TRAINING SESSION NAME

Firewall

Table of contents

A. Main Content

1. Concepts:

iptables defines five “hook points” in the kernel’s packet processing pathways:

PREROUTING, INPUT, FORWARD, POSTROUTING, and OUTPUT.

Built-in chains are attached to these hook points; you can add a sequence of rules for each hook point. Each rule represents an opportunity to affect or monitor packet flow

2. Hook points

Hook	Allows you to process packets...
FORWARD	... that flow through a gateway computer, coming in one interface and going right back out another.
INPUT	... just before they are delivered to a local process.
OUTPUT	... just after they are generated by a local process.
POSTROUTING	... just before they leave a network interface.

PREROUTING	... just as they arrive from a network interface (after dropping any packets resulting from the interface being in promiscuous mode and after checksum validation).

3. Built-in tables

Table	Description
nat	Used with connection tracking to redirect connections for network address translation; typically based on source or destination addresses. Its built-in chains are OUTPUT, POSTROUTING, and PREROUTING.
filter	Used to set policies for the type of traffic allowed into, through, and out of the computer. Unless you refer to a different table explicitly, iptables operate on chains within this table by default. Its built-in chains are: FORWARD, INPUT, and OUTPUT.
mangle	Used for specialized packet alteration, such as stripping off IP options (as with the IPV4OPTSSTRIP target extension). Its built-in chains are: FORWARD, INPUT, OUTPUT, POSTROUTING, and PREROUTING. making changes to packet header fields (such as network addresses and port numbers) or payloads.

