Firewalls are a mess!

Compiling and decompiling network policies

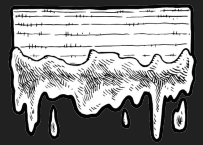
Lorenzo Veronese

wert310.github.io

310wert@gmail.com | 852058@stud.unive.it

@310wert

Università Ca' Foscari, Venezia



MOLECON 2019

Politecnico di Torino / November 30th /IT

```
speaker $ id
uid=100(wert310) groups=1337(mhackeroni),31337(c00kies@venice)
```

- MSc Student in CS @Ca' Foscari
- Playing CTFs with **mhackeroni** and **c00kies@venice**
  - Defense / Network / Infra / Web
- Organizer of **CCIT18/19 Finals**

Outline

Background on Netfilter
Configuring Firewalls
Validating/Decompiling Firewalls
Theoretical Background

netfilter/iptables

Background

Standard framework for packet filtering and address translation in Linux

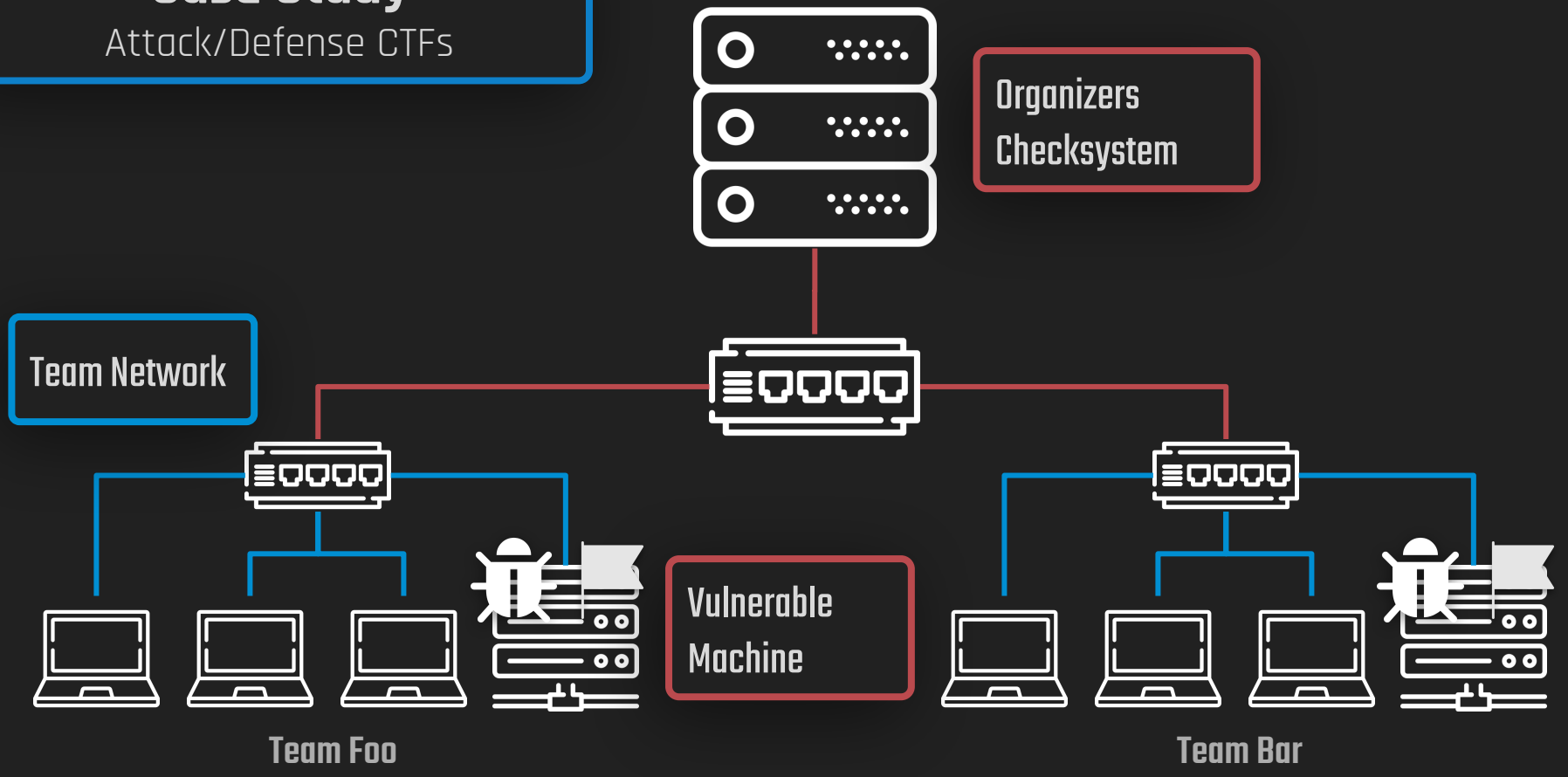
- Based on **tables** containing lists of rules called **chains,** inspected in specific moments of packets life cycle
- Each rule specifies a **condition** and a **target**
- Rules in a chain are evaluated **in order** (last rule: **default policy**)
- Supports **stateful** firewalling and **Network Address Translation** (NAT)

