Firewalls

Sicurezza (CT0539) 2019-20
Università Ca' Foscari Venezia

Riccardo Focardi
www.unive.it/data/persone/5590470
secgroup.dais.unive.it

Università
Ca'Foscari
Venezia

Motivations

Networking is **complex** and **pervasive**

- Local Area Networks (**LANs**) connecting **PCs**, **servers**, ...
- Wide Area Networks (**WANs**) connecting geographically **distributed** LANs
- **Internet** connectivity
- **Cloud** computing
- Internet of Things (**IoT**), **Industry 4.0**, ...

Motivations

Host-based vs network-based defence

Multitude of Operating Systems, e.g., Windows, Linux, MacOS, … and applications

Host-based defence: security flaws fixed on **every** system/application

Network-based defence: firewalls prevent attacks to **all systems**

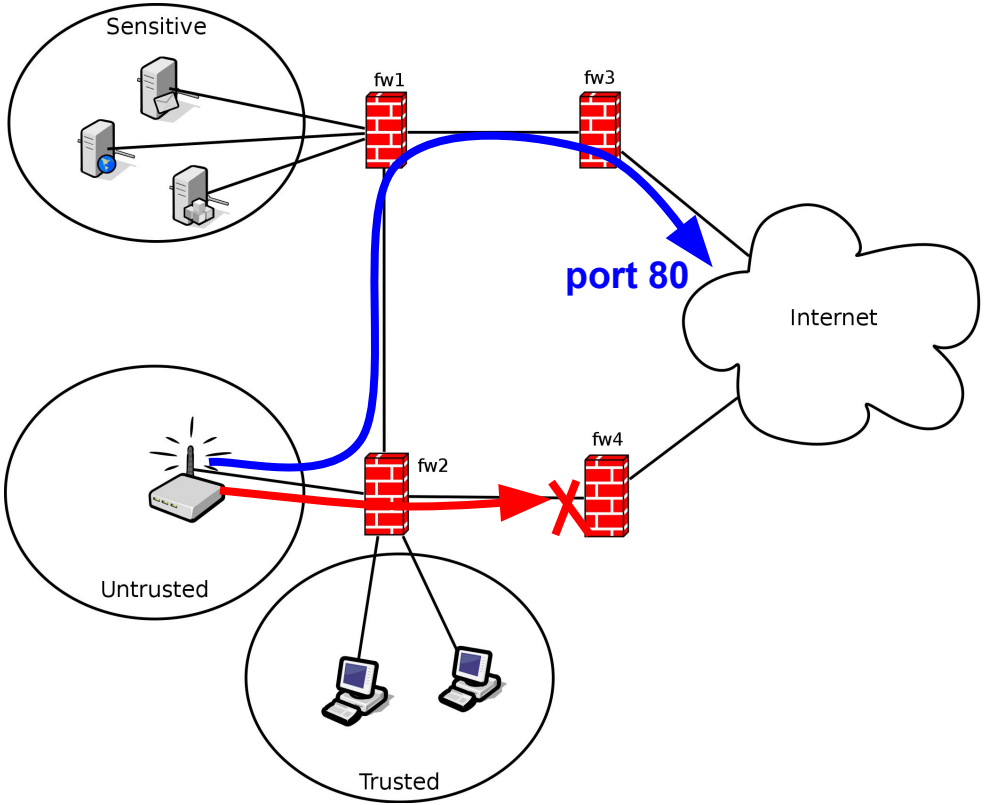
⇒ **Single point** for audit / security
⇒ Extra layer (***defence in depth***)