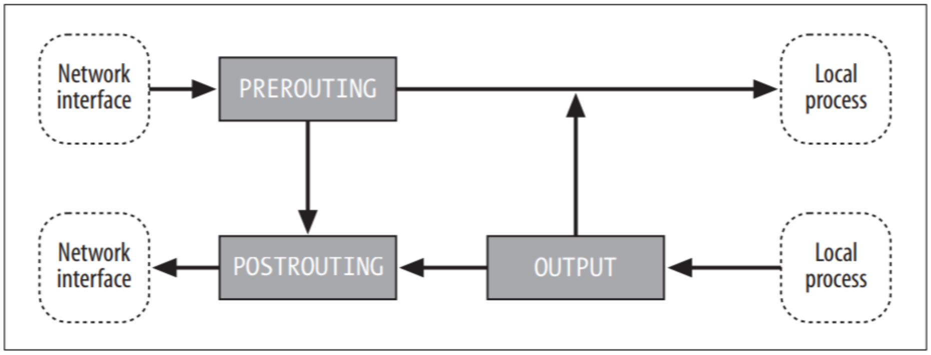


Figure 1. Network packet flow and hook points for NAT

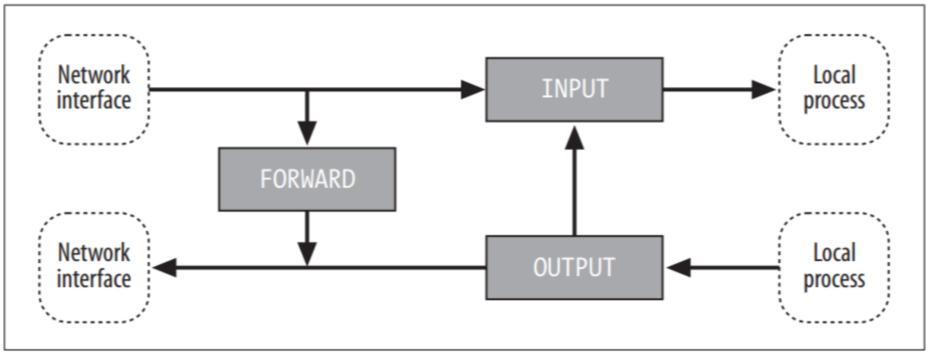


Figure 2. Network packet flow and hook points for filtering

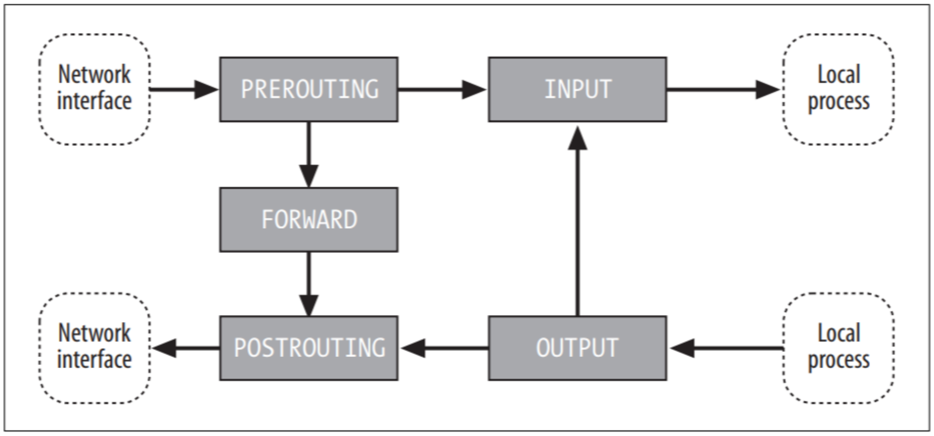


Figure 3. Network packet flow and hook points for mangling

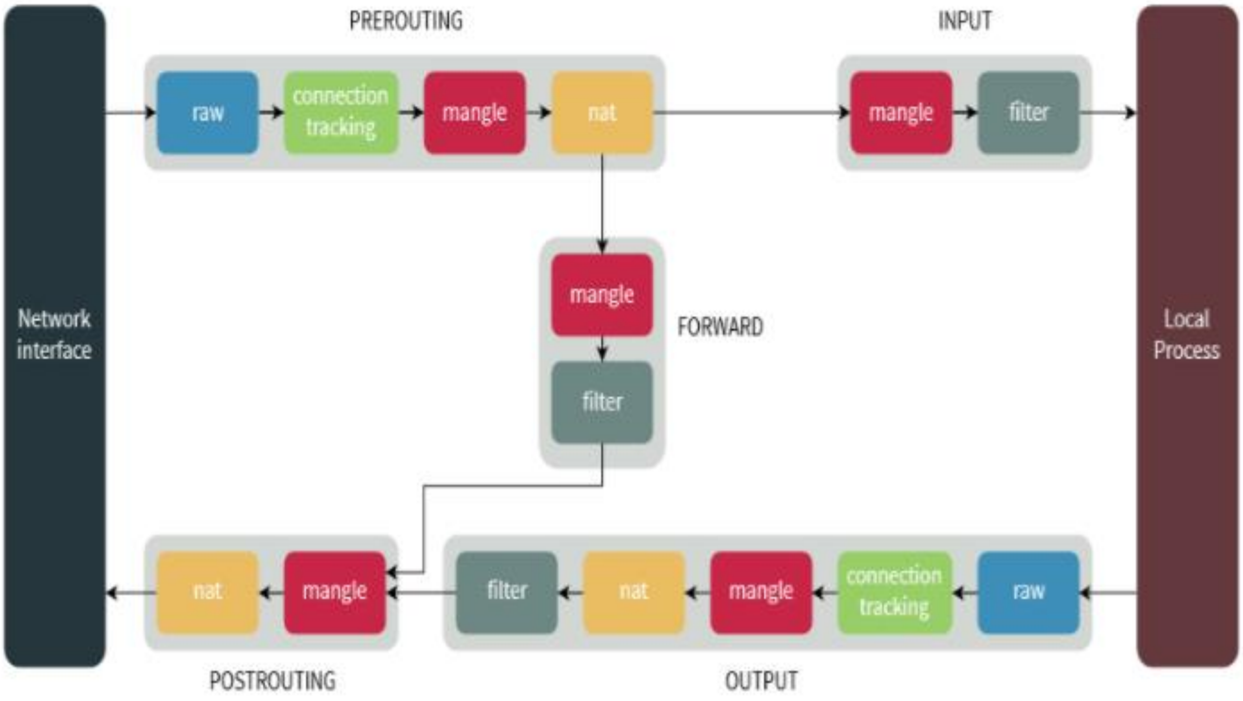


Figure 4. Overall diagram

4. Configuring iptables

Persistent rules

- On recent Red Hat systems, you can find the iptables rules stored in /etc/sysconfig/iptables. You can determine which runlevels have iptables enabled by running the command:

```
chkconfig --list iptables
```

- You can enable iptables for runlevels 3, 4, and 5 by running the command:

```
chkconfig --levels 345 iptables on
```

- You can start iptables manually by running:

```
service iptables start
```

- You can stop it with:

```
service iptables stop
```

Other configuration files

Path	Purpose
/etc/sysctl.conf	Contains settings for configurations in the /proc/sys directory that is applied at boot time
/proc/net/ip_conntrack	Dumps the contents of the connection tracking structures if you read it.
/proc/sys/net/ipv4/ip_conntrack_max	Controls the size of the connection tracking table in the kernel
/proc/sys/net/ipv4/ip_forward	forwarding packets among the networks connected to its interfaces

Compiling on kernel

CONFIG	Purpose
CONFIG_PACKET	direct communication with network interfaces
CONFIG_NETFILTER	the basic kernel support required by iptables
CONFIG_IP_NF_CONNTRACK	required for NAT and masquerading
CONFIG_IP_NF_FILTER	adds the filter table
CONFIG_IP_NF_IPTABLES	the basic support for user space iptables utility
CONFIG_IP_NF_MANGLE	adds the mangle table
CONFIG_IP_NF_NAT	adds the nat table

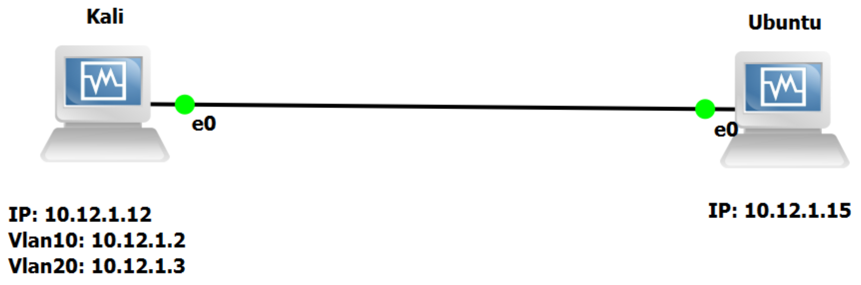
5. Tools of the Trade

Tool	Description
ethereal	Network protocol analyzer.
Nessus	Remote security scanner
nmap	Network mapper.
ntop	Network traffic probe
ping	Send ICMP ECHO_REQUEST to specific hosts
tcpdump	Packet capture and dumping
traceroute	Print the route packets take to a specific host.

