UNIVERSITY OF ZAGREB
**FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING**

**SEMINAR**
**COMPUTER FORENSIC CLASS**

# iproute2 and iptables packet

*Neven Miculinić*

Zagreb, January 2018.

# CONTENTS

# 1. Introduction

Networking is one of the most important topic in everyday computer use. Almost every meaningful action we do in digital forencis, sooner or later involves networkings. Whether it's simply performing backups, viewing facebook messages, sending emails, or accessing

databases and utilizing VPN/SSH. As in any complex system, one surely describing networking, many things may go wrong and multiple attacks are possible. For the computer forensics purpose, this seminar describes bacis linux networking primities, from the time network packet enters the machine, reaches local process, and exits the machine.

It describes two tools consisting used most commonly in Linux networking. First is iptables, part of Netfilter project. Netfilter net is a framework providing various kernel hooks within network stack allowing user to modify and alter network packages. IPtables is their most commonly used utility. It shall be decribed in more detail in following chapters.

Iproute she is a collection of userspace utilities for controlling and monitoring various aspects of networking in the Linux kernel, including routing, network interfaces, tunnels, traffic control, and network-related device drivers. In this seminar the focus in only on routing, and just a brief introduction and basic/most common commands.

# 2. iptables

This chapter desvribes packet path through various iptables tables and chains. The rest of the chapter is dedicated for explaining basic ip tables concept, with next chapter showing various application. Visual overview can be seen in figure 2.1. This chapter is introducion in the iptables concept. For more information about syntax refer the manual page `man iptables`.

## 2.1.  Rule

Iptables have various rule, when matches their target is executed. They function as if-then construct. The most common rule 'ifs' are source/destination address, protocol and/or interface. They can be combined with and/or clauses. Furthermore any valid BPF BPF bytecode can be rule 'if'