Allow only incoming SSH traffic to the firewall

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

Case Study

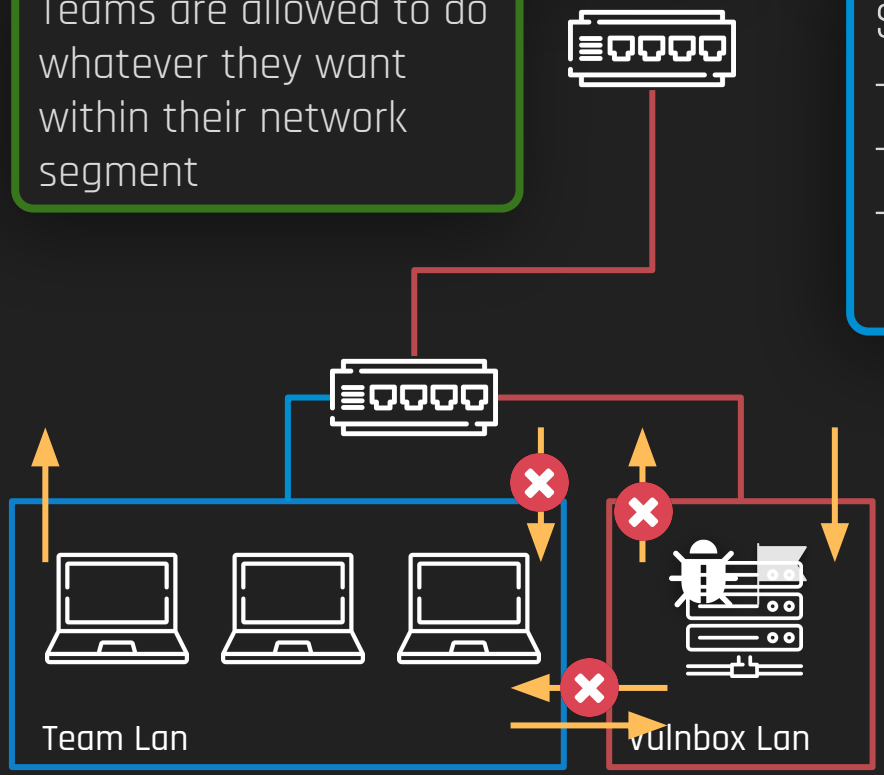
Attack/Defense CTFs



Case Study

Attack/Defense CTFs

Teams are allowed to do whatever they want within their network segment



Network Segmentation

Security Policy

- Team lan ⟶ game / Internet
- Team lan ⟶ Vulnbox (using ext ip)
- Vulnbox can only receive connections
    - on specific ports

Case Study

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A FORWARD -i team -j ACCEPT
iptables -A FORWARD -i game -o vuln -d $VULNIP -p tcp --dport $S1PRT -j ACCEPT
...
iptables -A FORWARD -i game -o vuln -d $VULNIP -p tcp --dport $SNPRT -j ACCEPT

