Mignis: a tool for declarative firewall configuration

Riccardo Focardi

joint work with
Pedro Adão, Claudio Bozzato,
Gian-Luca Dei Rossi and Flaminia L. Luccio

Università
Ca'Foscari
Venezia

TÉCNICO
LISBOA

Work partially supported by

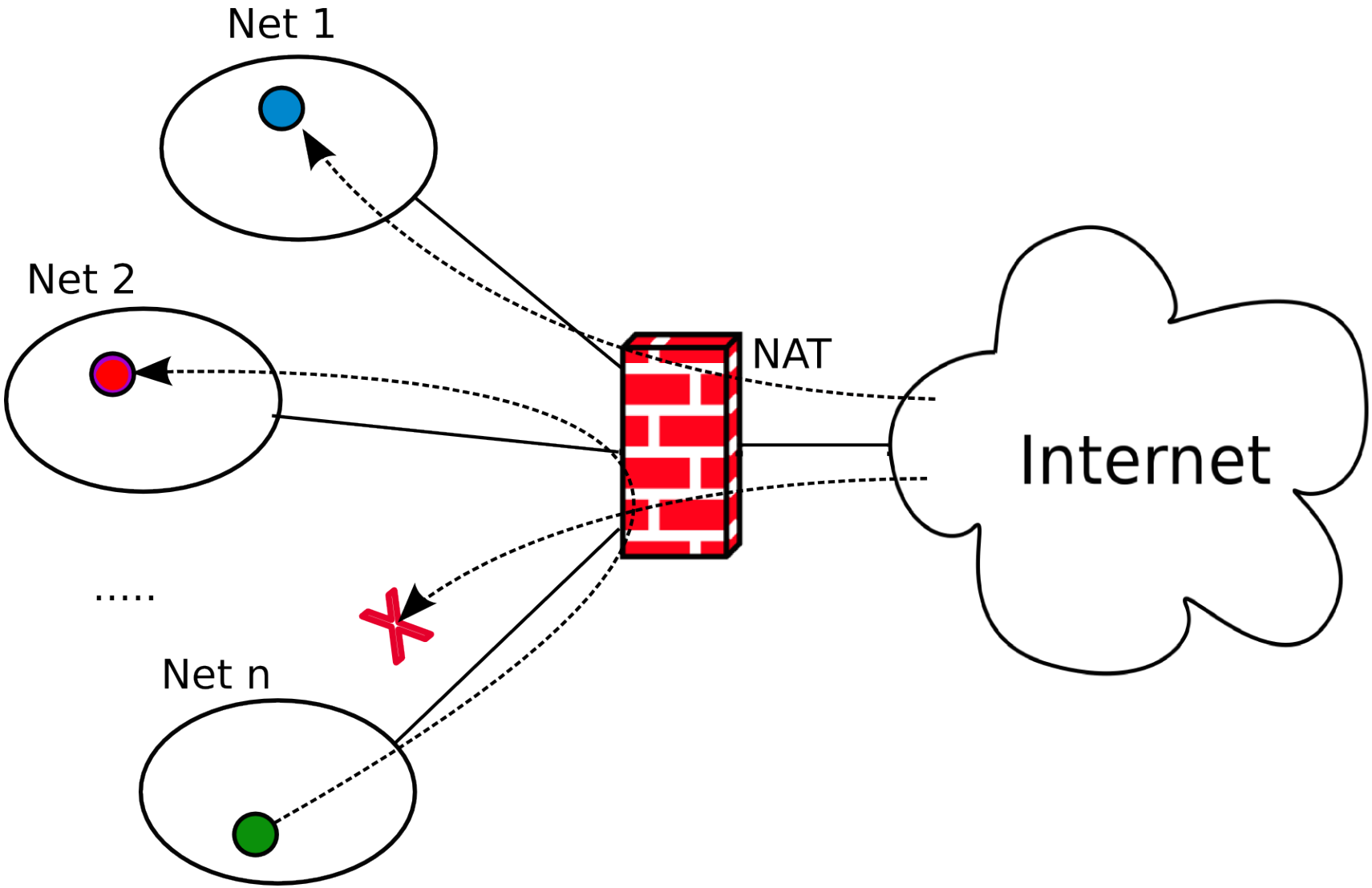
FCT projects ComFormCrypt PTDC/EIA-CCO/113033/2009 and PEst-OE/EEI/LA0008/2013,
and the Italian PRIN project "Security Horizons".