Example: *security and audit*

Three subnetworks: Sensitive, Trusted and Untrusted

Untrusted network should reach the Internet only

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

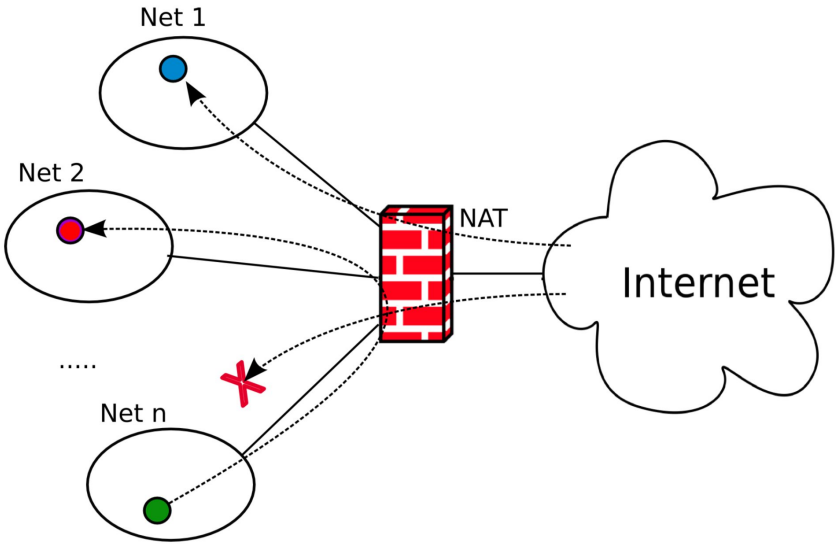


Typical firewall operations

Forward: packets are **forwarded** from one subnet to another
Example: from “Net n” to “Net 2”

Drop: packets are **forbidden** from one subnet to another
Example: from the Internet to “Net n”

Translate: packets addresses are **translated** while delivered
Example: from the Internet to “Net 1”

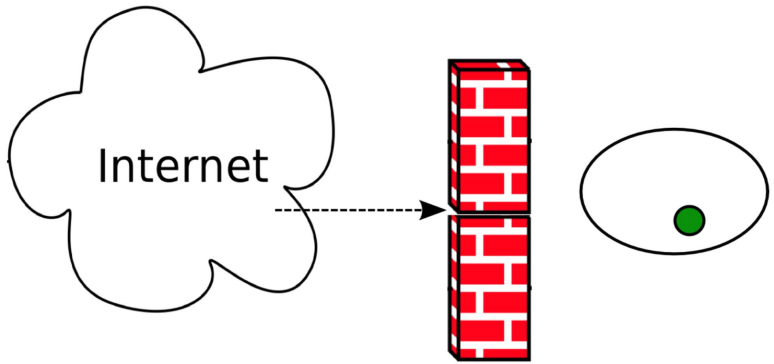


Drop

Firewalls forbid packets based on

- source/destination address
- source/destination port
- packet payload

Example: Forbid connections to specific ports (services) unless source address belongs to trusted networks