iptables -t nat -A POSTROUTING -i team -o game -j MASQUERADE
```

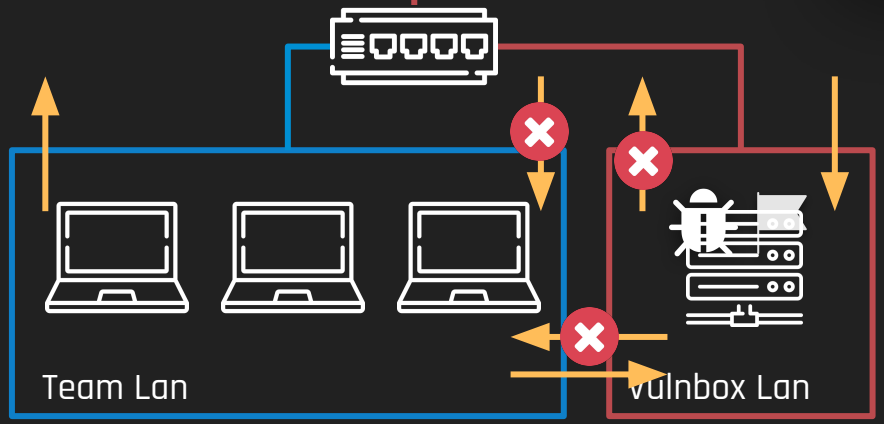
egmentation

olicy

→ game / Internet

→ Vulnbox (using ext ip)

- Vulnbox can only receive connections
  - on specific ports



Case Study

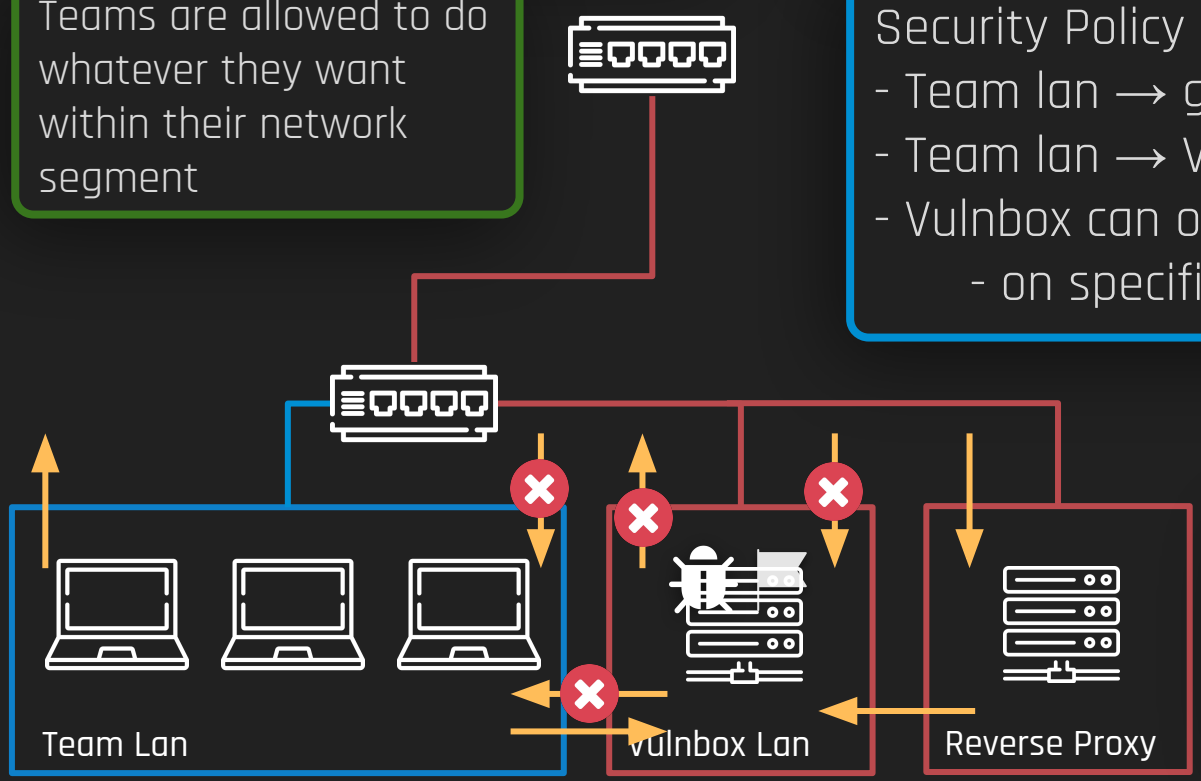
Attack/Defense CTFs

Network Segmentation

Teams are allowed to do whatever they want within their network segment

Security Policy

- Team lan ⟶ game / Internet
- Team lan ⟶ Vulnbox (using ext ip)
- Vulnbox can only receive connections
  - on specific ports

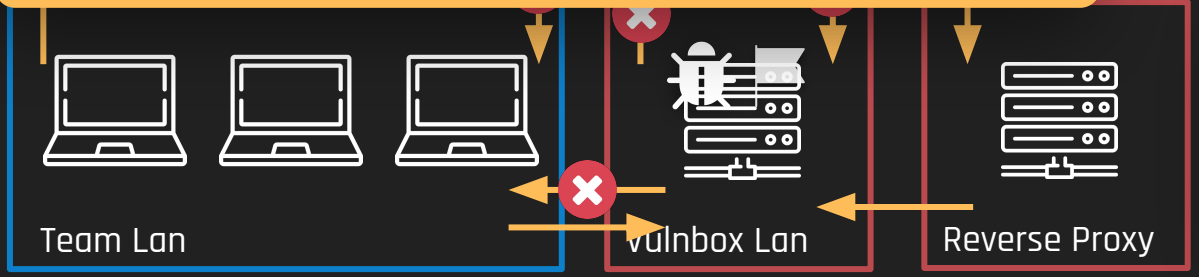


What if we need a **reverse proxy**?

egmentation

olicy
→ game / Internet
→ Vulnbox (using ext ip)
an only receive connections
pecific ports

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A FORWARD -i team -j ACCEPT
iptables -A FORWARD -i proxy -j ACCEPT

iptables -t mangle -A FORWARD -i game -d $PROXYIP -j DROP

iptables -A FORWARD -i game -d $PROXYIP -p tcp --dport $S1PRT -j ACCEPT
...
iptables -A FORWARD -i game -d $PROXYIP -p tcp --dport $SNPRT -j ACCEPT

iptables -t nat -A -i game -A PREROUTING -p tcp -d $VULNIP --dport $S1PRT \
   -j DNAT --to-destination $PROXYIP
...
iptables -t nat -A -i game -A PREROUTING -p tcp -d $VULNIP --dport $SNPRT \
   -j DNAT --to-destination $PROXYIP

iptables -t nat -A POSTROUTING -i team -o game -j MASQUERADE
```