6. Example

Prepare

GNS3:



Ubuntu

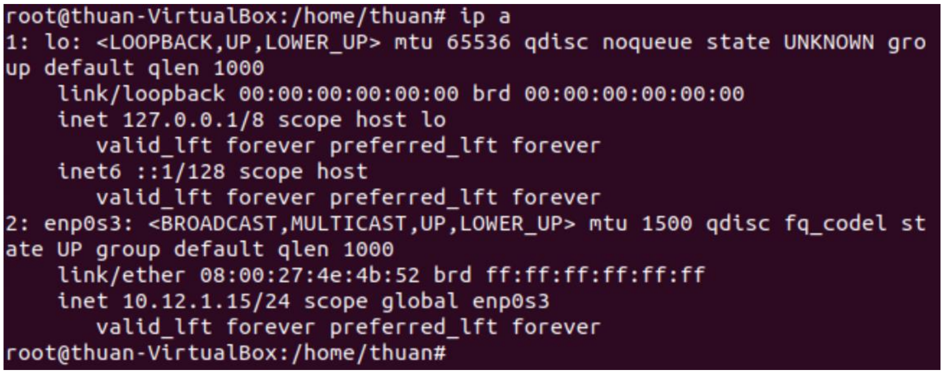
# Set ip for interface enp0s3

```
$ sudo ip addr add 10.12.1.15/24 dev enp0s3
```

# Delete all rules

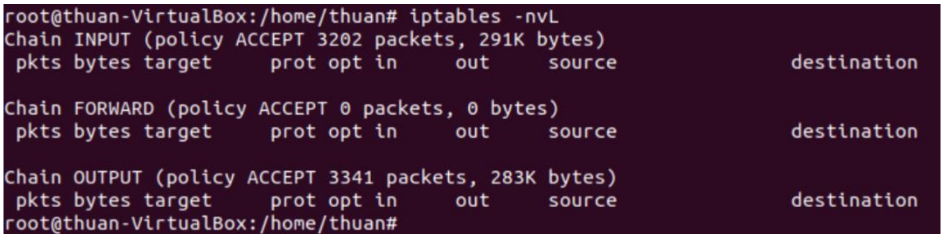
```
$ sudo iptables -t nat -F
$ sudo iptables -t mangle -F
$ sudo iptables -F
$ sudo iptables -X
```

# Show IP address



```
root@thuan-VirtualBox:/home/thuan# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gro
up default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether 08:00:27:4e:4b:52 brd ff:ff:ff:ff:ff:ff
    inet 10.12.1.15/24 scope global enp0s3
       valid_lft forever preferred_lft forever
root@thuan-VirtualBox:/home/thuan#
```

# Show firewall rules



```
root@thuan-VirtualBox:/home/thuan# iptables -nvL
Chain INPUT (policy ACCEPT 3202 packets, 291K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 3341 packets, 283K bytes)
 pkts bytes target     prot opt in     out     source               destination
root@thuan-VirtualBox:/home/thuan#
```

Kali

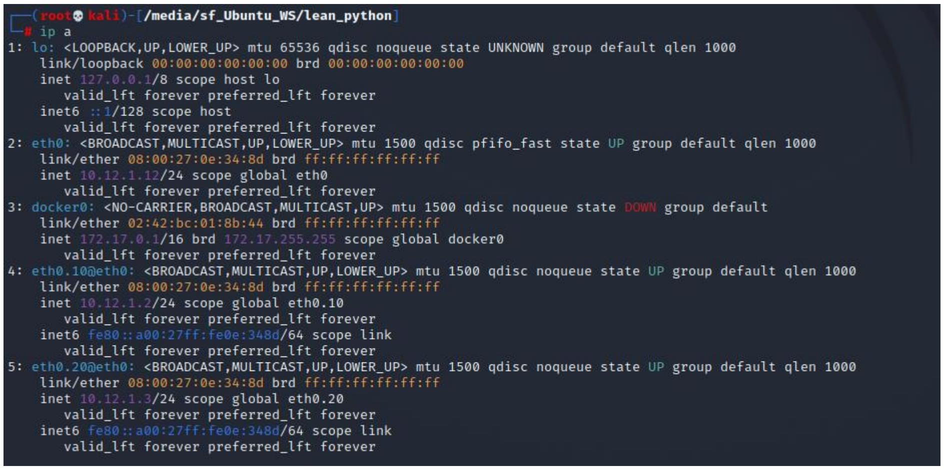
# Set ip for interface enp0s3

```
$ sudo ip addr add 10.12.1.12/24 dev eth0
```