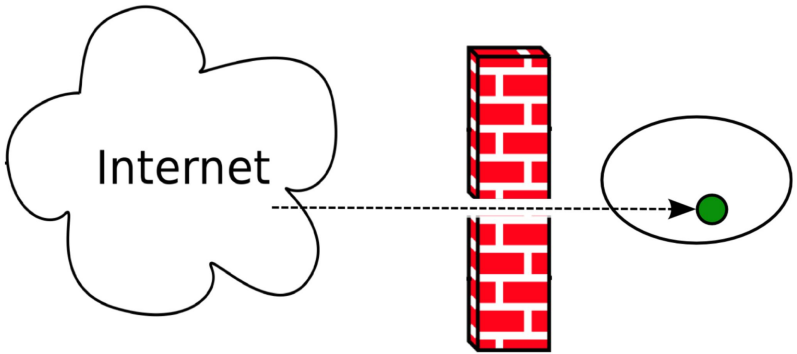
Accept/Forward

As for drop it is based on

- source/destination address
- source/destination port
- packet payload

Least privilege: drop connections unless they are really **needed**

⇒ **Drop by default**, and specify what to accept



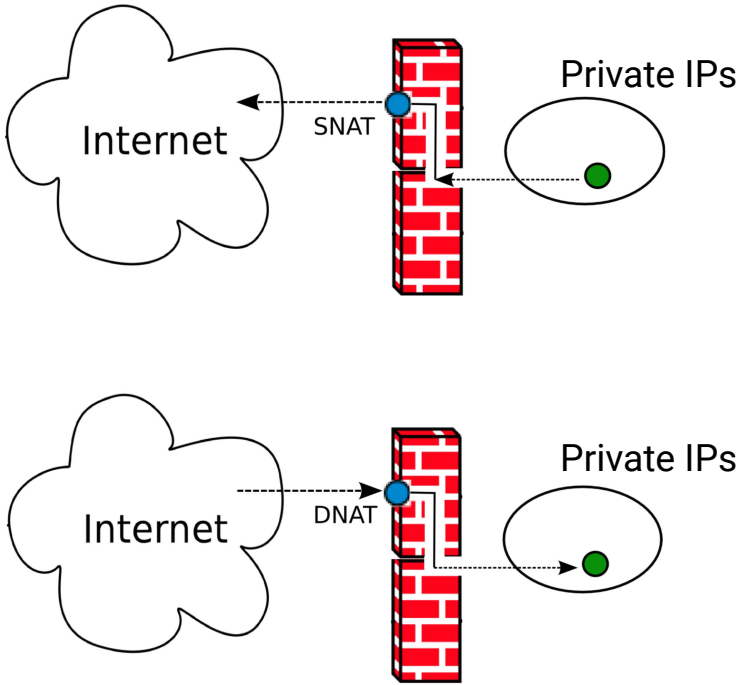
Network Address Translation (NAT)

Network Address Translation (**NAT**): is typically necessary in LANs with private IP addresses

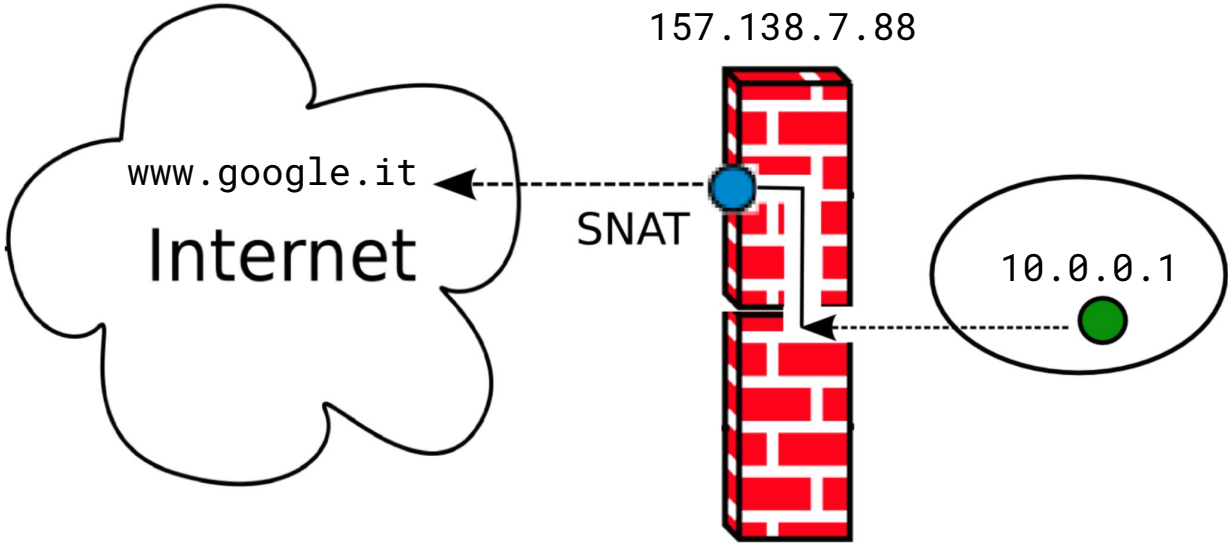
Source NAT: **outgoing** traffic needs a public IP **source** address