Presented at IEEE CSF'14, Vienna, July 19–22, 2014

Outline

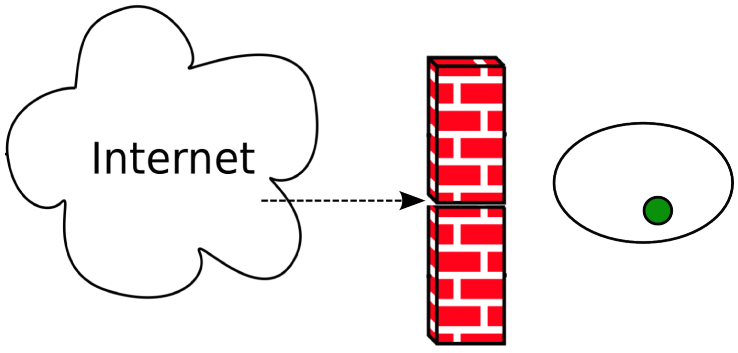
1. Introduction to Firewalls
2. Case study: The Linux Netfilter/iptables framework
3. Mignis, a new firewall specification language
4. Open-source implementation and examples

What is a Firewall?

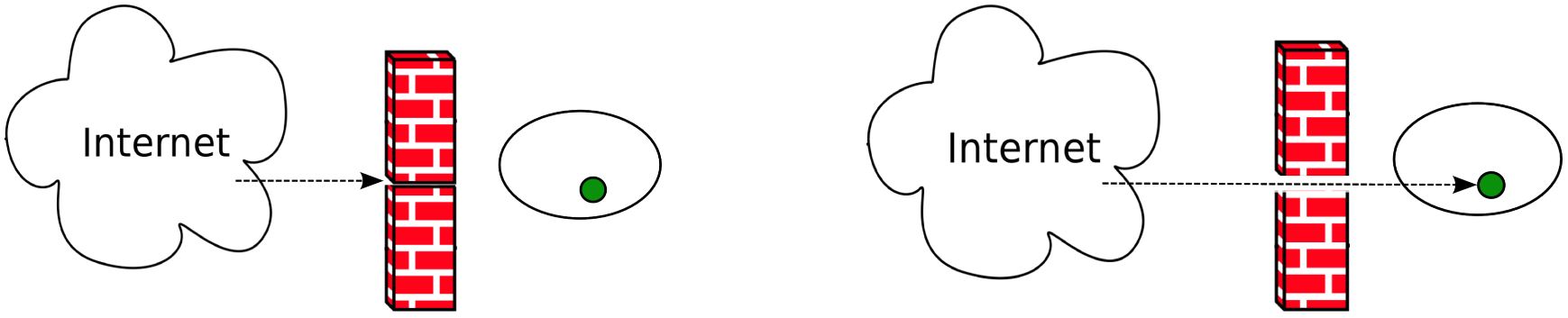


Firewall basic operations

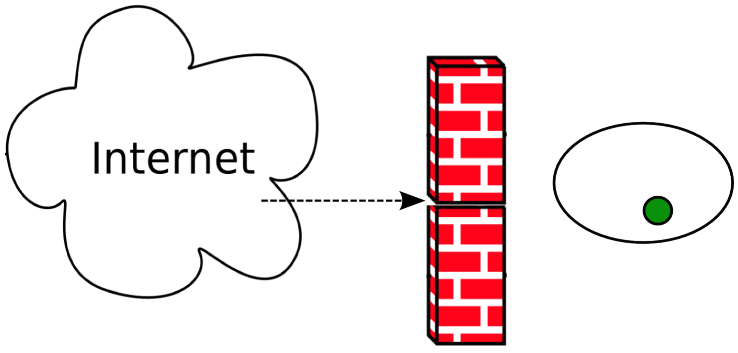
Firewall basic operations

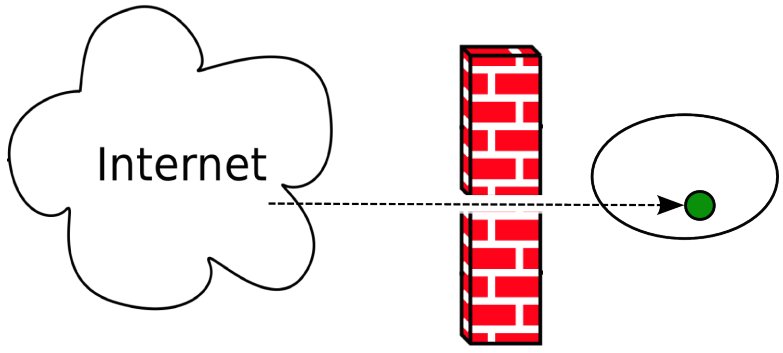


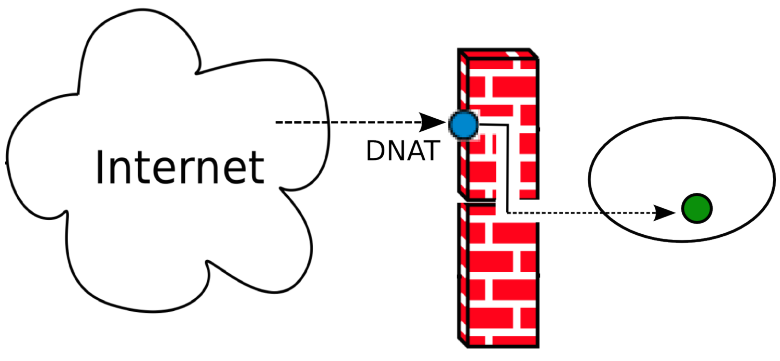
Firewall basic operations



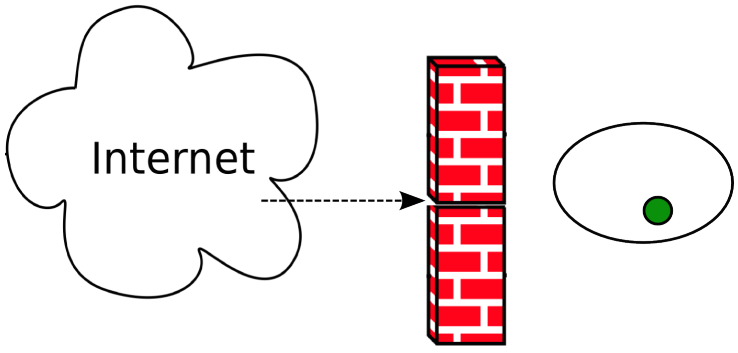
Firewall basic operations

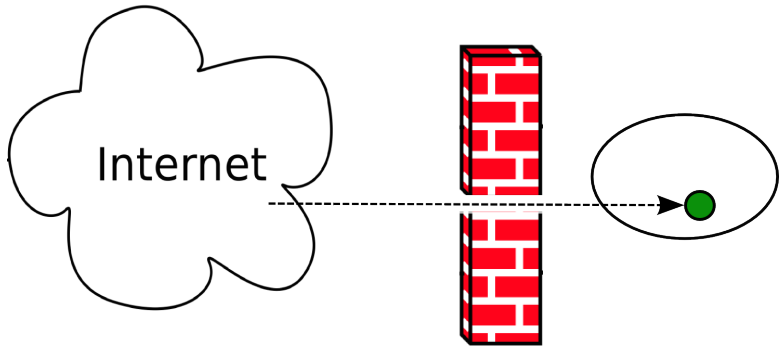