The most common rule targets are ACCEPT, DROP, REJECT ones which perform packet filtering. In NAT table, common ones are DNAT, SNAT and MASQUERADE which perform IP:port NATing. NAT related targets are explained in section 4.1. Other common rule targets are LOG and jump to another chain. It will be explain in more detail in later chapters.
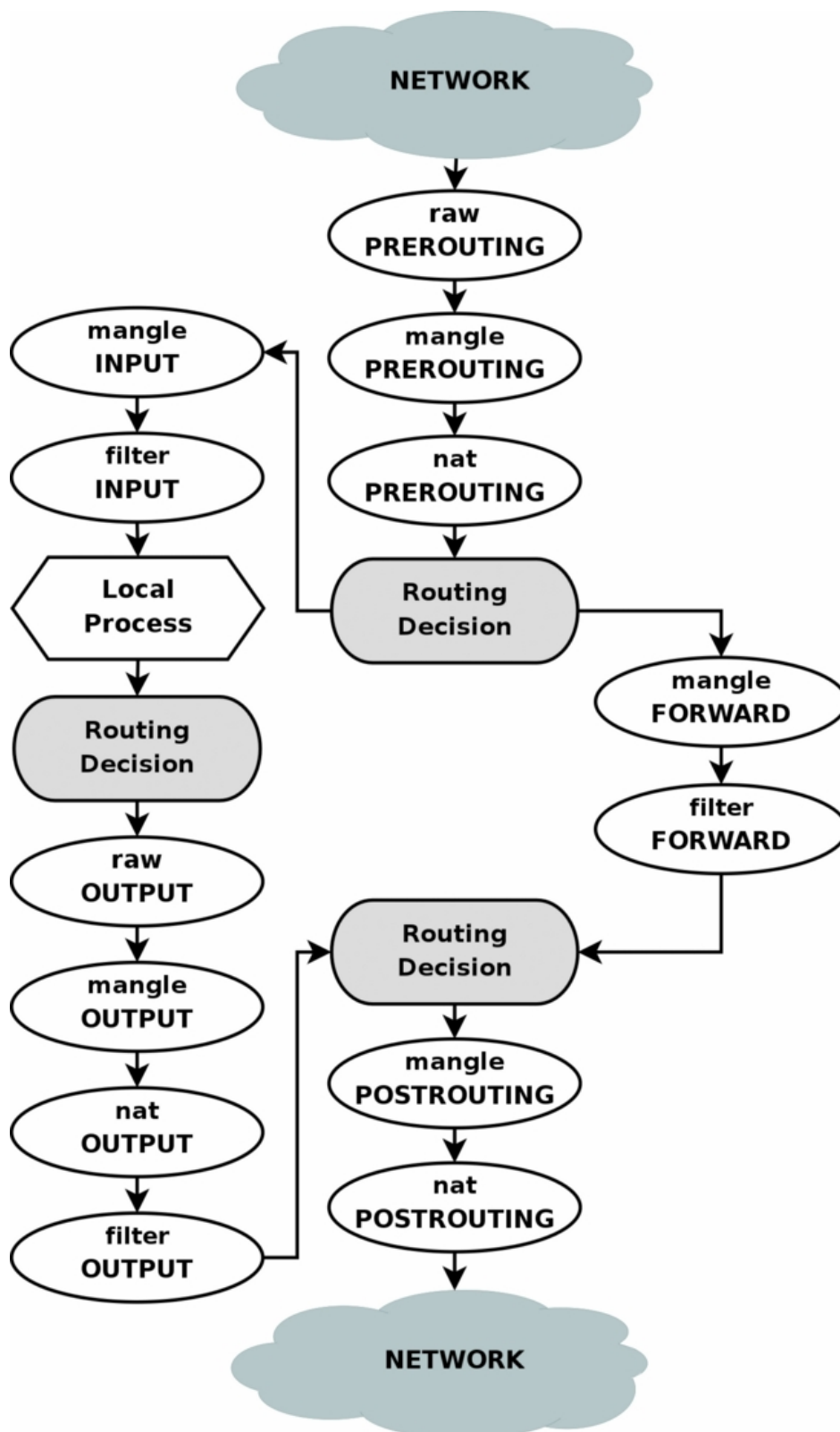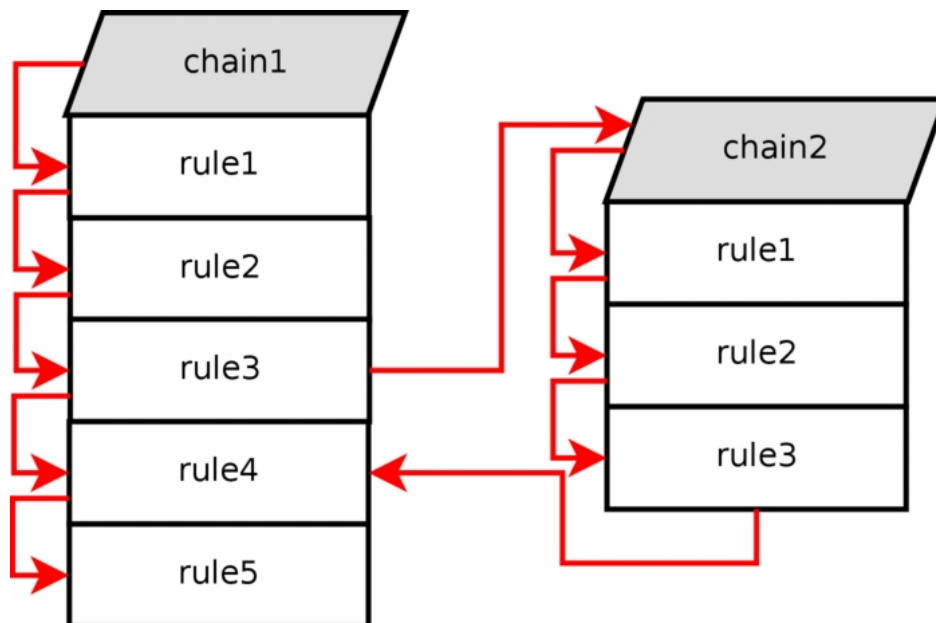
**Figure 2.1:** Overview of iptables. The lowercase word on top is the table and the upper case word below is the chain. Source Ipt

**Figure 2.2:** Table chain subtraverstion. Source Ipt

## 2.2. Chain

Chain is a list of rules which are matched in order. Rule can be terminal (most of them) or nonterminal (e.g. LOG, ULOG). Upon reaching the terminal rule (e.g. DROP) chain has reached its end. Chain can have it's default policy (e.g. DROP for table filter in INPUT chain). There are two types of chains – system (PREROURING, INPUT, FORWARD, OUTPUT, POSTROUTING) and user defined chains. User defined chains target jump within the same table (e.g. jump to user defined chain). They are created with `iptables -N <chain_name> -t <table name [filter default]>`. Chain traversal is depicted in figure 2.2. System chains are:

– PREROURING – Packet arriving in the kernel before any routing

– INPUT – Packet is destined for the local process

– FORWARD – Packet isn't destined for the local process

– OUTPUT – Packet originated from local process

– POSTROUTING – Packet departing from the machine after all routing takes place

Refer to figure 2.1 for their interaction.

