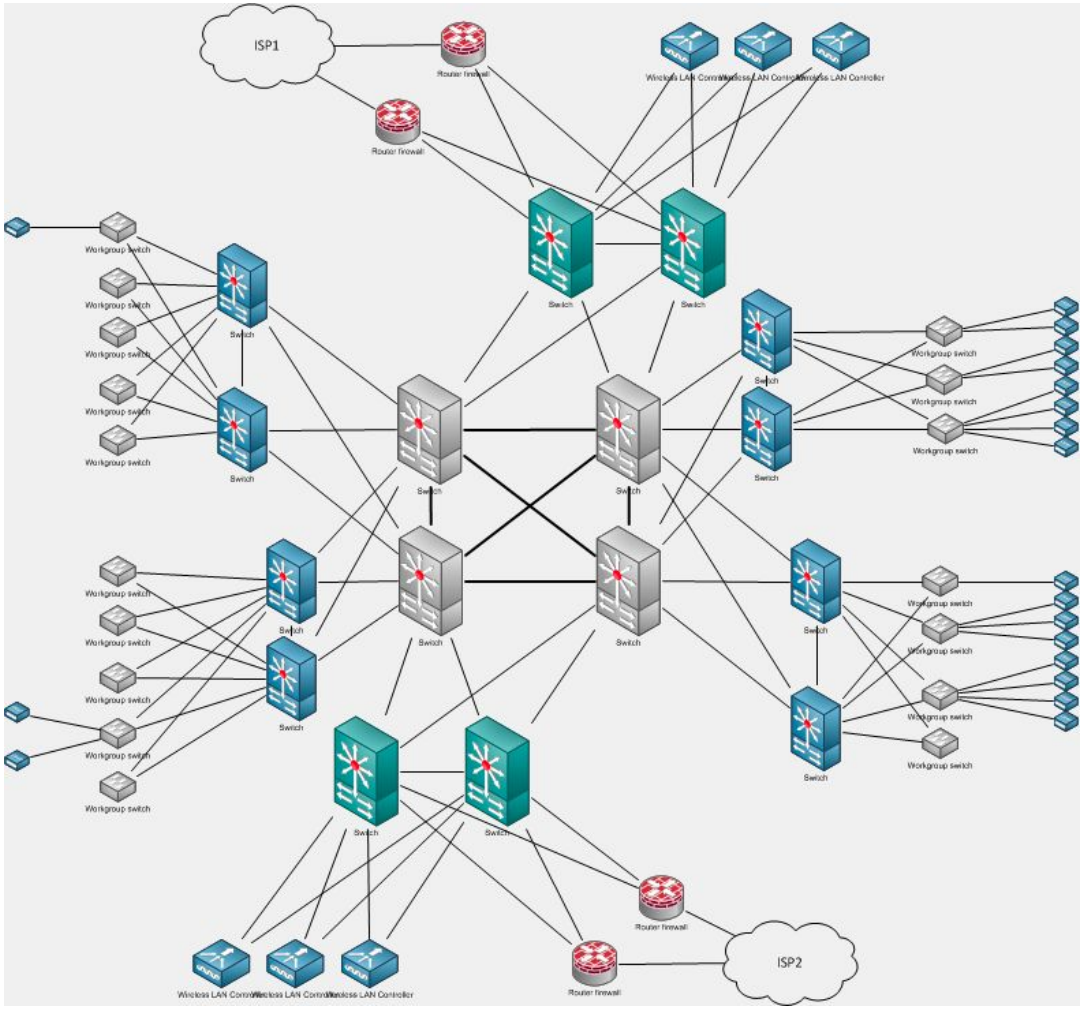
Firewalls

Security 1 2018-19

Università Ca' Foscari Venezia
www.dais.unive.it/~focardi
secgroup.dais.unive.it

Università
Ca'Foscari
Venezia

Networks are complex



(image from https://netcube.ru)