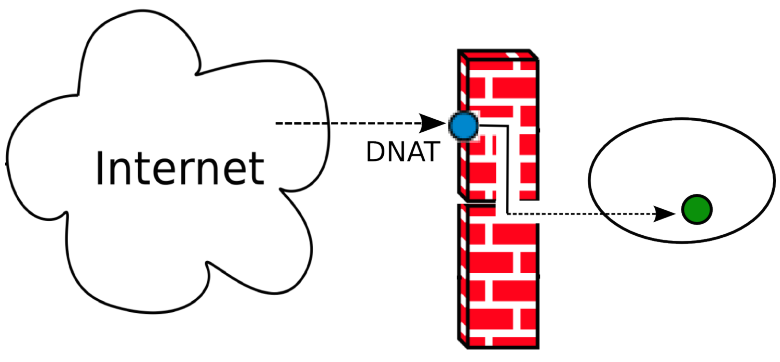


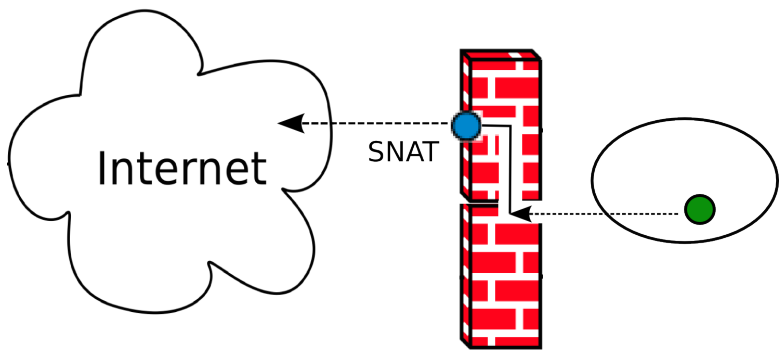


Firewall basic operations









Netfilter

Standard framework for packet filtering and address translation in Linux

- based on *tables* each containing lists of *rules* called *chains*, that are inspected in specific moments of packets life cycle
- If the rule matches the packet is processed as specified in the target. For example: `ACCEPT`, `DROP`, `DNAT`, `SNAT`
- If a rule does not match, the next rule in the chain is examined
- A default policy is triggered if none of the rules in the chain matches
- Target can also be a jump into a different chain

$\Rightarrow$ Powerful and flexible but configurations become often complex and hard to maintain

$\Rightarrow$ 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
```

Example: drop everything but ssh

- Default policy is `ACCEPT`
- We change the default policy to `DROP`

```
iptables -t filter -P INPUT DROP
```

- and we only enable ssh connections:

```
iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
```

- It is now possible to connect to the host only via ssh

Example: drop everything but ssh

- Default policy is `ACCEPT`
- We change the default policy to `DROP`

```
iptables -t filter -P INPUT DROP
```

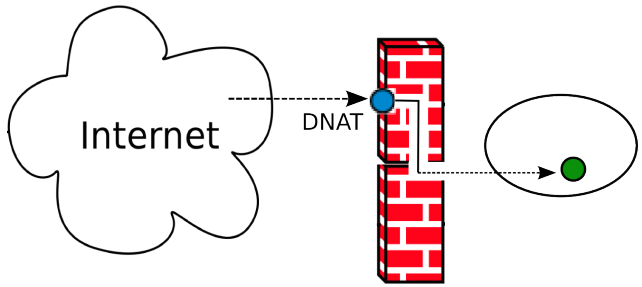
- and we only enable ssh connections:

```
iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
```

- It is now possible to connect to the host only via ssh

$\Rightarrow$ since rules are inspected in the order in which they appear, adding (another) similar rule but with target `DROP` to the end of the `Input` chain would not prevent ssh connections

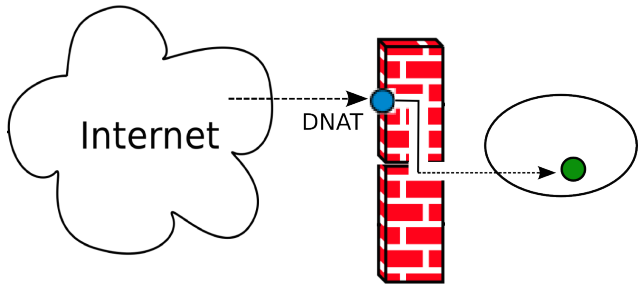
Example: forward web-traffic to a local server



- All connections addressed to 192.168.0.1 on port 80 are translated into 192.168.0.100 on the same port.

```
iptables -t nat -A PREROUTING -p tcp  -d 192.168.0.1 --dport 80
      -j DNAT --to-destination 192.168.0.100:80
```

Example: forward web-traffic to a local server

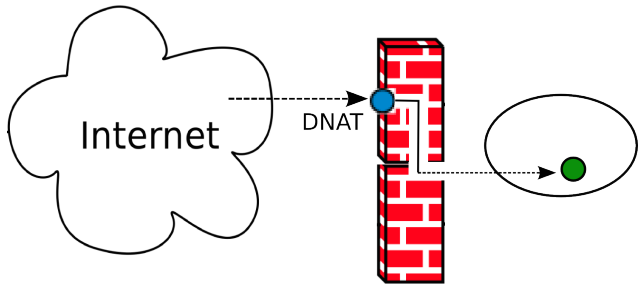


- All connections addressed to 192.168.0.1 on port 80 are translated into 192.168.0.100 on the same port.

```
iptables -t nat -A PREROUTING -p tcp  -d 192.168.0.1 --dport 80
      -j DNAT --to-destination 192.168.0.100:80
```

- Answers from host 192.168.0.100 will be translated as coming from host 192.168.0.1 in order to be accepted by the browser.

Example: forward web-traffic to a local server



- All connections addressed to 192.168.0.1 on port 80 are translated into 192.168.0.100 on the same port.

```
iptables -t nat -A PREROUTING -p tcp  -d 192.168.0.1 --dport 80
      -j DNAT --to-destination 192.168.0.100:80
```

- Answers from host 192.168.0.100 will be translated as coming from host 192.168.0.1 in order to be accepted by the browser.

$\Rightarrow$ Translation is applied to all packets on the same connection.

Maintaining configurations in Netfilter

Netfilter allows for alternation of `DROP` and `ACCEPT`:

```
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
```

which allows a compact specification but....

1. where can the packets from N2 go?
2. how do we accept packets from N4 that are not addressed to port 80?
3. how do we accept packets from N5 that are not addressed to port 22?

Maintaining configurations in Netfilter

Netfilter allows for alternation of `DROP` and `ACCEPT`:

```
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
```

which allows a compact specification but....

1. where can the packets from N2 go? Context-dependent!
2. how do we accept packets from N4 that are not addressed to port 80?
3. how do we accept packets from N5 that are not addressed to port 22?

Maintaining configurations in Netfilter

Netfilter allows for alternation of `DROP` and `ACCEPT`:

```
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
```

which allows a compact specification but....

1. where can the packets from N2 go? Context-dependent!
2. how do we accept packets from N4 that are not addressed to port 80?
3. how do we accept packets from N5 that are not addressed to port 22?

```
1) iptables ... --source N1 -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 configurations in Netfilter

Netfilter allows for alternation of `DROP` and `ACCEPT`:

```
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
```

which allows a compact specification but....