Destination NAT: **incoming** traffic needs a public IP **destination** address

NAT is implemented **transparently** in **stateful** firewalls

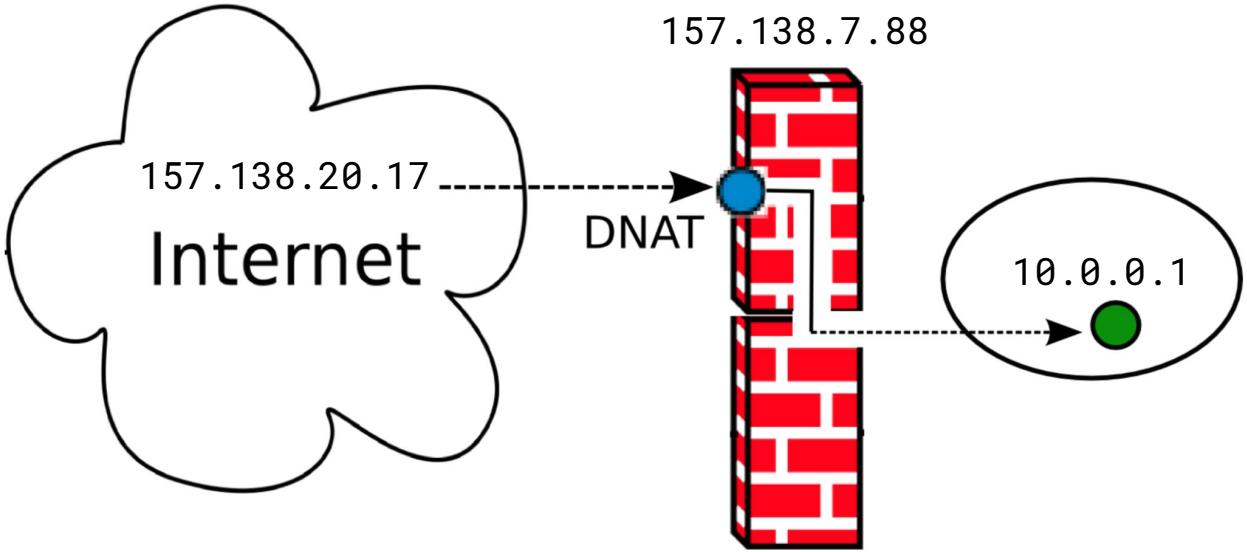


Example: Source NAT



`www.google.it` answers to `157.138.7.88`. The (stateful) firewall transparently **translates** the **destination** address into `10.0.0.1`

Example: Destination NAT



`10.0.0.1` answers to `157.138.20.17`. The stateful firewall transparently **translates** the **source** address into `157.138.7.88`

Case study: netfilter

Standard **firewall tool** in Linux

Netfilter allows for:

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

Configured through **`iptables`**, a very powerful and flexible tool

`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: lists of rules in netfilter

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

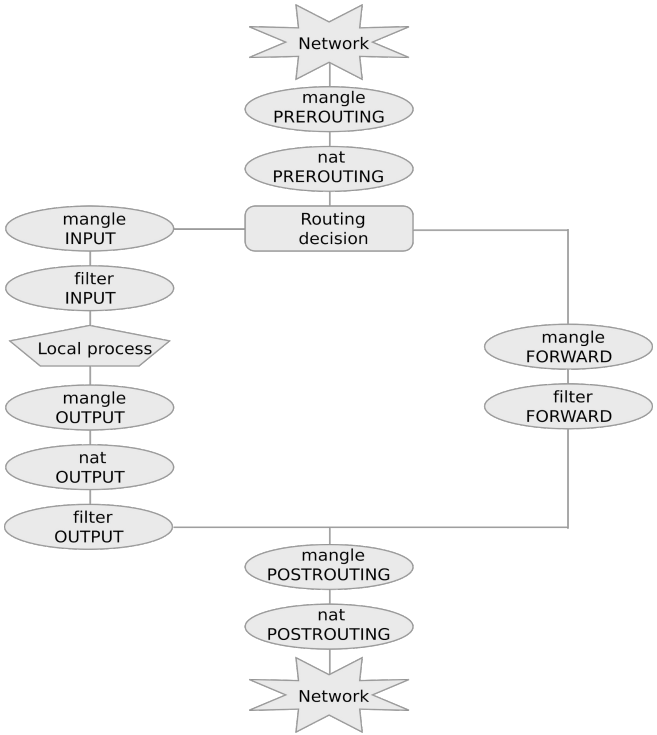
`PREROUTING` p **reaches** the host

`FORWARD` p is **forwarded**

`POSTROUTING` p is about to **leave**

`INPUT` p is **routed to** the host

`OUTPUT` p is **generated** by the host



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

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

The most commonly used targets are:

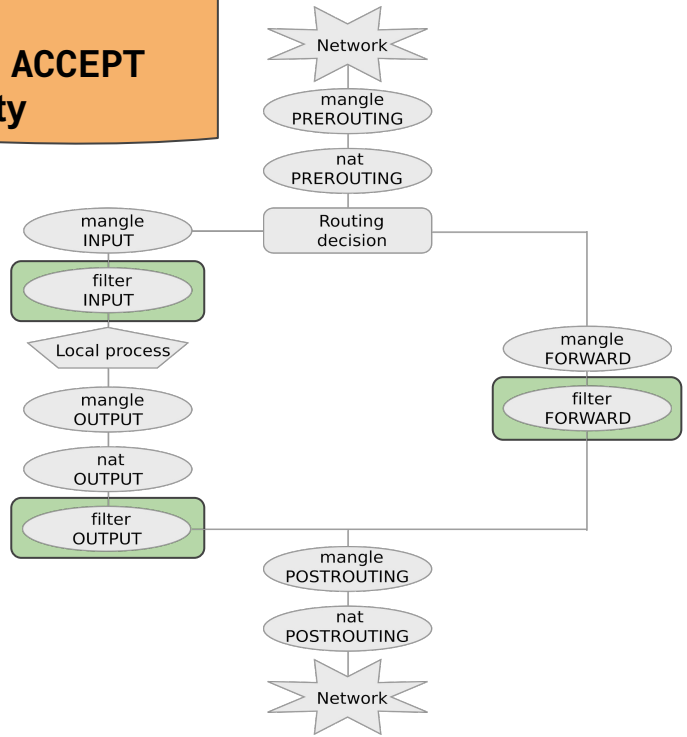
- **`ACCEPT`**, for **accepting** the packet
- **`DROP`**, for **discarding** it
- **`DNAT`**, for **destination NAT**
- **`SNAT`** for **source NAT**

Example: list rules and 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
```

Firewall is *disabled*:

1. Default policies are **ACCEPT**
2. All chains are **empty**



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

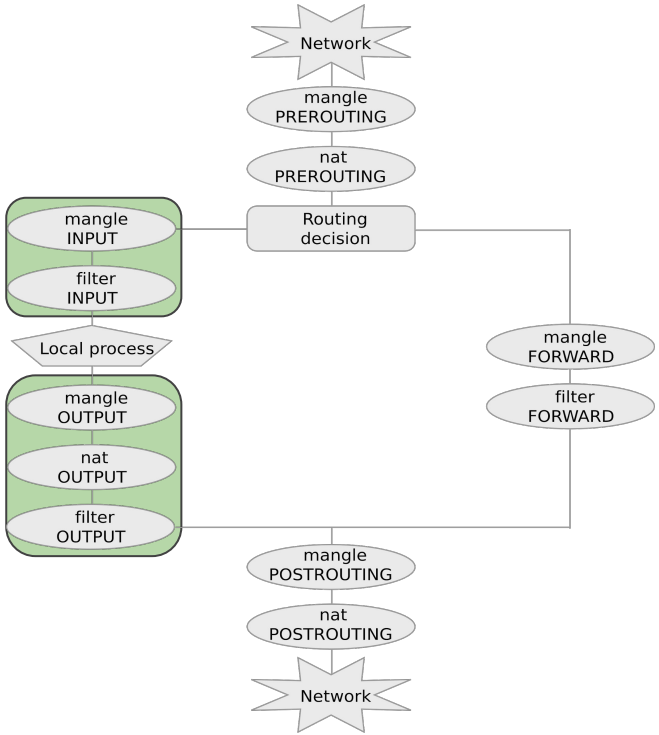
`-L` stands for “list”

Configuring a firewall

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

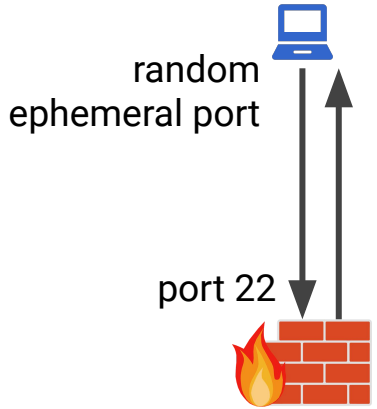
IMPORTANT: do not cut you off!
Enable ssh before default policy is set to DROP!

Notice that **both directions** are necessary
⇒ Both `INPUT` and `OUTPUT` chains!



Configuring a firewall: enable ssh

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



`-A` appends the rule to the specified chain

`-p tcp` specifies tcp protocol

`-dport` and `-sport` specify destination and source port

- in INPUT destination port is 22 (ssh)
- in OUTPUT source port is 22 (answers to ssh incoming packets)

`-j ACCEPT` specifies the ACCEPT target

Configuring a firewall: 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 match the two rules
(we can observe this through the **-v** option)

Least privilege: default DROP policy

Now that ssh is enabled it is possible to apply the least privilege principle and block any incoming packet which is not explicitly accepted

Set a **default DROP policy** for incoming connections (**INPUT** chain):