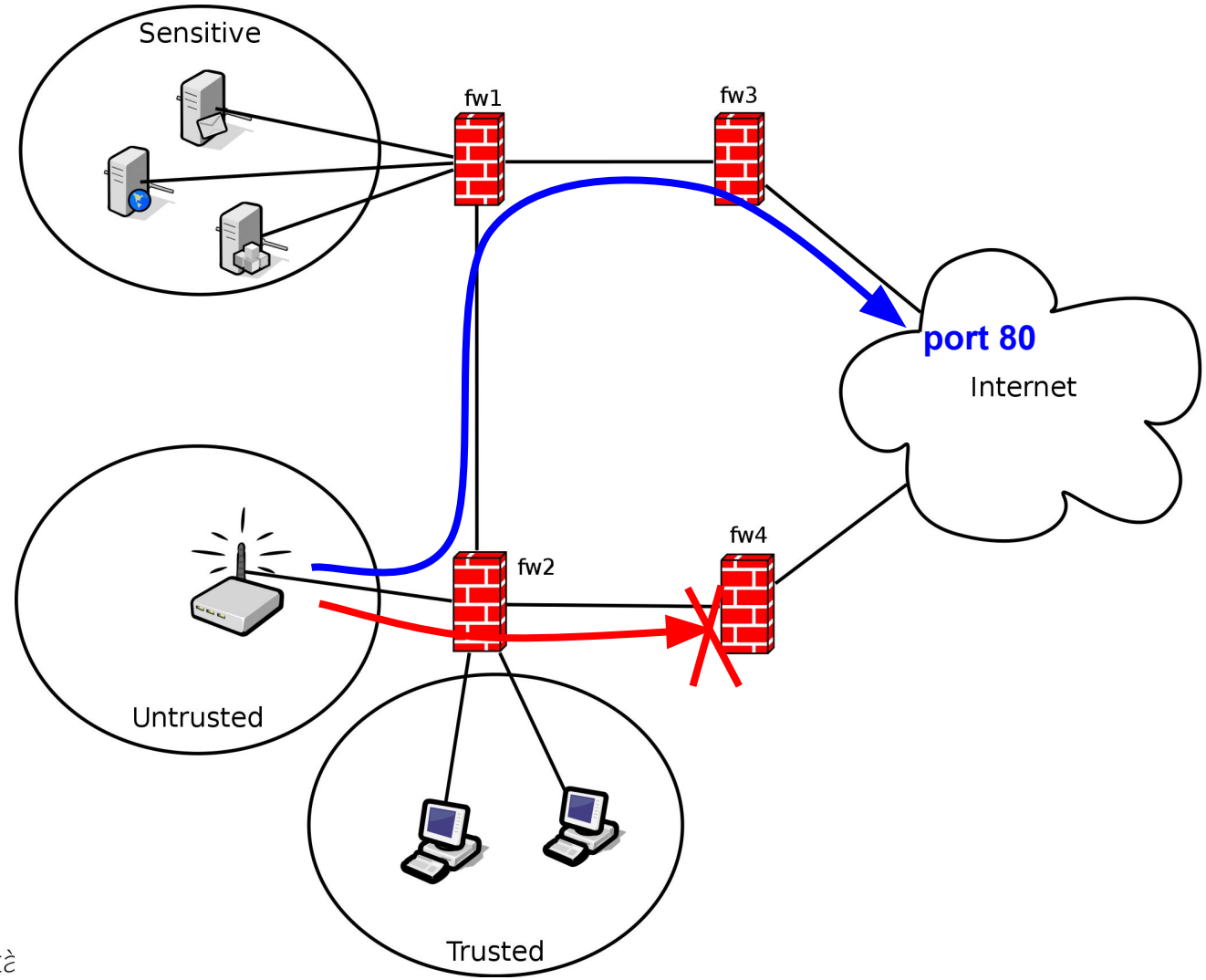
Example: *traversal control*

Three subnetworks: Sensitive, Trusted and Untrusted

