DROP	Firewall will drop the packet. Drop/ignore the connection. Use this option if you don't want the sender to realize your system exists. 
QUEUE	Firewall will pass the packet to the userspace.
RETURN / REJECT	Firewall will stop executing the next set of rules in the current chain for this packet. The control will be returned to the calling chain.Use this option if you don't want the sender to establish a connection but want them to know that your firewall blocked them. 
MASQUERADE	The masquerade option is used exactly the same as SNAT. The difference is each time that the MASQUERADE target gets hit by a packet, it automatically checks for the IP address to use-unlike SNAT, where the target is matched using single configured IP addresses. This means that the MASQUERADE option works properly with Dynamically allocated IP addresses,(DHCP).

Policy Chain DEFAULT behavior

When no matches are found for a specific packet, iptables will default to a set of pre-defined rules.To check what your iptables default policy chains are :

```
# iptables -L | grep policy
```

```
┌──(root㉿pwner3000)-[/home/deusxmachina]
└─# iptables -L | grep policy
Chain INPUT (policy ACCEPT)
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)
```

To change the default policy chain rules :

 # iptables --policy <chain-name> <target-value>
iptables --policy INPUT DROP
iptables --policy OUTPUT ACCEPT
iptables --policy FORWARD DROP
iptables -L | grep policy

```
┌──(root㉿pwner3000)-[/home/deusxmachina]
└─# iptables -L | grep policy
Chain INPUT (policy DROP)
Chain FORWARD (policy DROP)
Chain OUTPUT (policy ACCEPT)
```

Allowing or blocking specific connections

Aside from the default policy chains, you can configure additional rules to allow or block specific connections. You can either append a rule :

 # iptables -A <chain-name> <rule-specification> [options]

*{block all connections **from** IP address 10.0.0.10}*
iptables --append **INPUT** --source 10.0.0.10 -j DROP
iptables -A **INPUT** -s 10.0.0.10 -j DROP // same command as above

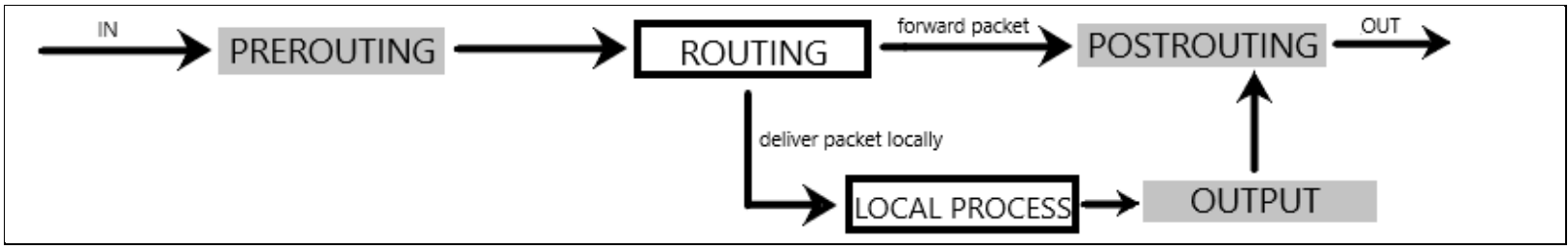
*{allow all connections **to** IP addresses 10.0.0.1 ~ 10.0.0.255}*
iptables --append **OUTPUT** --destination 10.0.0.0/24 -j ACCEPT
iptables -A **OUTPUT** -d 10.0.0.0/24 -j ACCEPT // same command as above
iptables -A **OUTPUT** -d 10.0.0.0/255.255.255.0 -j ACCEPT // same command as above

*{block SSH connections **from** 10.0.0.10 - assuming SSH is running on port #22}*
iptables -A **INPUT** -p tcp --dport 22 -s 10.0.0.10 -j DROP
iptables -A **INPUT** -p tcp --dport ssh -s 10.0.0.10 -j DROP // same command as above

*{block SSH connections **from** anywhere}*
iptables -A **INPUT** -p tcp --dport ssh -j DROP