```
iptables -P INPUT DROP
```

Any packet which is not directed to port 22 (ssh) will be dropped:

```
# iptables -L -v
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
```

Rule ordering, chains and connections

Q: What happens if we now add the following rule?

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

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

Q: Can we connect to a web server (port 80)?

A: Yes, **OUTPUT** policy is **ACCEPT**

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

A: Answer is **dropped**! Firewall only admits **ssh** incoming connections

Stateful filtering

`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**
- Network Address Translation is also tracked (NAT)

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

⇒ both **ssh** and **established** incoming packets will be accepted

DNAT example

Forward web-traffic to a local server

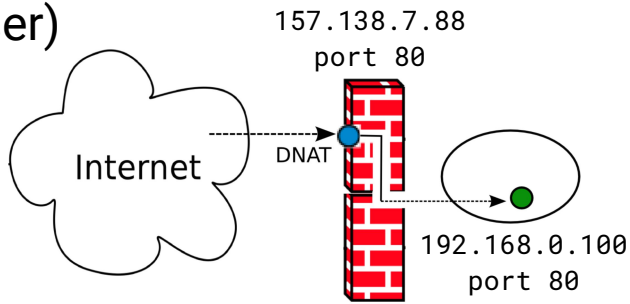
NOTE: DNAT is done before routing (chain PREROUTING)

```
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** (**stateful** firewall)



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
```

Assume N1, N2, ... are disjoint networks (no overlap!)

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
```

Assume N1, N2, ... are disjoint networks (no overlap!)

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

A2: adding rule `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
```

Assume N1, N2, ... are disjoint networks (no overlap!)

Q2: how can we accept packets from N5 that are not addressed to port 22?

A2: adding rules

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

Final configuration

```
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
```

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
  ⇒ Default DROP suggested (**least privilege**)
- 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:

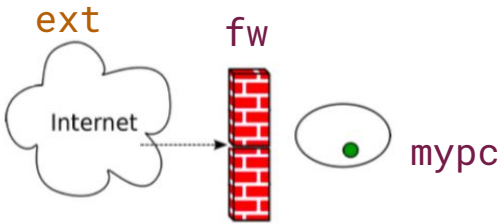
- Network Address Translation (NAT)
- Least privilege (default discard)
- **Declarative** style: order of the rules is irrelevant!
- Explicit rejects (with precedence over “positive” rules)
- Simple formal semantics (proof of correctness)

Developed by secgroup@unive:

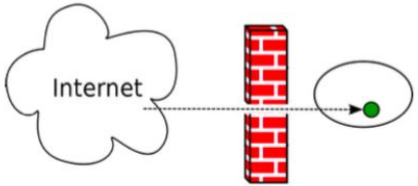
https://github.com/secgroup/Mignis

Mignis rules

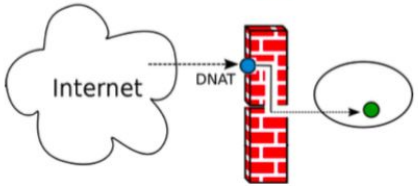
```
ext / mypc
```



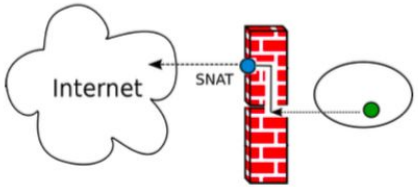

```
ext > mypc
```




```
ext > [fw:80] mypc:80  tcp
```




```
mypc [.] > ext
mypc [fw] > ext
```



Mignis example

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
```

Typical Mignis configuration

```
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
```

… and half of its translation

```
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
```

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
```

- Order of rules matter
- ACCEPT and DROP
- Hard to maintain

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

- Declarative style
- Least privilege
- Order does not matter
- Easy to maintain