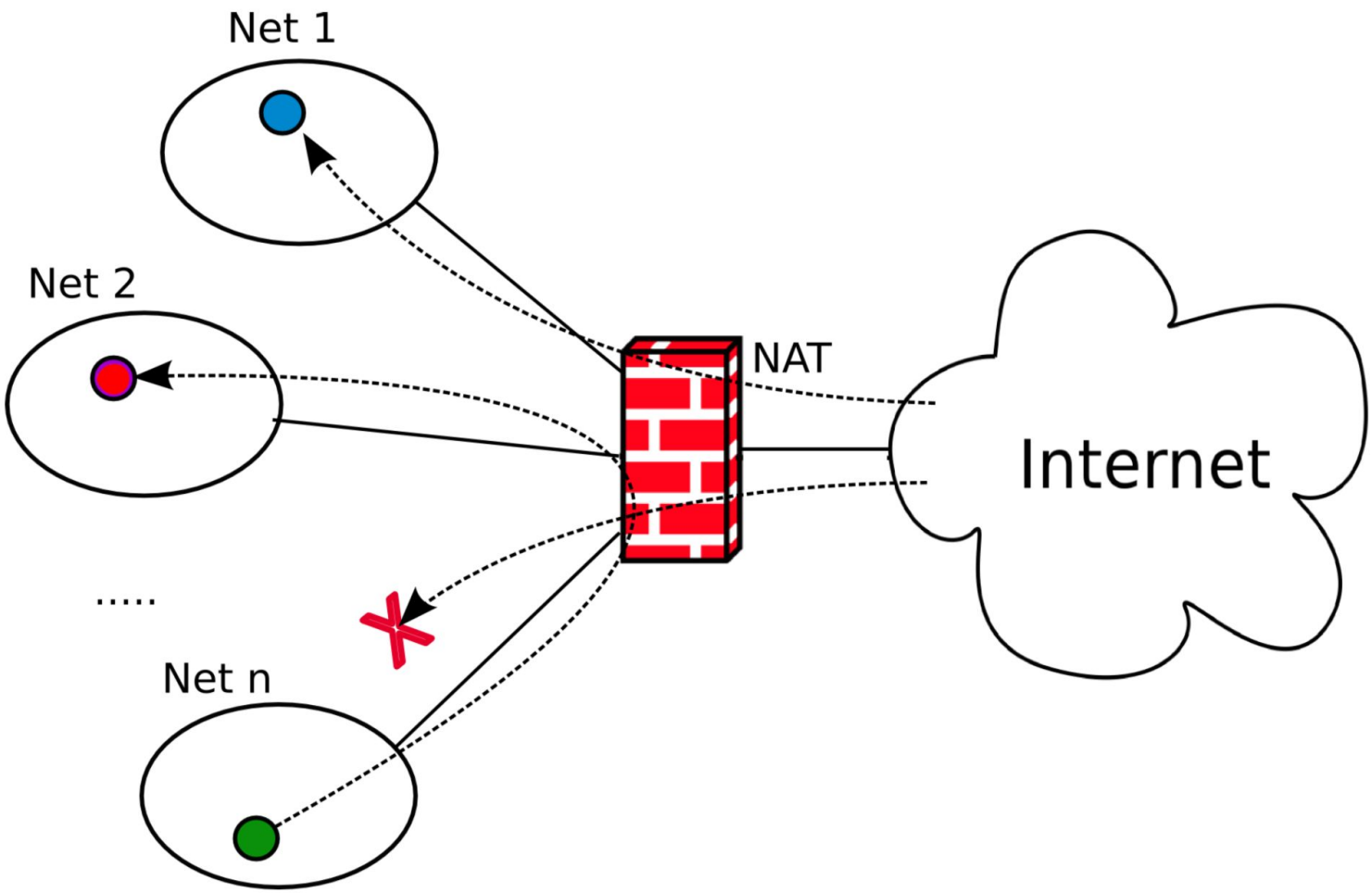
Untrusted network should reach the Internet only

1. **through fw3** and
2. when connecting to the Web (**port 80**)