What if we need a **reverse proxy**?

iptables issues

Firewall maintainability

Rules are **context-dependant!**

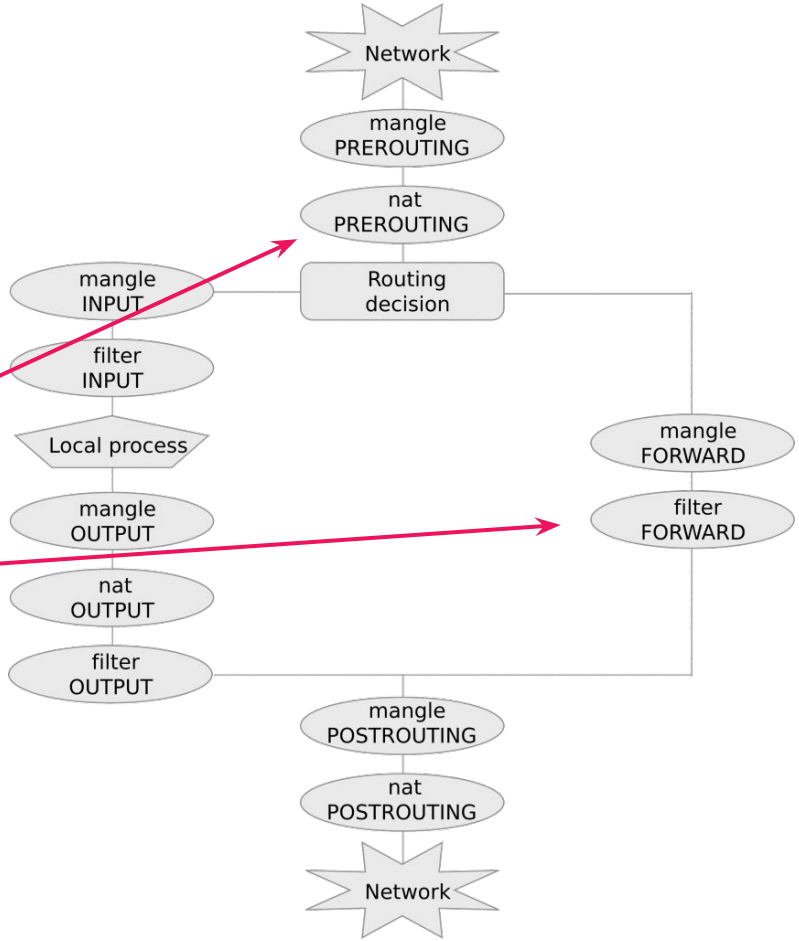
Packets from N2 to port 80 are **DROP**ed

```
iptables ... --source N1 -j ACCEPT
iptables ... --dport 80 -j DROP
iptables ... --source N2 -j ACCEPT
iptables ... --dport 22 -j DROP
```

Filters apply on **NAT**ed packets!

Order matters,
rule semantics depend on which **table and chain** is used.

