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 topics in the everyday computer use. Almost every meaningful action we do in the digital world sooner or later involves networking. Whether it's simply performing backups, viewing facebook messages, sending emails, or accessing databases and utilizing VPN/SSH.

As in any complex system, networking included, many things may go haywire and multiple attacks are possible. China accidentally routed 15% of internet traffic on April 2010 [How]. Viruses are spread through the network. The huge throughput makes it a hard and challenging problem to tame, understand and navigate.

For the computer forensics purpose, this seminar describes basic Linux networking primitives, 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 a user to modify and alter network packages. IPtables is their most commonly used utility. It shall be described in more detail in the chapter 2.

Iproute2 [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 the routing, and just a brief introduction and basic/most common commands.

2. iptables

This chapter describes packet path through various iptables tables and chains. The rest of the chapter is dedicated to explaining basic IP tables concept A visual overview can be seen in figure 2.1. This chapter is introduction in the iptables concept. For more information about syntax refer the manual page `man iptables`.

2.1. Rule

The most basic construct is an if-then rule. It's if side can be matched on various attributes – packet's interface, protocol, source and destination IP address, or even custom BPF [BPF] can be used. Upon rule matching, their 'then' side is executed, called *target* within this context.

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 3.1. Other common rule

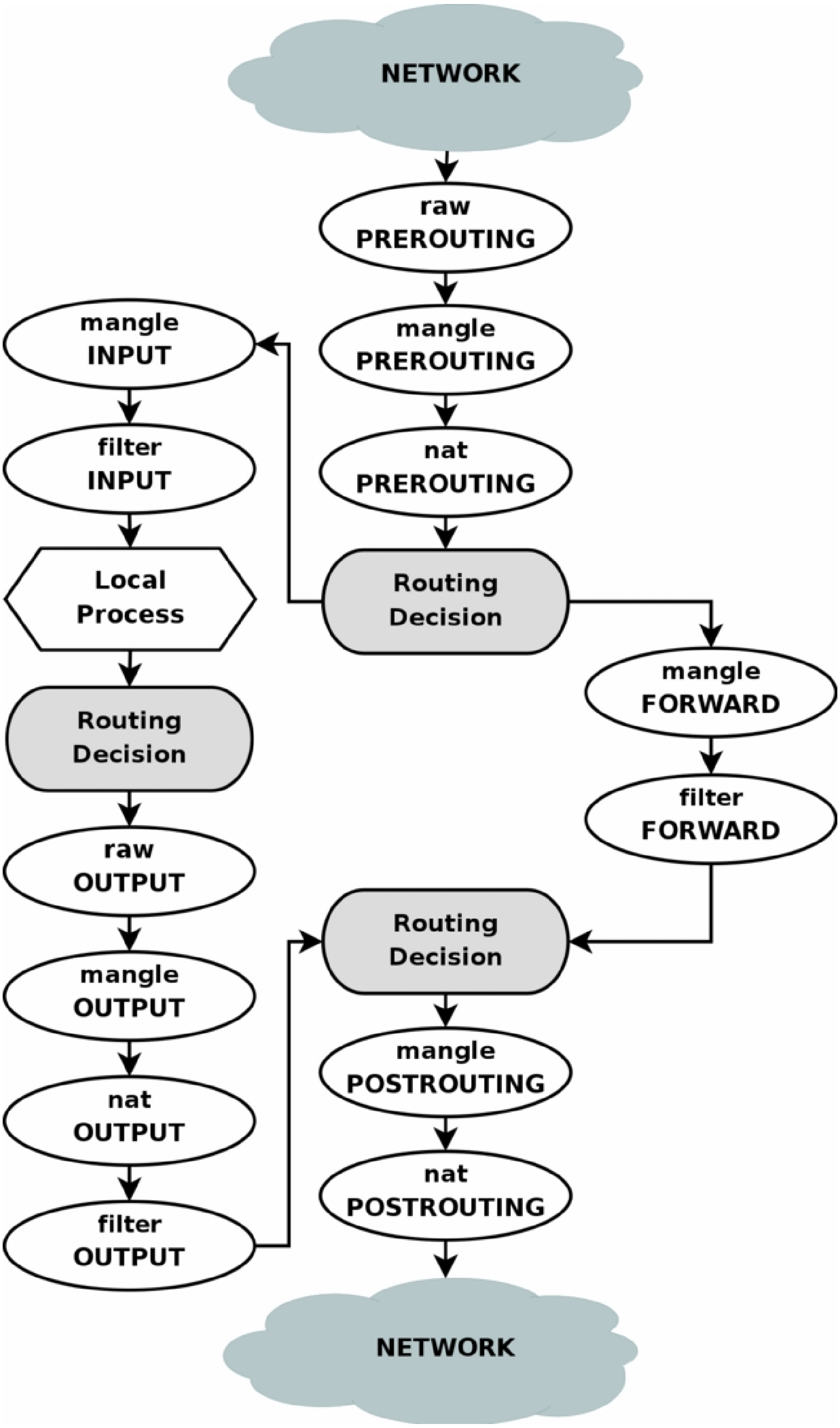

Figure 2.1: Overview of iptables. The lowercase word on top is the table and the upper case word below is the chain. Source [5]