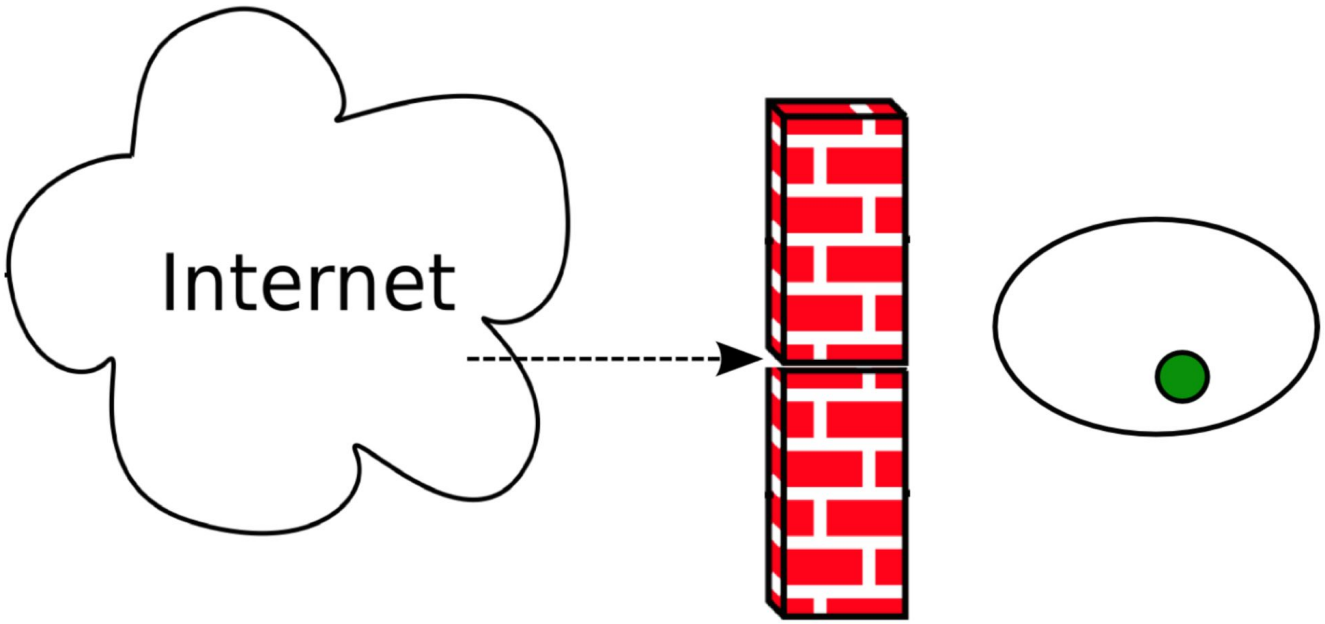
Example: *traversal control*



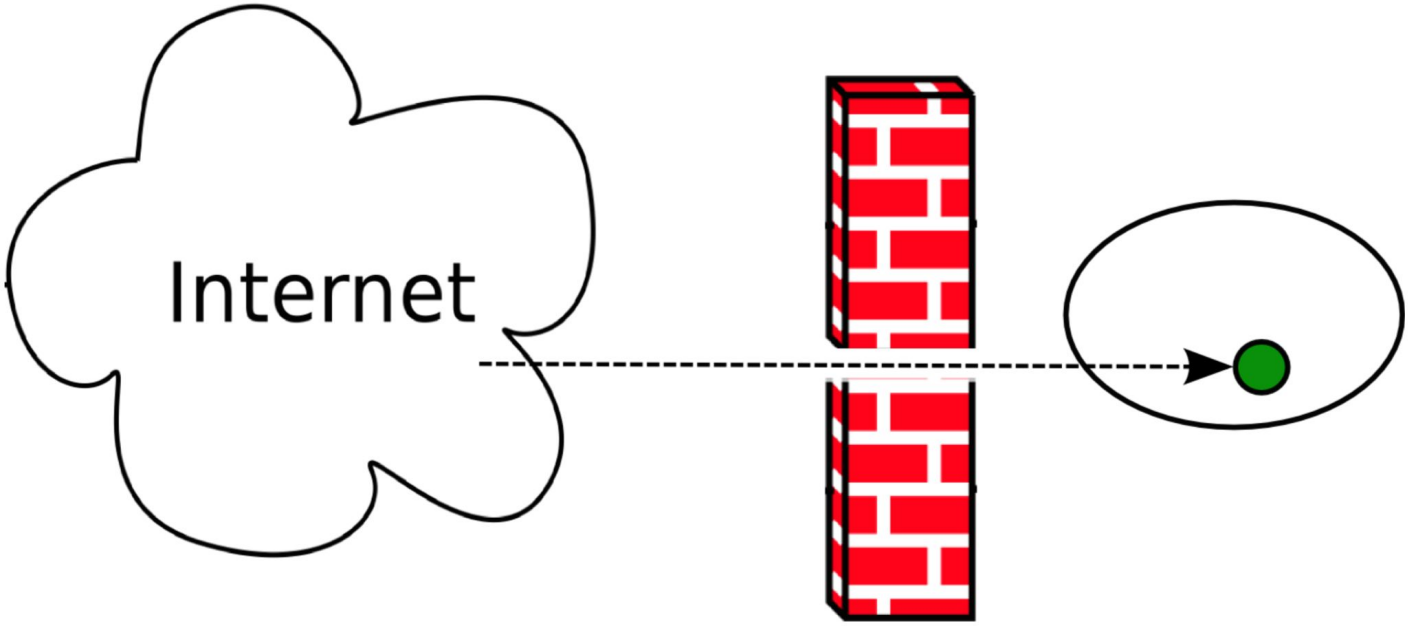
Typical firewall operations



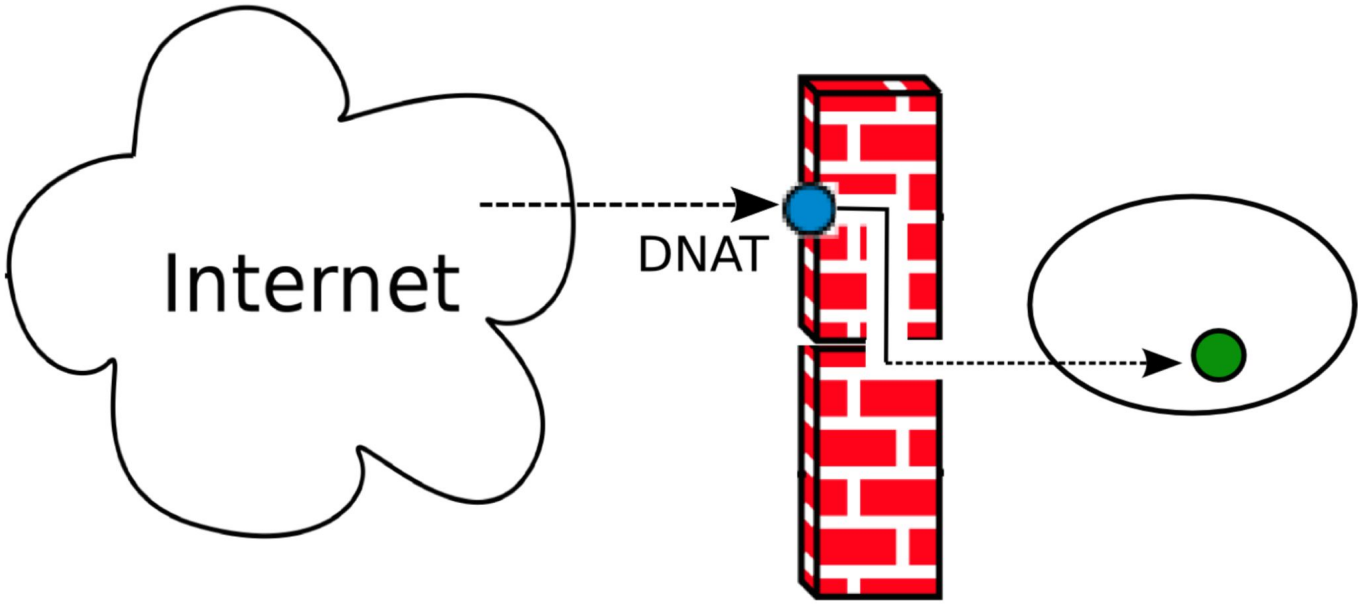
Drop



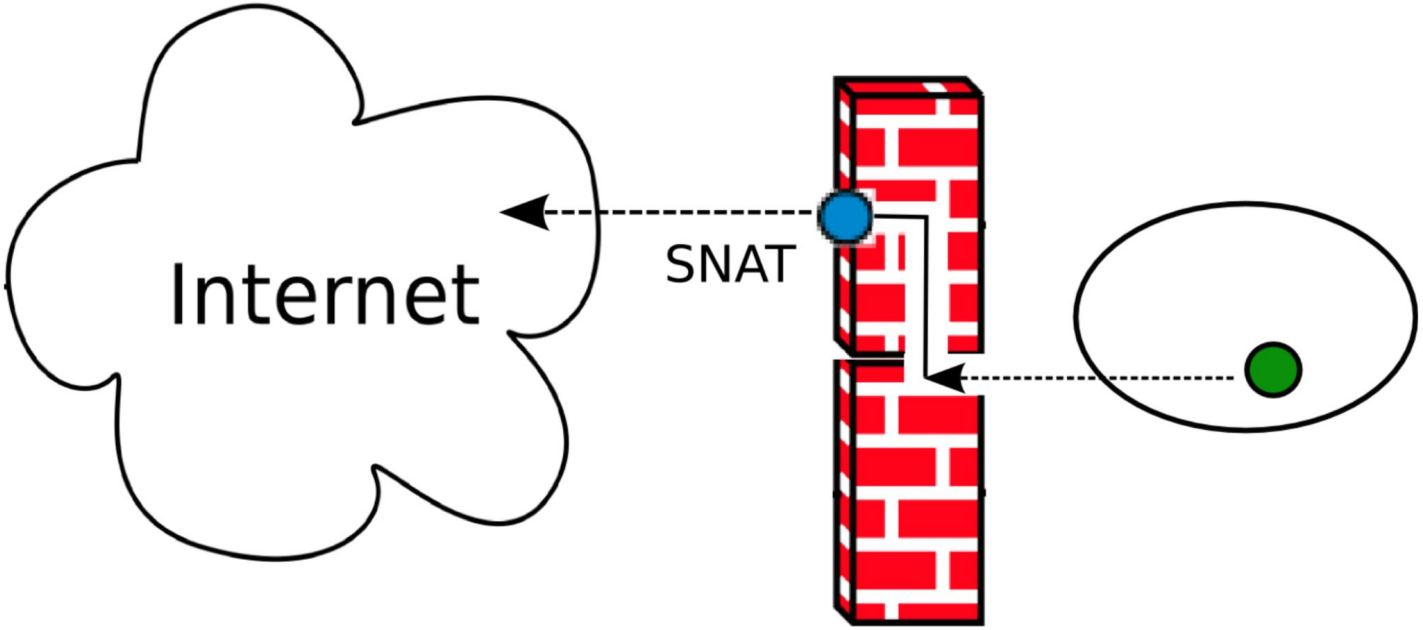
Accept



Destination NAT



Source NAT



netfilter

Standard firewall tool in Linux

Netfilter allows for:

- Packet **filtering**
- Network address **translation** (NAT)
- Packet **mangling** (packet transformation)

Netfilter can be configured through `iptables`, a very powerful and flexible tool

Tables

`netfilter` is based on *tables*

Tables group rules depending on the kind of *action*

The three most commonly used tables are:

- *filter* for packet filtering
- *nat* for NATs
- *mangle* for packet alteration