## 2.3. Tables

Tables are bread and butter of this package. Each table defines specific hooks in the kernel for various system chains. Each table is composed of multipe chains, what system defined ones, what user defined chains. There are 5 tables:

- raw – applied before any connection tracking takes place. Used mostly to disable connection tracking since it's moderatly expensive. More information about connection tracking in subsection 2.4.1

- mangle – Mostly used for quality-of-service (QoS) header bit setting

- filter – packet filtering (DROP, ACCEPT and REJECT tagets)

- nat – NATing packages (DNAT, SNAT, MASQUERADE)

- security – packet marking (SECMARK, CONNSECMARK) for SELinux.

Refer to link **?** for more detail. To view all chains and rule in particular table use `iptables -vnL -t <t`

## 2.4.   Extensions

Iptables offers multiple modules you can use. You can view all installed modules by `ls -l /lib/iptables` and iptables will load all required modules dynamically. One of the most common ones is connection tracking.

### 2.4.1.   Conection tracking

If connection tracking is enabled (and can be disabled in raw TABLE with `-j NOTRACK` for rule match) each packet can be in following states:

- NEW – first packet of the connection

- ESTABLISHED – both server and client have sent a package

- RELATED – related connection to an enstablished one. Protocol specific (e.g. there's FTP, IRC, etc. support in the kernel for RELATED connection tracking)

- INVALID – connection state cannot be determined

# 3. Routing tables

iproute2 **??** is the moden Linux networking toolkit. It's installed by default on almost all Linux distributions, or available from package manager. This explains only the basic routing primitives and usages. For more details refer to the man page, or this cheetcheet **?**. All IP subnet examples shall use CIDR **?** notation. Also a brief remined about two network stack models. OSI:

- layer 7 – application layer

- layer 6 – presentation layer

- layer 5 – session layer

- layer 4 – transport layer (e.g. UDP, TCP)

- layer 3 – network layer (e.g. IP, ICMP, IPSec)

- layer 2 – data link layer (e.g. ARP, IEEE 802.3)

- layer 1 – physical layer (e.g. Bluetooth, IEEE 802.3, IEEE 802.11)

and internet protocol suite:

- Application layer (e.g. HTTP, DNS, IMAP)

- Transport layer (e.g. TCP, UDP)

- Internet layer (e.g. IP)

- Link layer (e.g. PPP, IEEE 802.11)

First basic comand is `ip route` which display current routes. On my computer the following is the display:

```
default via 192.168.121.1 dev wlp1s0 proto dhcp metric 600
172.17.0.0/16 dev docker0 proto kernel scope
    link src 172.17.0.1
172.18.0.0/16 dev br-8fd3cee01eeb proto kernel scope
    link src 172.18.0.1 linkdown
192.168.121.0/24 dev wlp1s0 proto kernel scope link
    src 192.168.121.194 metric 600
```

In the first colum are the destination addresses for which this rule matches. For matching the longest matching prefix is used –> that mean if we have two rules:

```
172.17.0.0/16 dev eth0 ...
172.17.0.0/24 dev eth1 ...
```

and we're sending the package to 172.17.0.1 it shall be routed via interface eth1 since it's the longest matching prefix. For more details on rule selectino algorithm read section 3.1

If no prefix is matches, (e.g. 35.12.23.44) the packed shall be routed via default rule. The first rule:

```
default via 192.168.121.1 dev wlp1s0 proto dhcp metric 600
```

Says send package to 192.186.121.1 over device wlp1s0. It shall find how to send to that device (using ARP – address resolution protocol) and send the package unchanged over layer 2 (from OSI layer model).

dev <X> says which interface is used to send packet. It's a layer 2 endpoint – etherner, WLAN, PPP or any other layer 2 protocol (or Link layer in the Internet protocol suite)

src <X> suggest to kernel which is the source IP address if the package is originating within this host. It's used by kernel address selection algorithm.

proto <X> says who/how the route got configured. `proto kernel` means during kernel autoconfiguration, and `proto dhcp` is using DHCP protocol.

metric <X> says about the route metric.