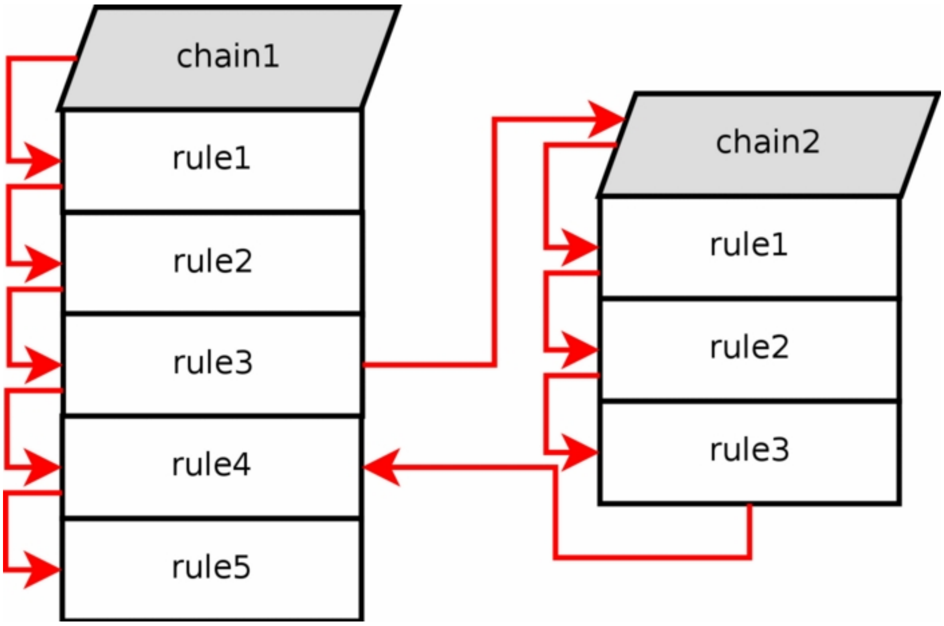

Figure 2.2: Table chain subtraverstion. Source [5]

targets are LOG and jump to another chain. Chain jumping is further explained in section 2.2.

2.2. Chain

A chain is a list of rules which are matched in order. A 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. The chain can have its default policy (e.g. DROP for table filter in INPUT chain). There are two types of chains – system (PREROUTING, 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:

- PREROUTING – 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 multiple 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 moderately 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 targets)
- nat – NATing packages (DNAT, SNAT, MASQUERADE)
- security – packet marking (SECMARK, CONNSECMARK) for SELinux. They are out of scope for this seminar.

Refer to link [6] for more detail. To view all chains and rule in particular table use `iptables -vnL -t <`

2.4. Extensions

Iptables offers multiple modules you can use. They extend the basic functionality. 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. For other refer to their manual pages and documentation.

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 established 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. IPtable examples

3.1. NAT

For NAT there are three specific targets related to NATing:

- SNAT – Source Network Address Translation. Exit packets source IP and port are rewritten to supplied source IP address. It is only valid in nat table within POSTROUTING chain. The 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 recipient, useful for servers behind firewall. It is only valid within nat table and PREROUTING 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 dynamically 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 traffic with the source IP equals instance IP.

3.2. Disabling internet access for specific device at specific time

For example, you might have a really smart teen addicted 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 simply 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 filtered, and to spoof it. Yet, hopefully, by the time he figures it out, he'll already be a functional adult.

4. Routing tables

iproute2 **??** is the modern 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 [ipr]. All IP subnet examples shall use CIDR [Cla] notation. Also a brief reminded 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 command is `ip route` which displays 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 column are the destination addresses for which this rule matches. For matching the longest matching prefix is used –> that means 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 selection algorithm read section 4.1

If no prefix 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 – ethernet, 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`

Similarly to remove a route:

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

4.1. 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

5. Conclusion

Linux networking is immensely broad topic. As such, this seminar is just a small overview of selected few topics of interest to the author. These techniques are useful in preliminary network debugging and forensics. The basic iptables configuration and routing have been covered. For further improvement in this topic reading manual pages is encouraged.

6. Bibliography

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

[Cla] Classless inter-domain routing - wikipedia. https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing?oldformat=true. (Accessed on 01/24/2018).

[How] How china swallowed 15% of ‘net traffic for 18 minutes | ars technica. https://arstechnica.com/information-technology/2010/11/how-china-swallowed-15-of-net-traffic-for-18-minutes/. (Accessed on 01/24/2018).

[ipr] iproute2 cheat sheet. http://baturin.org/docs/iproute2/. (Accessed on 01/24/2018).

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

[6] Iptables tutorial 1.2.2. https://www.frozentux.net/

iptables-tutorial/iptables-tutorial.html#TABLES. (Accessed on 01/23/2018).

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

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