NAT table examples

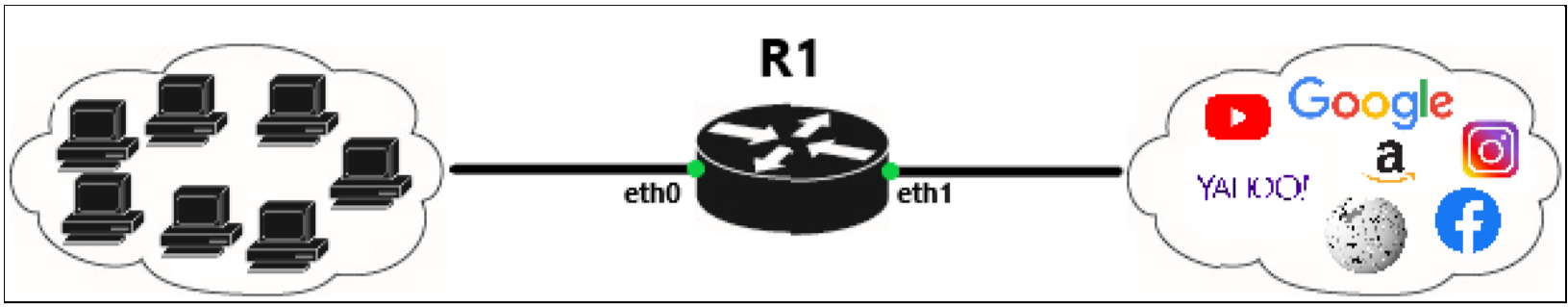
As shown in the ‘iptables tables’ table, the NAT table is used for mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device. This means that the NAT table option is used to change both the source address and destination address of a rule-matching packet.
The relationship between the three predefined chains-PREROUTING, OUTPUT, and POSTROUTING-are as following :



Before any further NAT configurations, we need to activate IP-forwarding in the kernel :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward          //enable IP-forwarding
# echo 0 > /proc/sys/net/ipv4/ip_forward          //disable IP-forwarding
```

In the following scenario, interface eth0 is connected to the local net, while eth1 is connected to the internet :



Packets arriving from the local net need their source IP address changed such that it is equal to the router's address. We configure on R1 :

```
# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

-t : select NAT table for configuration of NAT rules
-A POSTROUTING : append a rule to the POSTROUTING chain
-o eth1 : this rule is valid for packets that leave on the network interface 'eth1'
-j MASQUERADE : the action that should take place is to 'masquerade' packets; replace sender address with router's

Choosing match patterns for NAT table

To match TCP packets **from** 10.0.0.1 :

```
# iptables -t nat -A POSTROUTING -p tcp -s 10.0.0.1 [action]
```

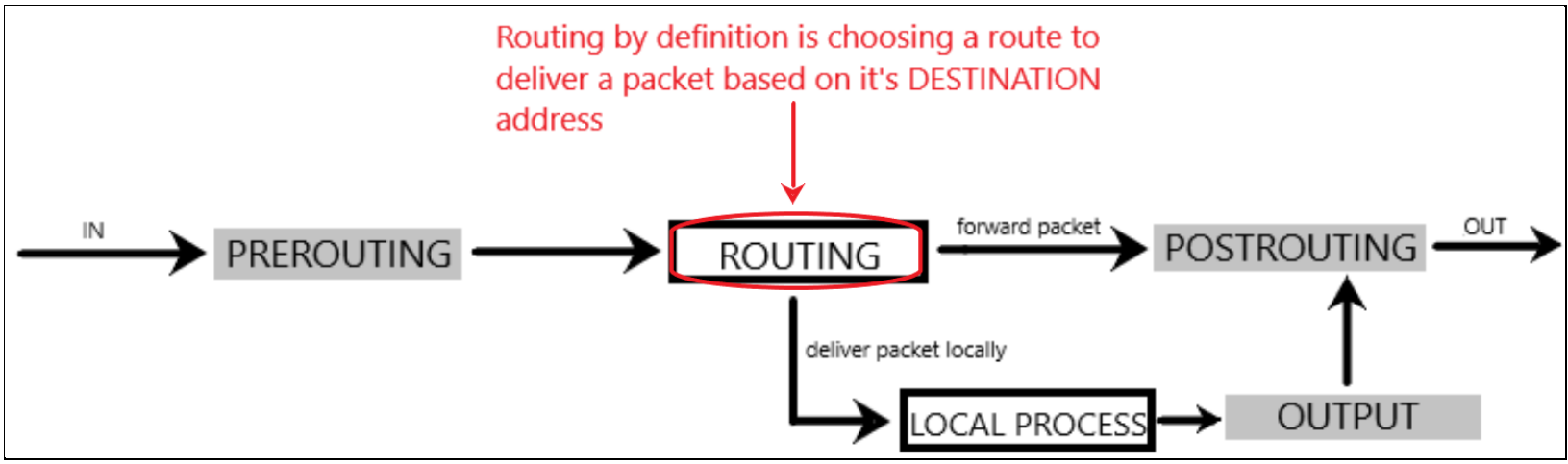
To match UDP packets going **to** 20.0.0.1 :