To add new route simply use `ip route add`, example:

```
ip route add 192.0.2.0/25 dev eth0
ip route add default dev eth0
ip route add 0.0.0.0/0 dev eth0
```

Last two lines command are equvivalent. To add route via a gateway use `ip route add ${addres` example `ip route add 192.0.2.128/25 via 192.0.2.1`

Similaraly for remove a route:

```
ip route delete 10.0.1.0/25 via 10.0.0.1
ip route delete default dev ppp0
```

## 3.1.  Route selection

Route selection is done in the following way. They are matched against this rules until only one possible route remains.

– longest matching prefix –> Find the route with the most specific prefix. 10.0.0.0/24 is more specific than 10.0.0.0/8, and both are matched for 10.0.0.1

– Lowest administrative distance –> Manually set routes have (`proto static`) have lower administrative distance then automatically set by protocols, e.g. `proto dhcp`

– Lowest metric

# 4. Example usecases

## 4.1. NAT

For NAT there are three specific targets related to NATing:

– SNAT – Source Network Address Translation. Exit packets source IP and port are rewriten to supplied source IP address. It is only valid in nat table within POSTROUTING chain. Downside is our source IP address must be known and static (or static range). Examples:

```
# Change source addresses to 1.2.3.4.
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4


# Change source addresses to 1.2.3.4, 1.2.3.5 or 1.2.3.6
iptables -t nat -A POSTROUTING -o eth0 -j SNAT \
    --to 1.2.3.4-1.2.3.6


# Change source addresses to 1.2.3.4, ports 1-1023
iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT \
    --to 1.2.3.4:1-1023
```

– DNAT – Destination Network Address Translation. It changes the package reciepeing, useful for servers behind firewall. It is only valid within nat table and PREROURING and OUTPUT chain. Examples:

```
iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 \
    -j DNAT --to-destination 192.168.1.1-192.168.1.10
```

– REDIRECT – DNAT but make destination local host. Only the port is changed.

```
iptables -t nat -A PREROURING -p tcp -o eth0 -j REDIRECT \
    --to-ports 1234
```

– MASQUERADE – it's similar to SNAT, but it doesn't require source IP address. It automatically grabs IP address information from sending interface. This is used in dymanically assigned IP connections. Example:

```
iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE
```

I've used in real life while configuring OpenVPN server on google cloud, since google cloud instances only allow outgoing trafic with the source IP equals intance IP.

## 4.2. Disabling internet access for specific device at specific time

For example you might have a really smart teen adicted to the internet. And you'd like disabling his internet access at the router level at certain times, while keeping rest functioning. It can be simly done with one iptables command and few extra modules

```
iptables -A PREROURING -m mac --mac-source 00:0F:EA:91:04:08 \
    -m time --timestart 9:00 --timestop 18:00 -j ACCEPT
iptables -A PREROURING -m mac --mac-source 00:0F:EA:91:04:08 \
    -j DROP
```

This is more efficient than IP filtering since you're probably running DHCP on your network dynamically assigning IP addresses. Nevertheless, it's easy for attacker (your teen) to figure out his MAC address is filterer, and to spoof it. Yet, hopefully by the time he figures it out, he'll already be a functional adult.

# 5. Conclusion

Linux networking is immensly broad topic. As such, this seminar is just a small overview of selected few topics of interest to the authour. This techniques are useful in preliminary

network debugging and forensics.

# 6. Bibliography

Bpf - the forgotten bytecode. `https://blog.cloudflare.com/bpf-the-forgotten-bytecode/`. (Accessed on 01/23/2018).

Iptables tutorial 1.2.2. `https://www.frozentux.net/iptables-tutorial/iptables-tutorial.html#TRAVERSINGOFTABLES`. (Accessed on 01/22/2018).

netfilter/iptables project homepage - the netfilter.org project. `http://www.netfilter.org/`. (Accessed on 01/23/2018).

shemminger/iproute2: Linux routing utilities. `https://github.com/shemminger/iproute2`. (Accessed on 01/23/2018).