1. where can the packets from N2 go? Context-dependent!
2. how do we accept packets from N4 that are not addressed to port 80?
3. how do we accept packets from N5 that are not addressed to port 22?

```
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 configurations in Netfilter

In general it can be hard to maintain Netfilter configurations as

- they do not have a fixed structure
- order matters and `ACCEPT` and `DROP` are often alternating
- semantics depends on which table and chain a rule belongs to
- to drop a packet, one must be sure that the rule is placed before all the `ACCEPT` rules of the corresponding chain
- policy can be default `ACCEPT` or default `DROP`
- real configurations can easily have more than 1000 lines!
- real configurations grow over time, and are maintained by several systems administrators.

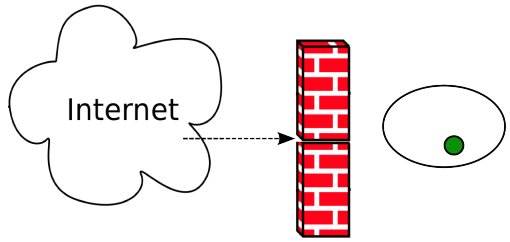
Mignis

Mignis is a tool firewall specification that supports:

- Network Address Translation (NAT)
- Minimum privilege
- Declarative style/Order of the rules is irrelevant
- Explicit rejects (with precedence over "positive" rules)
- Simple formal semantics

Mignis rules

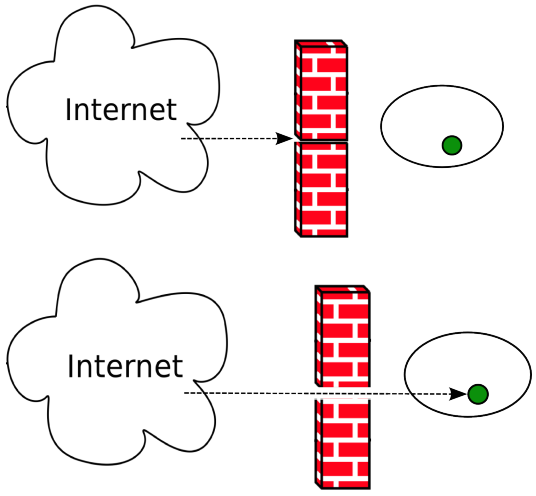