# Config VLAN tag

```
$ sudo apt-get install vlan
$ sudo modprobe 8021q
$ sudo vconfig add eth0 10
$ sudo vconfig add eth0 20
$ sudo ip addr add 10.12.1.2/24 dev eth0.10
$ sudo ip addr add 10.12.1.3/24 dev eth0.20
$ sudo ifconfig eth0.10 up
$ sudo ifconfig eth0.20 up
$ sudo su -c 'echo "8021q" >> /etc/modules'
$ sudo systemctl restart networking
```

# Show IP address

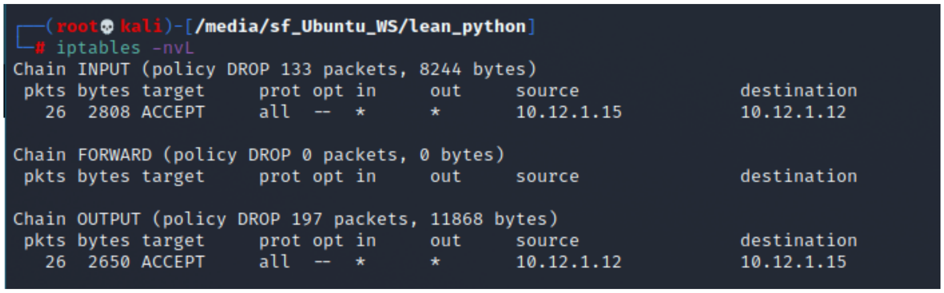


```
┌──(root💀 kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:0e:34:8d brd ff:ff:ff:ff:ff:ff
    inet 10.12.1.12/24 scope global eth0
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:bc:01:8b:44 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: eth0.10@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 08:00:27:0e:34:8d brd ff:ff:ff:ff:ff:ff
    inet 10.12.1.2/24 scope global eth0.10
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe0e:348d/64 scope link
       valid_lft forever preferred_lft forever
5: eth0.20@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 08:00:27:0e:34:8d brd ff:ff:ff:ff:ff:ff
    inet 10.12.1.3/24 scope global eth0.20
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe0e:348d/64 scope link
       valid_lft forever preferred_lft forever
```

# Firewall rules setup

```
sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -F
sudo iptables -X

iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP

iptables -I OUTPUT -d 10.12.1.15 -s 10.12.1.12 -j ACCEPT
iptables -I INPUT -d 10.12.1.12 -s 10.12.1.15 -j ACCEPT
```

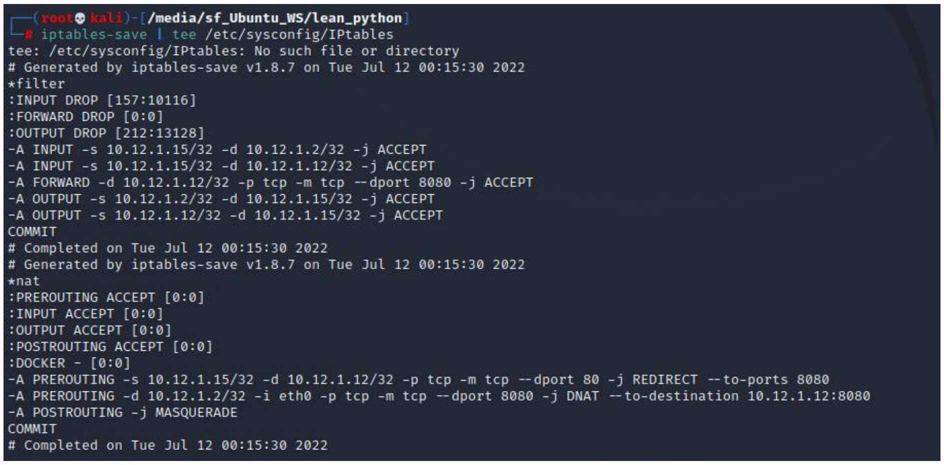
# Show firewall rules



```
┌──(root💀 kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# iptables -nvL
Chain INPUT (policy DROP 133 packets, 8244 bytes)
 pkts bytes target     prot opt in     out     source               destination
   26  2808 ACCEPT     all  --  *      *       10.12.1.15           10.12.1.12

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 197 packets, 11868 bytes)
 pkts bytes target     prot opt in     out     source               destination
   26  2650 ACCEPT     all  --  *      *       10.12.1.12           10.12.1.15
```