Configurations grow over time and are maintained by several system administrators



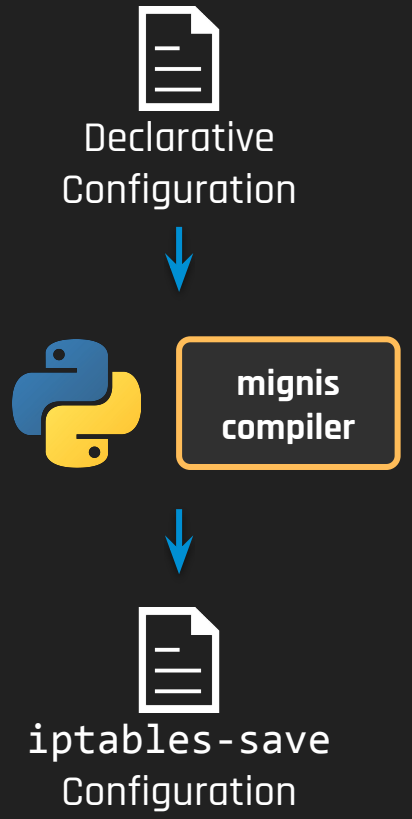
First Solution

Declarative Configurations

```
INTERFACES
ext    ethX  0.0.0.0/0
lan    ethX  192.168.XX.0/24
game   game  10.0.0.0/8
proxy  proxy 10.XX.XX.0/24

ALIASES
proxy_ip 10.XX.XX.2
vuln_ip  10.60.XX.2

FIREWALL
local > *
game > [vuln_ip:80] proxy_ip tcp
game > [vuln_ip:31337] proxy_ip tcp

lan [.] > ext
```