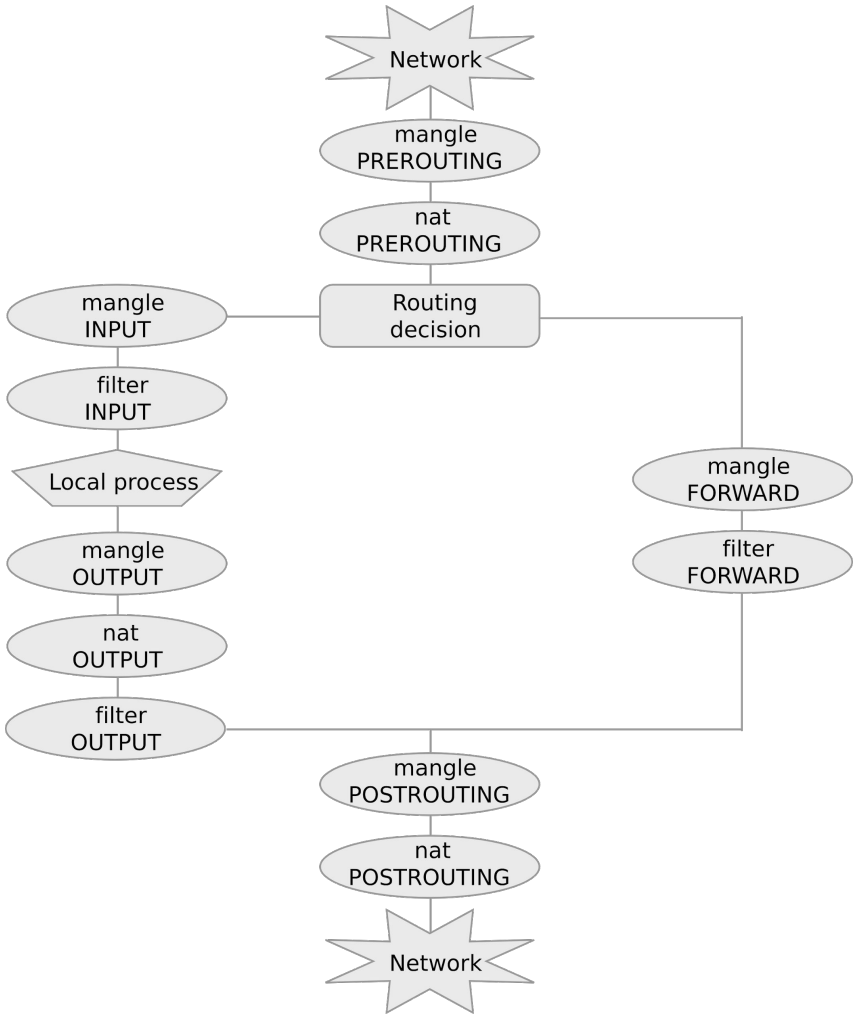
Chains

Chains are lists of rules that are **inspected sequentially**

There are *five predefined chains* that are inspected in specific moments of a packet (p) life cycle:

- `PREROUTING` when p **reaches** the host
- `FORWARD` when p is **routed through** the host;
- `POSTROUTING` when p is about to **leave** the host;
- `INPUT` when p is **routed to** the host
- `OUTPUT` when p is **generated** by the host

Tables and chains



Rules

Rules (in a chain) are inspected one after the other

- If matched then p is processed along the *rule target*
- Otherwise the next rule in the chain is examined.

The most commonly used targets are:

- `ACCEPT`, for **accepting** the packet
- `DROP`, for **dropping** it
- `DNAT`, for **destination NAT**
- `SNAT` for **source NAT**

A **default policy** is triggered if no rule matches

User-defined chains

Target can also be a **jump** into a different chain, defined by users

Similar to function calls

⇒ Powerful and flexible but configurations become often **complex** and **hard to maintain**

⇒ **Ordering** of rules (and tables) is fundamental to decide the "fate" of a packet

“Demotivating” example

```
iptables -P INPUT DROP

iptables -P FORWARD DROP

iptables -P OUTPUT DROP

iptables -t mangle -P PREROUTING DROP

iptables -A INPUT -i lo -j ACCEPT

iptables -A OUTPUT -j ACCEPT

iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT

iptables -A INPUT -s 0.0.0.0 -d 255.255.255.255 -j ACCEPT

iptables -t mangle -A PREROUTING -s 0.0.0.0 -d 255.255.255.255 -j ACCEPT

iptables -t mangle -A PREROUTING -i eth0 -s 10.0.0.0/24 -j ACCEPT iptables -t mangle -A PREROUTING -i lo -s 127.0.0.1 -j
ACCEPT

iptables -A FORWARD -i eth0 -d 5.6.7.8 -j DROP

iptables -t mangle -A PREROUTING -i eth1 -d 10.0.0.2 -p tcp --dport 22 -m state --state NEW -j DROP iptables -A FORWARD -i
eth1 -d 10.0.0.2 -p tcp --dport 22 -j ACCEPT

iptables -A FORWARD -s 10.0.0.2 -p tcp --sport 22 -o eth1 -m state --state ESTABLISHED -j ACCEPT iptables -t nat -A
PREROUTING -i eth1 -d 1.2.3.4 -p tcp --dport 22 -j DNAT --to-destination 10.0.0.2:22

iptables -t mangle -A PREROUTING -i eth1 -d 10.0.0.2 -p tcp --dport 80 -m state --state NEW -j DROP iptables -A FORWARD -i
eth1 -d 10.0.0.2 -p tcp --dport 80 -j ACCEPT

iptables -A FORWARD -s 10.0.0.2 -p tcp --sport 80 -o eth1 -m state --state ESTABLISHED -j ACCEPT iptables -t nat -A
PREROUTING -i eth1 -d 1.2.3.4 -p tcp --dport 80 -j DNAT --to-destination 10.0.0.2:80

iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT

iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED -j ACCEPT iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o
eth1 -j MASQUERADE
```

Default policy

```
# iptables -t filter -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
```

`-t` specifies the table (filter is the default)

`-L` stands for “list”

Configuring a firewall

Least privilege principle: set the default policy to DROP and only enable packets that are necessary

IMPORTANT: <u>do not cut you off</u>!