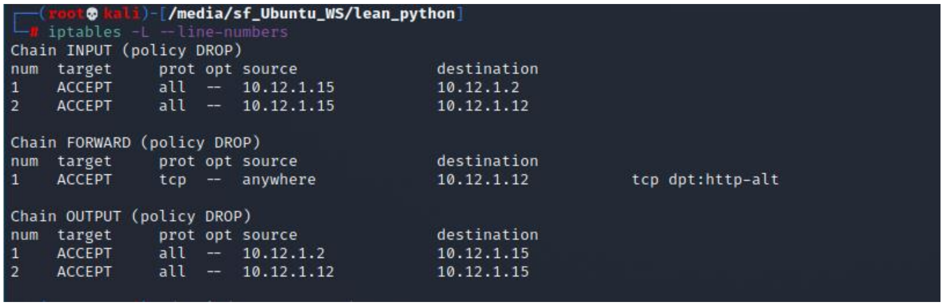
Display firewall rules

- iptables-save | tee /etc/sysconfig/IPtables



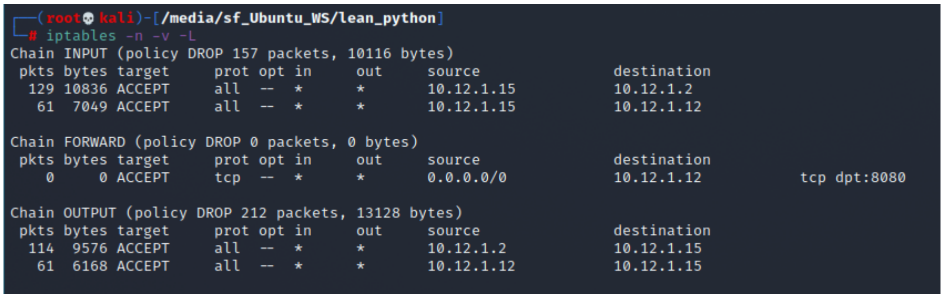
```
┌──(root💀 kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# iptables-save | tee /etc/sysconfig/IPtables
tee: /etc/sysconfig/IPtables: No such file or directory
# Generated by iptables-save v1.8.7 on Tue Jul 12 00:15:30 2022
*filter
:INPUT DROP [157:10116]
:FORWARD DROP [0:0]
:OUTPUT DROP [212:13128]
-A INPUT -s 10.12.1.15/32 -d 10.12.1.2/32 -j ACCEPT
-A INPUT -s 10.12.1.15/32 -d 10.12.1.12/32 -j ACCEPT
-A FORWARD -d 10.12.1.12/32 -p tcp -m tcp --dport 8080 -j ACCEPT
-A OUTPUT -s 10.12.1.2/32 -d 10.12.1.15/32 -j ACCEPT
-A OUTPUT -s 10.12.1.12/32 -d 10.12.1.15/32 -j ACCEPT
COMMIT
# Completed on Tue Jul 12 00:15:30 2022
# Generated by iptables-save v1.8.7 on Tue Jul 12 00:15:30 2022
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:DOCKER - [0:0]
-A PREROUTING -s 10.12.1.15/32 -d 10.12.1.12/32 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8080
-A PREROUTING -d 10.12.1.2/32 -i eth0 -p tcp -m tcp --dport 8080 -j DNAT --to-destination 10.12.1.12:8080
-A POSTROUTING -j MASQUERADE
COMMIT
# Completed on Tue Jul 12 00:15:30 2022
```

- iptables -L --line-numbers



```
┌──(root💀 kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# iptables -L --line-numbers
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  10.12.1.15           10.12.1.2
2    ACCEPT     all  --  10.12.1.15           10.12.1.12

Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             10.12.1.12           tcp dpt:http-alt

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  10.12.1.2            10.12.1.15
2    ACCEPT     all  --  10.12.1.12           10.12.1.15
```

- iptables -n -v -L



```
┌──(root💀 kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# iptables -n -v -L
Chain INPUT (policy DROP 157 packets, 10116 bytes)
 pkts bytes target     prot opt in     out     source               destination
  129 10836 ACCEPT     all  --  *      *       10.12.1.15           10.12.1.2
   61  7049 ACCEPT     all  --  *      *       10.12.1.15           10.12.1.12

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            10.12.1.12           tcp dpt:8080

Chain OUTPUT (policy DROP 212 packets, 13128 bytes)
 pkts bytes target     prot opt in     out     source               destination
  114  9576 ACCEPT     all  --  *      *       10.12.1.2            10.12.1.15
   61  6168 ACCEPT     all  --  *      *       10.12.1.12           10.12.1.15
```

TEE Target

It will clone a packet and redirect this clone to another machine on the local subnet
It's used for traffic mirroring
Ex:

#linux1: 10.12.1.2
#linux2: 10.12.1.12
#ubuntu: 10.12.1.15

```
root@linux1: iptables -A INPUT -p icmp --icmp-type echo-request -j TEE -gateway 10.12.1.12
root@linux2: sudo tcpdump icmp -n
ubuntu: ping 10.12.1.2
```

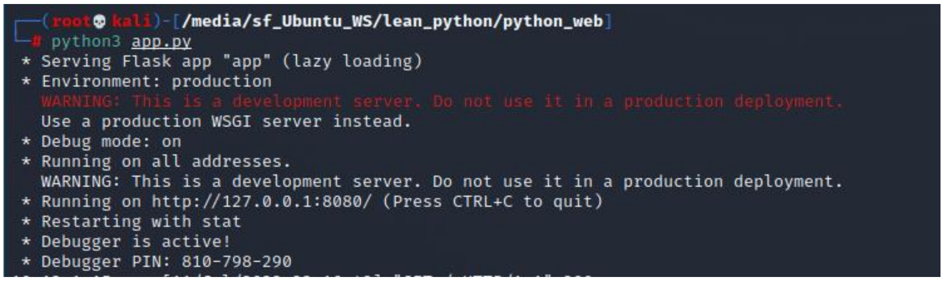
=> in linux2 will capture the packet from window to linux1