1. Declarative style
2. Order does not matter
3. No need to think about tables/chains

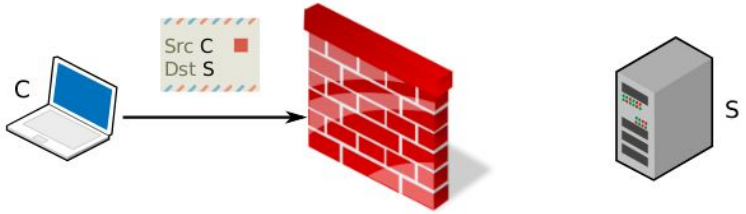
Default DROP
Explicit ACCEPT



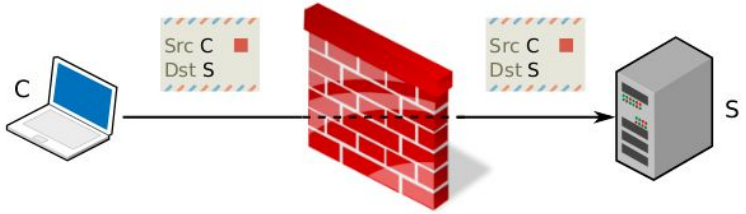
https://github.com/secgroup/mignis

Mignis Rules

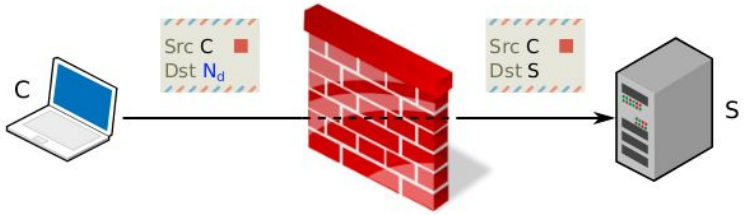
DROP: C / S | ϕ



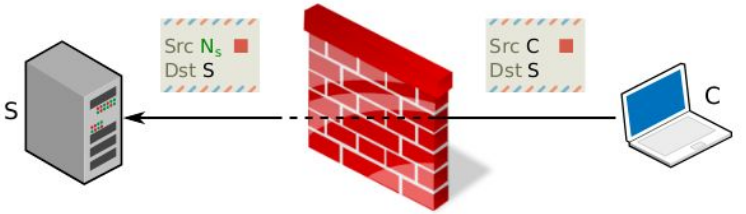
ACCEPT: C > S | ϕ



DNAT: C > [N_d] S | ϕ

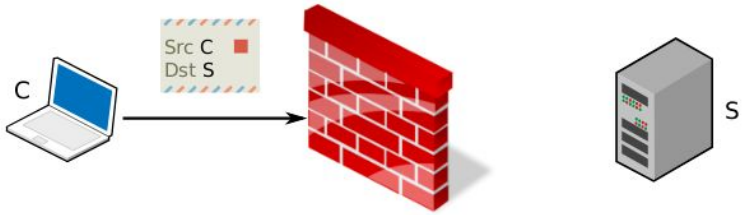


SNAT: C [N_s] > S | ϕ

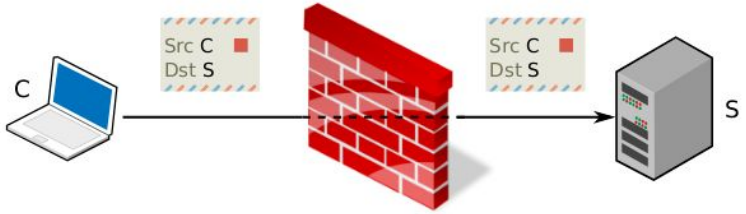


Mignis Rules

$\textsc{Drop}: \mathtt{C} \;/\; \mathtt{S} \mid \phi$



$\textsc{Accept}: \mathtt{C} > \mathtt{S} \mid \phi$



$\textsc{Dnat}: \mathtt{C} > [\mathtt{N_d}]\ \mathtt{S} \mid \phi$

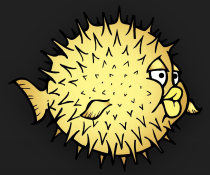
$[\mathtt{N_s}] > \mathtt{S} \mid \phi$

Abstract high level language with **single-step** semantics

The translation has been **formally verified** in a CSF '14 paper

General issues

Firewall maintainability

- **Low-level** configuration languages
- Rules are **context-dependent**
- Packet **routing** determines which rulesets are inspected
- **NAT** modifies the packet while it traverses the firewall

Huge already existent rulesets!

- We cannot just rewrite everything in mignis

Existing firewall systems differ in:

- How rules are **organized and inspected**
- How to **select the matching rule** (e.g., first vs last)

Second Solution

Validating firewalls and automated porting

`iptables` frontend

`pf` frontend

`ipfw` frontend

`Cisco IOS` frontend

Porting Module

Analysis Module

Declarative Specification

✓ ?

Queries

Multiple Policies

Equivalence ?

Implication ?

Diff ?

https://github.com/secgroup/fws

Second Solution

Validating firewalls and automated porting

`iptables` frontend

`pf` frontend

`ipfw` frontend

`Cisco IOS` frontend

Porting Module

Source IP	SNAT IP	DNAT IP	Destination IP	Destination Port
internal	ext_ip	-	* \ {internal}	443 80
*	-	web_server	ext_ip	443
*	-	ssh_server	ext_ip	22

Specification

✓ ?

Queries

Multiple Policies

Equivalence ?

Implication ?

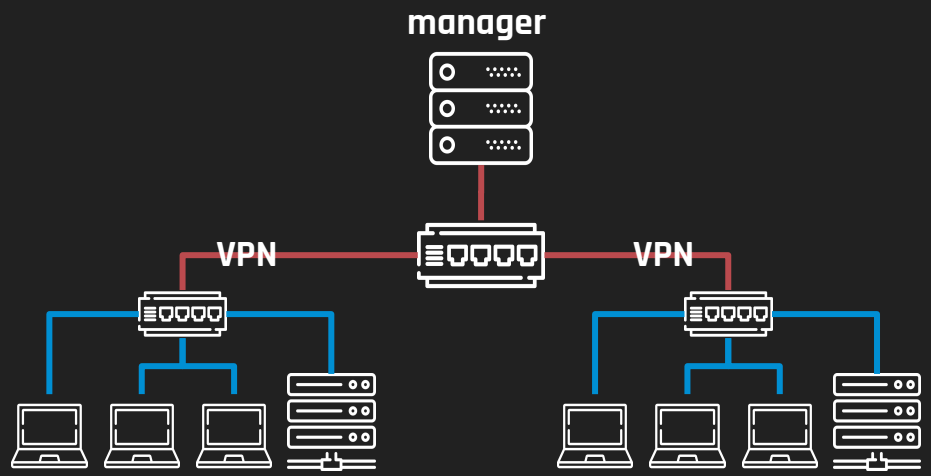
Diff ?

https://github.com/secgroup/fws

Case Study
Revisited

The network can be **open** or **closed** depending on the state of the game

CTF CheckSystem
3 Teams A/D CTF
~250 iptables rules



Case Study

```
FWS> synthesis(policy)
       in forward
       where srcIp = team03
```

Source IP	Source Port	Destination IP	Destination Port	Protocol	State
team03	*	manager_ip	*	icmp	NEW

Source IP	SNAT IP	Destination IP	Destination Port	DNAT IP	DNAT Port	Protocol	State
team03	vpn01	team01	*	-	-	*	NEW
team03	vpn02	team02	*	-	-	*	NEW
team03	vpn03	team03	*	-	-	*	NEW
team03	-	ext_ip	443 80	manager_ip	-	tcp	NEW
team03	-	ext_ip	2222	manager_ip	22	tcp	NEW
team03	*	*	*	*	-	*	ESTABLISHED