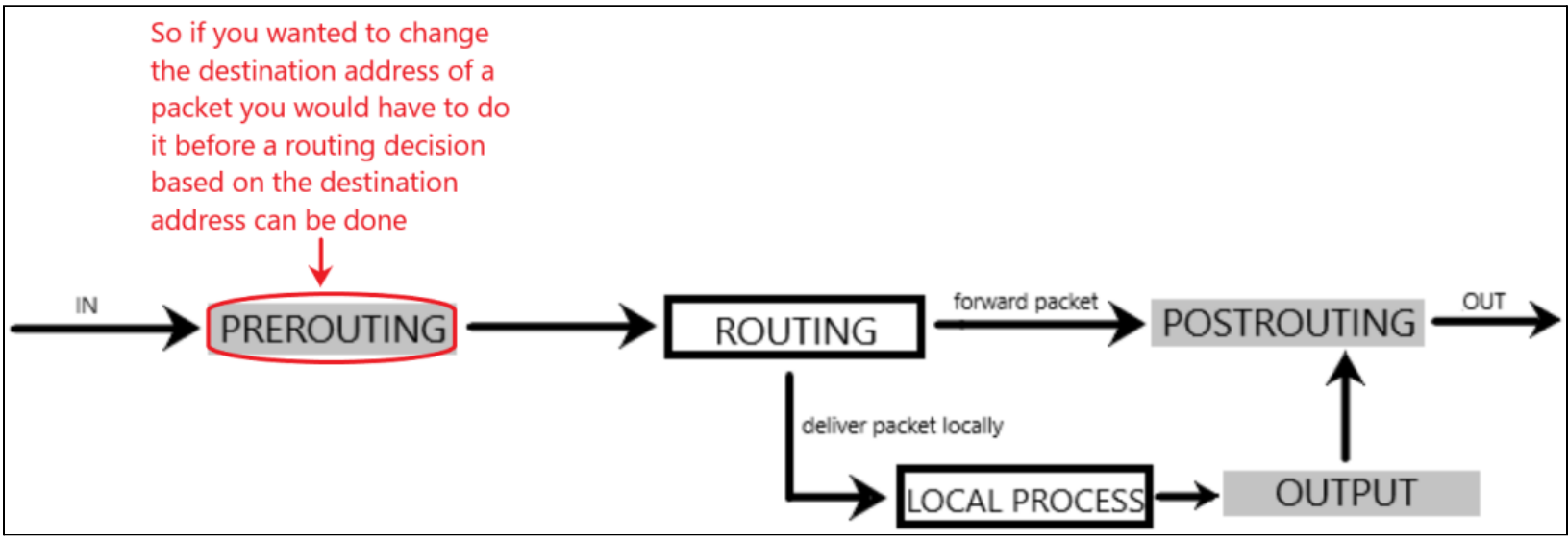
```
# iptables -t nat -A POSTROUTING -p udp -d 20.0.0.1 [action]
```