REDIRECT

# Prepare python web server

python webserver

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(host="0.0.0.0",port="8080",debug=True)
```



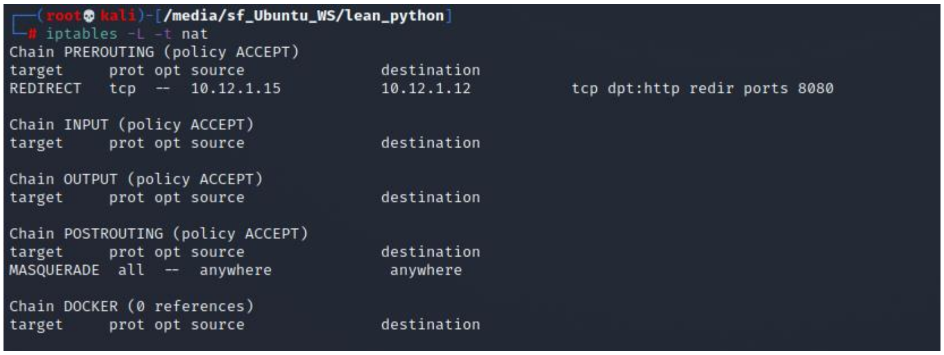
```
┌──(root💀kali)-[/media/sf_Ubuntu_WS/lean_python/python_web]
└─# python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 810-798-290
```

# Config firewall rule

Redirect rule

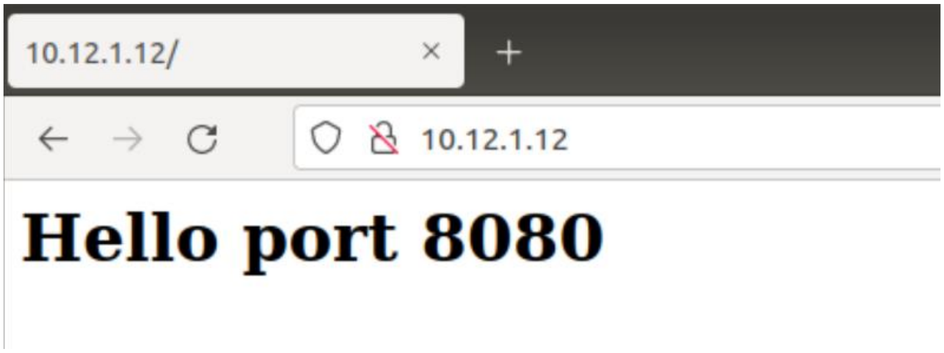
```
$ iptables -t nat -I PREROUTING -d 10.12.1.12 -s 10.12.1.15 -p tcp --dport 80 -j REDIRECT --to 8080
```

# Check firewall rules



```
┌──(root💀kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  10.12.1.15           10.12.1.12           tcp dpt:http redir ports 8080

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere

Chain DOCKER (0 references)
target     prot opt source               destination
```

# Try to connect from Ubuntu machine with **port 80** and it auto redirect to **port 8080**



Hello port 8080

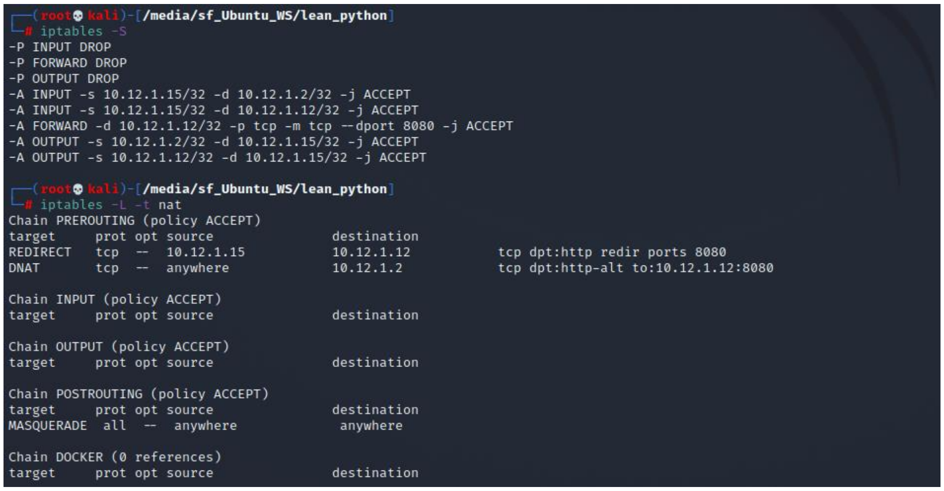
FORWARD

# Config firewall rule

Forward config

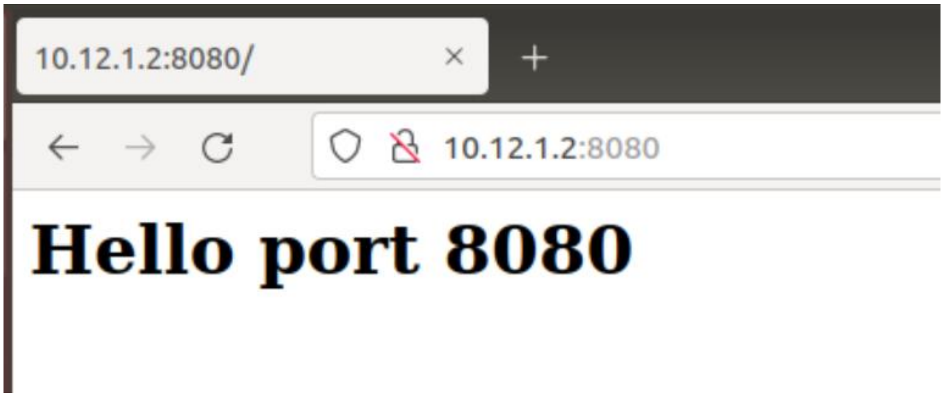
$ iptables -I OUTPUT -d 10.12.1.15 -s 10.12.1.2 -j ACCEPT
$ iptables -I INPUT -d 10.12.1.2 -s 10.12.1.15 -j ACCEPT
$ iptables -A PREROUTING -t nat -i eth0 -d 10.12.1.2 -p tcp --dport 8080 -j DNAT --to-destination 10.12.1.12:8080
$ iptables -A FORWARD -p tcp -d 10.12.1.12 --dport 8080 -j ACCEPT

# Check firewall rules