Case Study

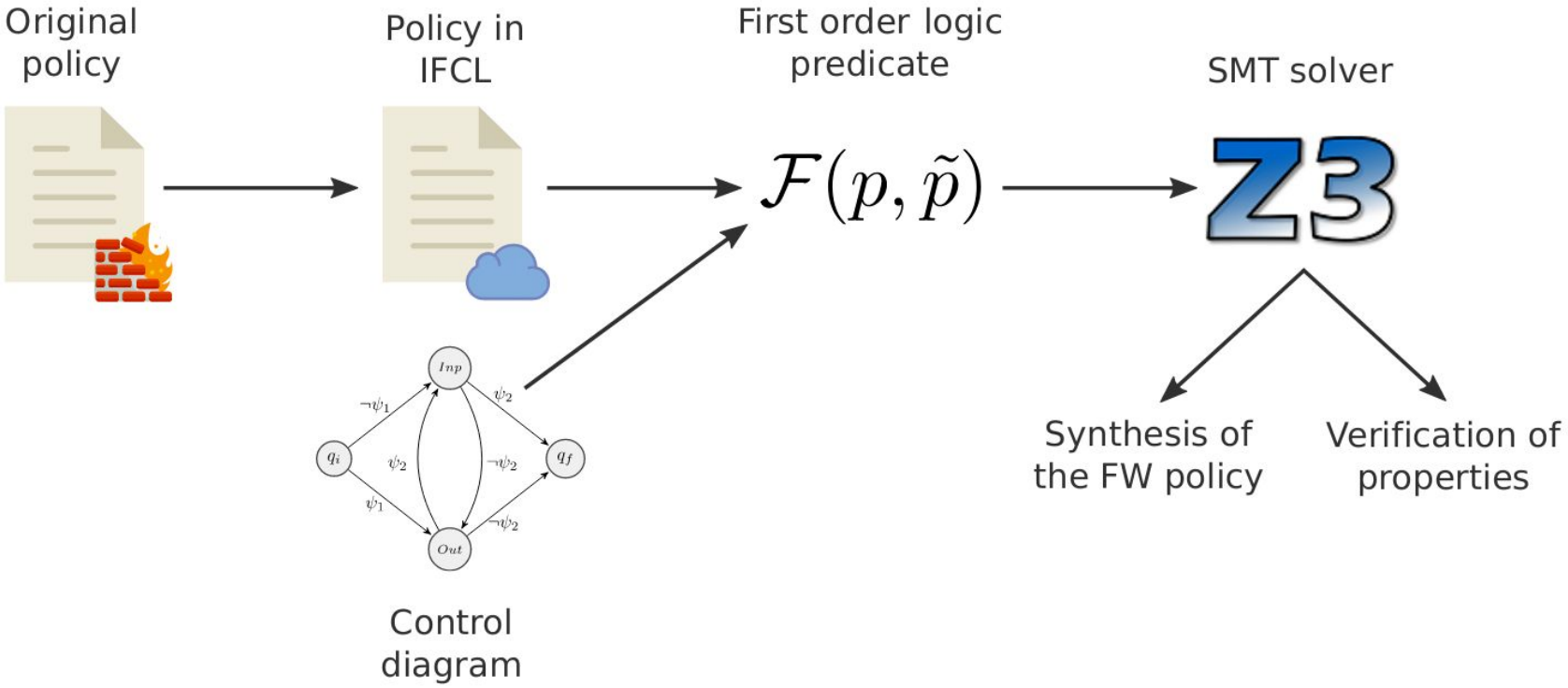
```
FWS> diff(policy, policy-closed)
       in forward
       where srcIp = team03
```

```
+++ iptables.rules
--- iptables_closed.rules
```

+/-	Source IP	Source Port	SNAT IP	Destination IP	Destination Port	State
-	team03	*	vpn01	team01	*	NEW
-	team03	*	vpn02	team02	*	NEW
-	team03	*	vpn03	team03	*	NEW