DROP: $\mathtt{n}_1$ / $\mathtt{n}_2$ | ϕ



Mignis rules

DROP: $\mathtt{n}_1$ / $\mathtt{n}_2$ | ϕ

ACCEPT: $\mathtt{n}_1 > \mathtt{n}_2$ | ϕ

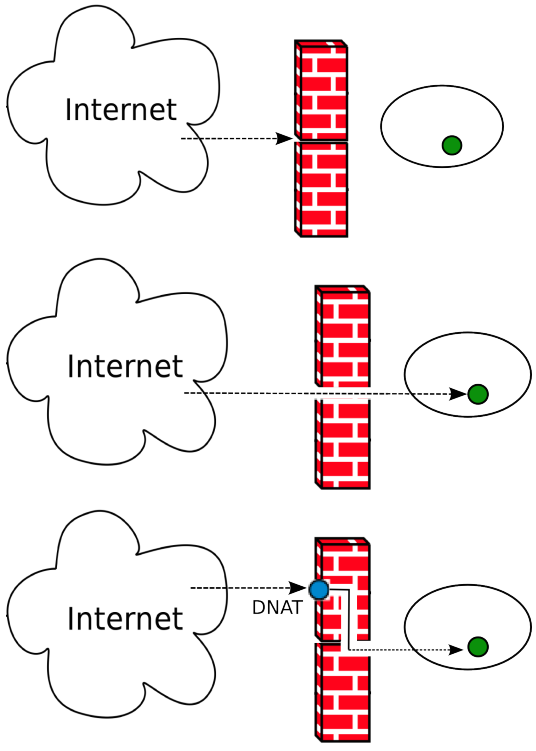


Mignis rules

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

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

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



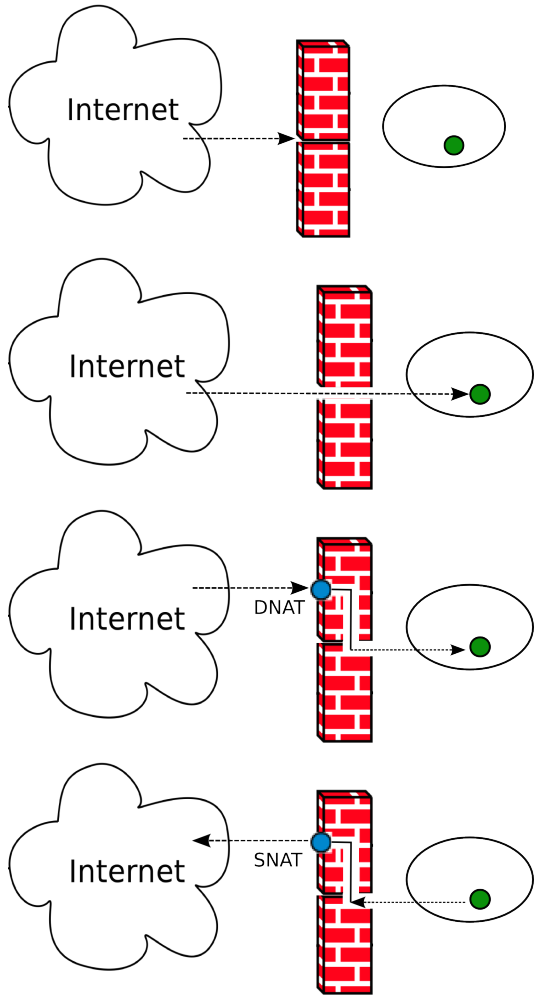
Mignis rules

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

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

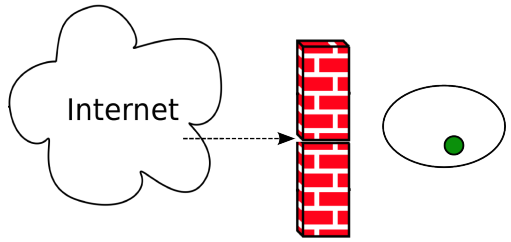
DNAT: $\mathtt{n}_1 > [\mathtt{n}_2]\ \mathtt{n}_t \mid \phi$

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

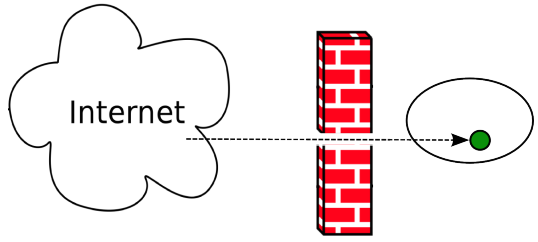


Mignis rules

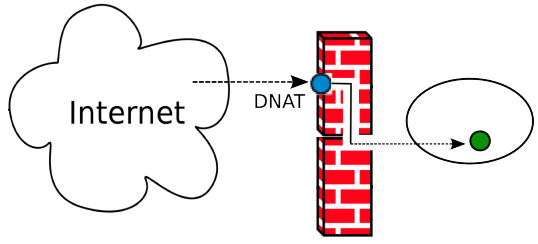
DROP: $\mathtt{n_1}$ / $\mathtt{n_2}$ | ϕ



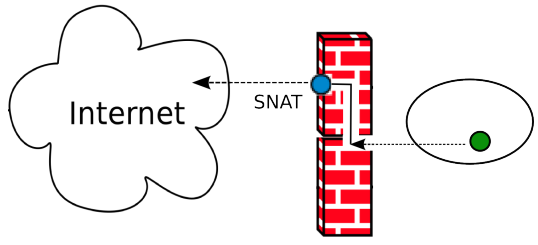
ACCEPT: $\mathtt{n_1} > \mathtt{n_2} \mid \phi$



DNAT: $\mathtt{n_1} > [\mathtt{n_2}]\ \mathtt{n_t} \mid \phi$



SNAT: $\mathtt{n_1}\ [\mathtt{n_t}] > \mathtt{n_2} \mid \phi$



$\Rightarrow$ Packets on established connections not explicitly forbidden go through

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
```