```
┌──(root💀kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# iptables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT DROP
-A INPUT -s 10.12.1.15/32 -d 10.12.1.2/32 -j ACCEPT
-A INPUT -s 10.12.1.15/32 -d 10.12.1.12/32 -j ACCEPT
-A FORWARD -d 10.12.1.12/32 -p tcp -m tcp --dport 8080 -j ACCEPT
-A OUTPUT -s 10.12.1.2/32 -d 10.12.1.15/32 -j ACCEPT
-A OUTPUT -s 10.12.1.12/32 -d 10.12.1.15/32 -j ACCEPT

┌──(root💀kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  10.12.1.15           10.12.1.12           tcp dpt:http redir ports 8080
DNAT       tcp  --  anywhere             10.12.1.2            tcp dpt:http-alt to:10.12.1.12:8080

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere

Chain DOCKER (0 references)
target     prot opt source               destination
```

# Try to connect from Ubuntu machine with **10.12.1.2:8080** and it auto forward to **10.12.1.12:8080**



10.12.1.2:8080/

10.12.1.2:8080

Hello port 8080

DNAT/SNAT FORWARD

# Config firewall rule

Kali firewall log config

```
$ sudo iptable --table nat \
> --append PREROUTING
> --protocol tcp \
> --destination 10.12.1.2 \
> --dport 80 \
> --jump DNAT
> --to-destination 10.12.1.12:8080
$ sudo iptable --table nat \
> --append POSTROUTING
> --protocol tcp \
> --destination 10.12.1.12 \
> --dport 8080 \
> --jump SNAT
> --to-source 10.12.1.2
```

Logging Packet

- LOG is a non-terminating target
- It logs detailed information about packet headers
- logs can be read with dmesg or from syslogd daemon
- LOG is used instead of DROP in the debugging phase

Option:

--log-prefix

--log-level

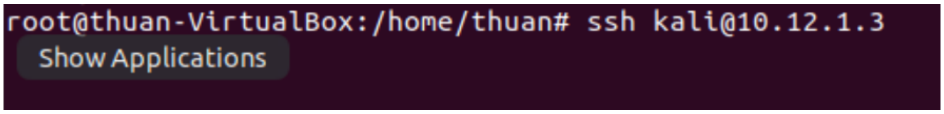
Test:

# Config firewall rule

Kali firewall log config

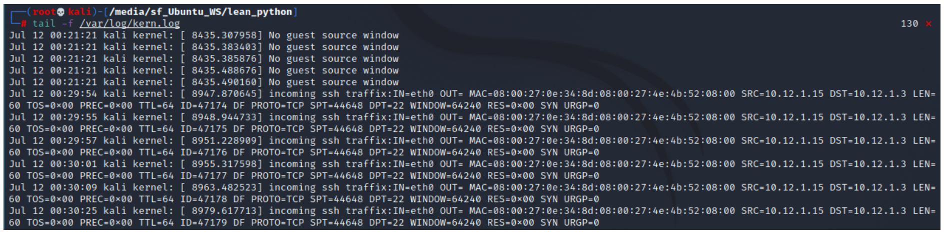
```
$ iptables -A INPUT -p tcp --dport 22 --syn -j LOG --log-prefix="incoming ssh traffix:" --log-level info
$ iptables -A INPUT -p tcp --dport 22 -j DROP
$ iptables -A OUTPUT -p tcp --dport 22 -j DROP
```

# 10.12.1.15 machine try to connect to 10.12.1.3:22



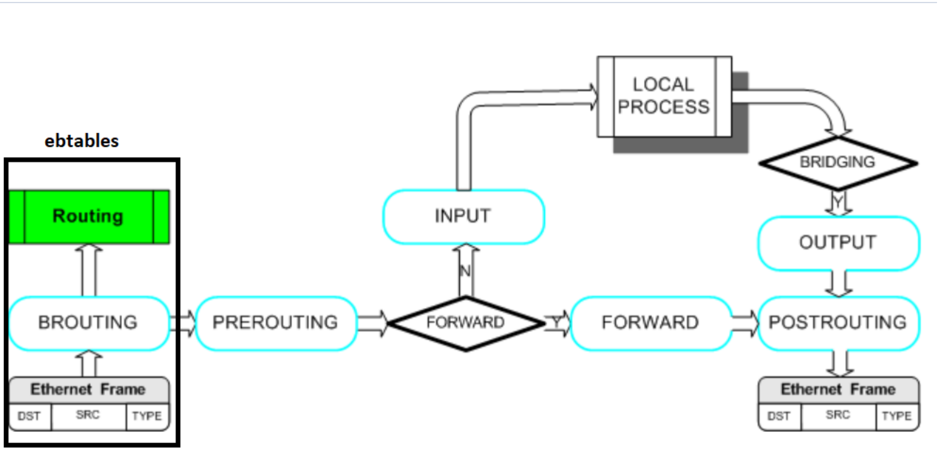
```
root@thuan-VirtualBox:/home/thuan# ssh kali@10.12.1.3
```

# 10.12.1.3 capture log file