Theoretical Background

FWS: Overview of the approach



IFCL - Intermediate firewall language

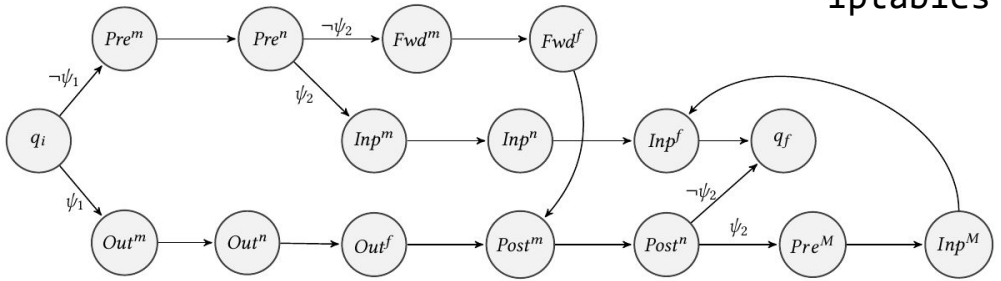
Supports NAT, Call/Jump, Stateful filters

Rulesets: list of rules applied to packets

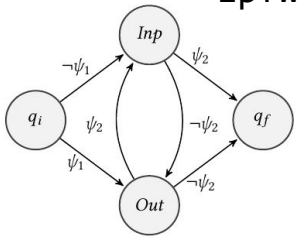
Chain Inp^{f} :
(state = 1, ACCEPT)
(protocol = icmp $\wedge$ dstPort = 1194, ACCEPT)
(protocol = tcp $\wedge$ dstPort = 80, DROP)

Control diagram: which rulesets are applied when processing packets

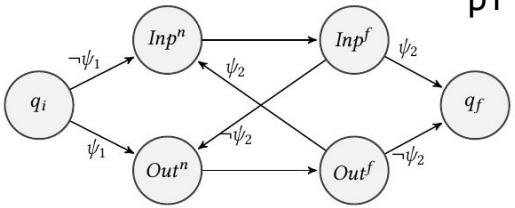
iptables



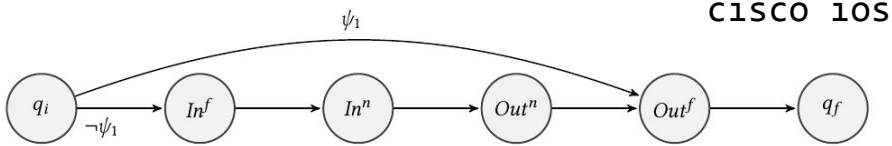
ipfw



pf



cisco ios



Solving firewalls as logic formulas

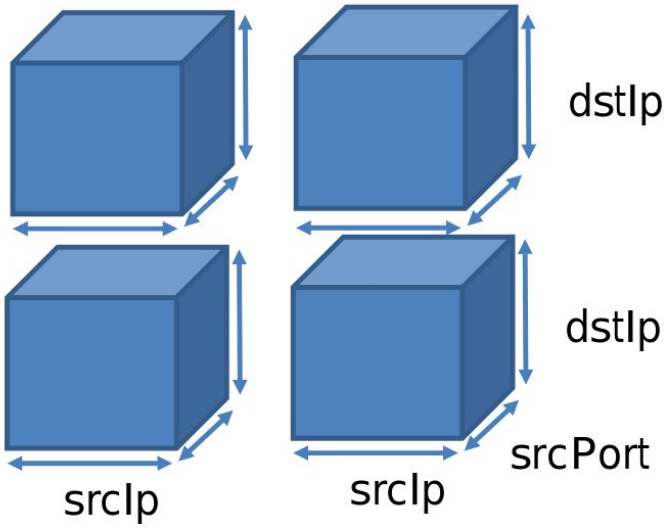
Packetsare tuples of Z3 **bit-vector** variables
`(srcIP, srcPort, dstIP, dstPort, protocol, state)`

Rule constraints are expressed as **logical formulas on the packet variables**

We extend the ALL-BV-SAT algorithm of *Jayaraman et al.* to work with NAT

The output is a set of **multi-cubes** that represent groups of accepted packets in a succinct way

$$\texttt{dstIp} = \{10.0.2.15\} \cup [10.0.1.0,\ 10.0.1.255],$$
$$\texttt{dstPort} = \{22\} \cup \{443\}$$



References

P. Adão, C. Bozzato, G. D. Rossi, R. Focardi, and F. L. Luccio,
Mignis: A semantic based tool for firewall configuration
in IEEE 27th Computer Security Foundations Symposium, CSF 2014.

C. Bodei, P. Degano, R. Focardi, L. Galletta, M. Tempesta, L. Veronese.
Language-Independent Synthesis of Firewall Policies.
In 3rd IEEE European Symposium on Security and Privacy (EuroS&P 2018).

https://github.com/secgroup/mignis

https://github.com/secgroup/fws

Thank You!

Lorenzo Veronese 310wert@gmail.com | 852058@stud.unive.it