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
```

DNAT rules such as

```
iptables -t nat -A PREROUTING -p tcp -i eth0
      -d 192.168.0.1 --dport 80 -j DNAT --to-destination 192.168.0.100:80
```

are easily expressed in Mignis as

```
* > [192.168.0.1:80] 192.168.0.100:80  tcp
```

Maintaining configurations in our abstract language

Recall the Netfilter example

```
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
```

Maintaining configurations in our abstract language

Recall the Netfilter example

```
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
```

in Mignis it is written as

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

1. where can the packets from N2 go?
2. how do we accept packets from N4 that are not addressed to port 80?
3. how do we accept packets from N5 that are not addressed to port 22?

Maintaining configurations in our abstract language

Recall the Netfilter example

```
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
```

in Mignis it is written as

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

1. where can the packets from N2 go? Immediate! Context-independent.
2. how do we accept packets from N4 that are not addressed to port 80?
3. how do we accept packets from N5 that are not addressed to port 22?

Maintaining configurations in our abstract language

Recall the Netfilter example

```
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
```

in Mignis it is written as

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

1. where can the packets from N2 go? Immediate! Context-independent.
2. how do we accept packets from N4 that are not addressed to port 80?
3. how do we accept packets from N5 that are not addressed to port 22?

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

The Mignis tool

Mignis is written in Python and is about 1500 lines of code

1. Translates a firewall specification into real `iptables`. Translation has been proved correct on a formal model
2. Looks for overlaps and inconsistencies
3. Adds default router rules to initiate external connections, receive pings, and broadcast traffic which are usually considered safe
4. Adds rules that guarantee that each IP address will originate only from its assigned interface
5. Adds log rules

In use since January 2014 at the CS Department in Venice!

Available at `https://github.com/secgroup/Mignis`

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
```

... and its Mignis counterpart

```
INTERFACES
lan eth0   10.0.0.0/24
ext eth1   0.0.0.0/0

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 its Mignis counterpart

```
INTERFACES
lan eth0   10.0.0.0/24
ext eth1   0.0.0.0/0

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
```

Which one do you prefer? ;)

Conclusion and future work

- First tool for firewall configuration with
  1. Support for NAT
  2. Both `ACCEPT` and `DROP` rules
  3. Declarative style
  4. Explicit rejects
  5. proof of correctness
- Good compromise between expressiveness and simplicity, confirmed by the real case-study
- Declarative spec that can be "queried" to show the configuration of specific hosts: found very useful by our admins!
- We are extending Mignis to networks with more than one firewall and "information-flow style" specifications
- We intend to compile Mignis to other firewall systems

Selected references

Uncomplicated firewall.
`https://help.ubuntu.com/community/UFW`, 2013.

F. Cuppens, N. Cuppens-Boulahia, T. Sans, and A. Miège.
A formal approach to specify and deploy a network security policy.
In *Formal Aspects in Security and Trust (FAST'04)*, pages 203–218, 2004.

Joshua D. Guttman, Amy L. Herzog.
Rigorous automated network security management.
Int. Journal Information Secururity (2005) 4: 2948.

A. Jeffrey and T. Samak.
Model checking firewall policy configurations.
In *Proc. of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY '09)*, pages 60–67. IEEE Computer Society, 2009.

R. M. Marmorstein.
Formal Analysis of Firewall Policies.
PhD thesis, College of William and Mary, Williamsburg, VA, May 2008.

S. Pozo, R. Ceballos, and R. M. Gasca.
Afpl, an abstract language model for firewall acls.
In *Proc. of the international conference on Computational Science and Its Applications, Part II*, ICCSA '08, pages 468–483. Springer-Verlag, 2008.