```
┌──(root💀kali)-[/media/sf_Ubuntu_WS/lean_python]
└─# tail -f /var/log/kern.log                                                                     130 ✕
Jul 12 00:21:21 kali kernel: [ 8435.307958] No guest source window
Jul 12 00:21:21 kali kernel: [ 8435.383403] No guest source window
Jul 12 00:21:21 kali kernel: [ 8435.385876] No guest source window
Jul 12 00:21:21 kali kernel: [ 8435.488676] No guest source window
Jul 12 00:21:21 kali kernel: [ 8435.490160] No guest source window
Jul 12 00:29:54 kali kernel: [ 8947.870645] incoming ssh traffix:IN=eth0 OUT= MAC=08:00:27:0e:34:8d:08:00:27:4e:4b:52:08:00 SRC=10.12.1.15 DST=10.12.1.3 LEN=
60 TOS=0×00 PREC=0×00 TTL=64 ID=47174 DF PROTO=TCP SPT=44648 DPT=22 WINDOW=64240 RES=0×00 SYN URGP=0
Jul 12 00:29:55 kali kernel: [ 8948.944733] incoming ssh traffix:IN=eth0 OUT= MAC=08:00:27:0e:34:8d:08:00:27:4e:4b:52:08:00 SRC=10.12.1.15 DST=10.12.1.3 LEN=
60 TOS=0×00 PREC=0×00 TTL=64 ID=47175 DF PROTO=TCP SPT=44648 DPT=22 WINDOW=64240 RES=0×00 SYN URGP=0
Jul 12 00:29:57 kali kernel: [ 8951.228909] incoming ssh traffix:IN=eth0 OUT= MAC=08:00:27:0e:34:8d:08:00:27:4e:4b:52:08:00 SRC=10.12.1.15 DST=10.12.1.3 LEN=
60 TOS=0×00 PREC=0×00 TTL=64 ID=47176 DF PROTO=TCP SPT=44648 DPT=22 WINDOW=64240 RES=0×00 SYN URGP=0
Jul 12 00:30:01 kali kernel: [ 8955.317598] incoming ssh traffix:IN=eth0 OUT= MAC=08:00:27:0e:34:8d:08:00:27:4e:4b:52:08:00 SRC=10.12.1.15 DST=10.12.1.3 LEN=
60 TOS=0×00 PREC=0×00 TTL=64 ID=47177 DF PROTO=TCP SPT=44648 DPT=22 WINDOW=64240 RES=0×00 SYN URGP=0
Jul 12 00:30:09 kali kernel: [ 8963.482523] incoming ssh traffix:IN=eth0 OUT= MAC=08:00:27:0e:34:8d:08:00:27:4e:4b:52:08:00 SRC=10.12.1.15 DST=10.12.1.3 LEN=
60 TOS=0×00 PREC=0×00 TTL=64 ID=47178 DF PROTO=TCP SPT=44648 DPT=22 WINDOW=64240 RES=0×00 SYN URGP=0
Jul 12 00:30:25 kali kernel: [ 8979.617713] incoming ssh traffix:IN=eth0 OUT= MAC=08:00:27:0e:34:8d:08:00:27:4e:4b:52:08:00 SRC=10.12.1.15 DST=10.12.1.3 LEN=
60 TOS=0×00 PREC=0×00 TTL=64 ID=47179 DF PROTO=TCP SPT=44648 DPT=22 WINDOW=64240 RES=0×00 SYN URGP=0
```

7. Appendix:



B. Questions, Exercises

TBD

C. References

No.	Info	Link/ file/ name of ebook
1	Linux Iptables Pocket Reference	https://linuxbg.eu/books/Linux%20Iptables%20Pocket%20Reference.pdf
2	ebtables	http://ebtables.netfilter.org/br_fw_ia/br_fw_ia.html
3	A Deep Dive into Iptables and Netfilter Architecture	https://www.digitalocean.com/community/tutorials/a-deep-dive-into-iptables-and-netfilter-architecture#the-filter-table