Remember to **enable ssh before** you change the default policy to DROP!

Enable ssh

```
iptables -A INPUT  -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
```

-A stands for append

-p tcp specifies tcp protocol

-dport and -sport specify destination and source port

-j ACCEPT specifies the ACCEPT target

Notice that **both directions** are necessary

Enable ssh

```
# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   57  3844 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh

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

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   25  2740 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp spt:ssh
```

Packets are matching the two rules
(we can observe this through -v option)

Least privilege

```
iptables -P INPUT DROP
```

Drops any incoming packet except SSH:

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  126  8632 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh
```

What happens if we issue the following?

```
iptables -A INPUT  -p tcp --dport 22 -j DROP
```

Rules are inspected sequentially: this rule will **never be matched**, since ssh packets will be accepted by the previous one!

Stateful firewall

Suppose we connect to a web server (port 80)

- Since OUTPUT policy is ACCEPT the connections goes through

Q: What happens to the server **answer**?

A: The **answer is dropped**!

⇒ Recall that we only admit ssh incoming connections!

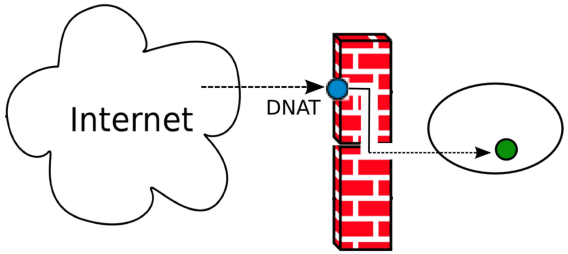
Connection tracking

`netfilter` tracks connections:

- when a new connection starts the packet has state **NEW**
- packets belonging to the same connection has state **ESTABLISHED**
- some protocols start new connections (e.g. ftp). These packets have state **RELATED**
- address translation is also tracked (see next slides)

```
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
```

DNAT example



Forward web-traffic to a local server

```
iptables -t nat -A PREROUTING -p tcp  -d 157.138.7.88 --dport 80
-j DNAT --to-destination 192.168.0.100:80
```

Packets to `157.138.7.88:80` go to `192.168.0.100:80` instead

Answers from `192.168.0.100` will appear to be from `157.138.7.88` (in order to be accepted by the browser)

⇒ Translation is applied to all packets on the **same connection**

Maintaining a configuration

```
1) iptables ... --source N1 -j ACCEPT
2) iptables ... --dport 80 -j DROP
3) iptables ... --source N2 -j ACCEPT
4) iptables ... --dport 22 -j DROP
5) iptables ... --source N3 -j ACCEPT
```

Q1: Where can the packets from N2 go?

A1: Everywhere except port 80 (context dependent)

Maintaining a configuration

```
1) iptables ... --source N1 -j ACCEPT
2) iptables ... --dport 80 -j DROP
3) iptables ... --source N2 -j ACCEPT
4) iptables ... --dport 22 -j DROP
5) iptables ... --source N3 -j ACCEPT
```

Q2: how can we accept packets from N4 that are not addressed to port 80?

A2:

```
3a) iptables ... --source N4 -j ACCEPT
```

Maintaining a configuration

```
1) iptables ... --source N1 -j ACCEPT
2) iptables ... --dport 80 -j DROP
3) iptables ... --source N2 -j ACCEPT
4) iptables ... --dport 22 -j DROP
5) iptables ... --source N3 -j ACCEPT
```

Q3: how can we accept packets from N5 that are not addressed to port 22?

A3:

```
1a)iptables ... --source N5 --dport 22 -j DROP
1b)iptables ... --source N5 -j ACCEPT
```

Maintaining a configuration is hard

- **no fixed structure**
- **order** matters, ACCEPT and DROP can alternate
- **semantics** depends on tables and chains
- to drop a packet, one must be sure that the rule is placed **before** all the ACCEPT rules (same chain)
- **policy** can be default ACCEPT or default DROP
- real configurations easily have more than **1000 lines**!
- real configurations grow over time, and are maintained by **different** systems administrators.

Mignis

Mignis is a tool for firewall configuration that supports:

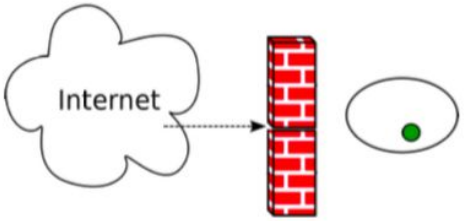
- Network Address Translation (NAT)
- Least privilege
- **Declarative** style: order of the rules is irrelevant!
- Explicit rejects (with precedence over “positive” rules)
- Simple formal semantics (proof of correctness)

Developed by secgroup@unive:

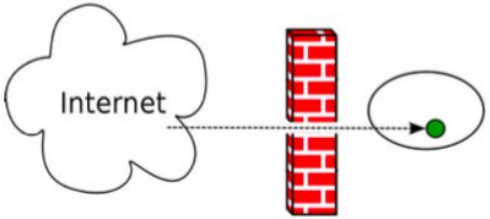
https://github.com/secgroup/Mignis

Mignis rules

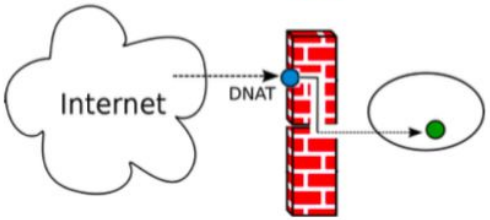
DROP: $\mathtt{n}_1 \ / \ \mathtt{n}_2 \mid \phi$



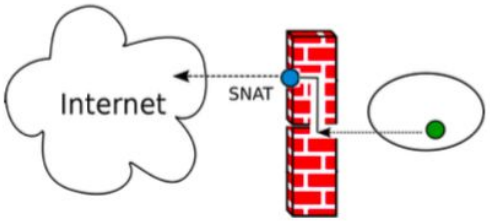
ACCEPT: $\mathtt{n}_1 > \mathtt{n}_2 \mid \phi$



DNAT: $\mathtt{n}_1 > [\mathtt{n}_2]\ \mathtt{n}_t \mid \phi$



SNAT: $\mathtt{n}_1\ [\mathtt{n}_t] > \mathtt{n}_2 \mid \phi$



ssh example revisited

Mignis rule:

```
* > local:22  tcp
```

corresponds to:

```
iptables -t filter -P INPUT DROP
iptables -t filter -P OUTPUT DROP
iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -t filter -A OUTPUT -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

iptables vs Mignis

```
1) iptables ... --source N1 -j ACCEPT
1a)iptables ... --source N5 --dport 22 -j DROP
1b)iptables ... --source N5 -j ACCEPT
2) iptables ... --dport 80 -j DROP
3) iptables ... --source N2 -j ACCEPT
3a)iptables ... --source N4 -j ACCEPT
4) iptables ... --dport 22 -j DROP
5) iptables ... --source N3 -j ACCEPT
```

```
N1 > *
N2 > *:*\80
N3 > *:*\(80,22)
N4 > *:*\80
N5 > *:*\22
```

The “demotivating” example

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
iptables -t mangle -P PREROUTING DROP
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A INPUT -s 0.0.0.0 -d 255.255.255.255 -j ACCEPT
iptables -t mangle -A PREROUTING -s 0.0.0.0 -d 255.255.255.255 -j ACCEPT
iptables -t mangle -A PREROUTING -i eth0 -s 10.0.0.0/24 -j ACCEPT iptables -t mangle -A PREROUTING -i lo -s 127.0.0.1 -j
ACCEPT
iptables -A FORWARD -i eth0 -d 5.6.7.8 -j DROP
iptables -t mangle -A PREROUTING -i eth1 -d 10.0.0.2 -p tcp --dport 22 -m state --state NEW -j DROP iptables -A FORWARD -i
eth1 -d 10.0.0.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -s 10.0.0.2 -p tcp --sport 22 -o eth1 -m state --state ESTABLISHED -j ACCEPT iptables -t nat -A
PREROUTING -i eth1 -d 1.2.3.4 -p tcp --dport 22 -j DNAT --to-destination 10.0.0.2:22
iptables -t mangle -A PREROUTING -i eth1 -d 10.0.0.2 -p tcp --dport 80 -m state --state NEW -j DROP iptables -A FORWARD -i
eth1 -d 10.0.0.2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -s 10.0.0.2 -p tcp --sport 80 -o eth1 -m state --state ESTABLISHED -j ACCEPT iptables -t nat -A
PREROUTING -i eth1 -d 1.2.3.4 -p tcp --dport 80 -j DNAT --to-destination 10.0.0.2:80
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED -j ACCEPT iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o
eth1 -j MASQUERADE
```

… in Mignis

```
ALIASES
mypc              10.0.0.2
router_ip         1.2.3.4
malicious_host    5.6.7.8

FIREWALL
lan [.] > ext
ext > [router_ip:80] mypc:80    tcp
ext > [router_ip:22] mypc:22    tcp
lan / malicious_host
```