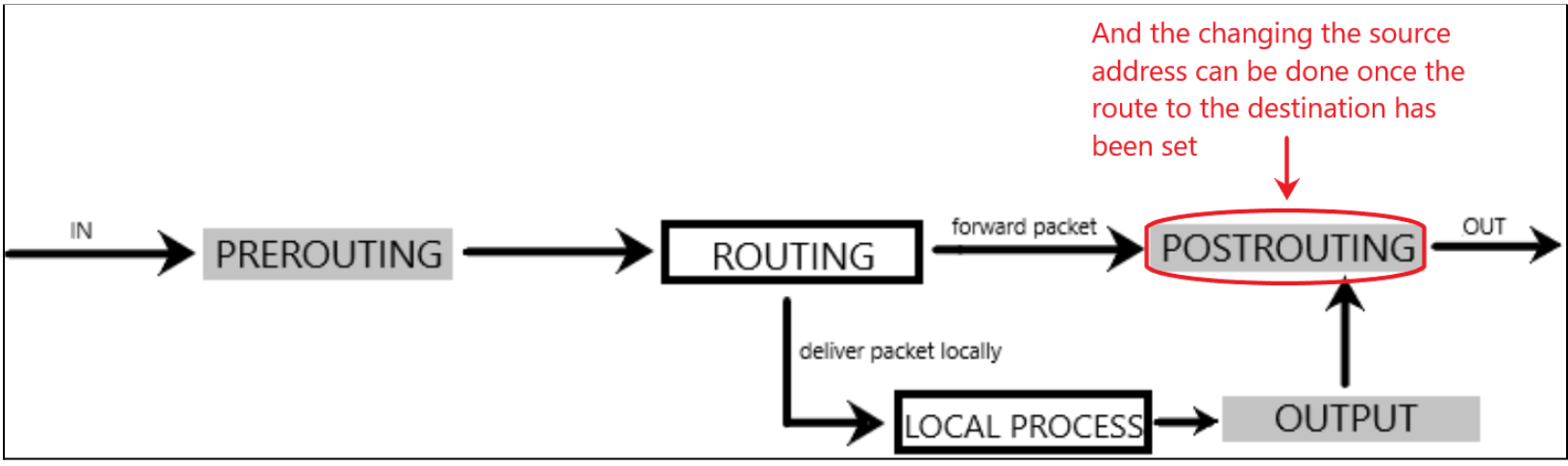
To match all packets from 30.0.0.0/16 arriving at eth0 :

```
# iptables -t nat -A PREROUTING -s 30.0.0.0/16 -i eth0  [action]
```

IMPORTANT! Remember that
PREROUTING is used for DNAT for incoming packets
OUTPUT is used for DNAT for outgoing packets
POSTROUTING is used for SNAT for outgoing (local or forwarded) packets
An easy way to understand and memorize these distinctions is :







To match all packets except TCP packets and except packets from 192.168.1.2 :

```
# iptables -t nat -A PREROUTING -p ! tcp -s ! 192.168.1.2 [action]
```

To match packets leaving at eth1 :

```
# iptables -t nat -A POSTROUTING -o eth1 [action]
```

To match TCP packets from 192.168.1.2 ports 12345~12356 to 1.1.1.1 port 22 :

```
# iptables -t nat -A PREROUTING -p tcp -s 192.168.1.2 --sport 12345:12356 -d 1.1.1.1 --dport 22 [action]
```

Actions for NAT table

To change sender address to 123.123.123.123 :

```
# iptables [match pattern] -j SNAT --to-source 123.123.123.123
```

To change sender address to outgoing network interface :

```
# iptables [match pattern] -j MASQUERADE
```

To change destination address to 123.123.123.123 port 22 :

```
# iptables [match pattern] -j DNAT --to-destination 123.123.123.123:22
```

To redirect packet to local port 8080 :

```
# iptables [match pattern] -j REDIRECT --to-ports 8080
```

Solve two-way communication problems with ‘Connection States’

You may have recognized a problem with the previous SSH filtering rules :
Many protocols require a two-way communication to function. Ex. when you ping www.google.com you need an OUTPUT rule allowing ping to the target AND an INPUT rule to retrieve the ping response. In the case of SSH, our previous OUTPUT and INPUT ACCEPT configuration would allow both external users to connect to our SSH server and us to connect to their SSH server. What if we want only external users to connect to our SSH(and block us from establishing a connection to their SSH)?
In the following example, we allow SSH connections FROM 10.0.0.1

```
# iptables -A INPUT -p tcp --dport ssh -s 10.0.0.1 -m state --state NEW,ESTABLISHED -j ACCEPT
```

But new SSH connections TO 10.0.0.1 are not. However, the system is permitted to send back information over SSH as long as the session has already been established(as in not initiating the connection).

```
# iptables -A OUTPUT -p tcp --sport 22 -d 10.0.0.1 -m state --state ESTABLISHED -j ACCEPT
(default deny any)
```

Save iptables configuration

To prevent current iptables configured rules from disappearing upon the next service reboot, run the following command(for Ubuntu)

```
# iptables-save
```

Resources

https://www.karlrupp.net/en/computer/nat_tutorial