# Verified Firewall Ruleset Verification

## with Isabelle/HOL

Cornelius Diekmann

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0    state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0    tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0    multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0    multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0    icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0    icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0    tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0    tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0    state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0    tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0    multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0    multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0    icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0    icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0    tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0    tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

# Introduction to Firewalls

```
Chain INPUT (policy ACCEPT)
target      prot source          destination
DOS_PROTECT all  0.0.0.0/0       0.0.0.0/0
ACCEPT      all  0.0.0.0/0       0.0.0.0/0   state RELATED,ESTABLISHED
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp dpt:22
DROP        tcp  0.0.0.0/0       0.0.0.0/0   multiport dports 21,873,5005,5006,80,548,...
DROP        udp  0.0.0.0/0       0.0.0.0/0   multiport dports 123,111,2049,892,5353
ACCEPT      all  192.168.0.0/16  0.0.0.0/0
DROP        all  0.0.0.0/0       0.0.0.0/0

Chain DOS_PROTECT (1 references)
target      prot source          destination
RETURN      icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8 limit: avg 1/sec burst 5
DROP        icmp 0.0.0.0/0       0.0.0.0/0   icmptype 8
RETURN      tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04 limit: avg 1/sec burst 5
DROP        tcp  0.0.0.0/0       0.0.0.0/0   tcp flags:0x17/0x04
```

## Problem

"*there are no good high-complexity rule sets*"

A. Wool, A Quantitative Study of Firewall Configuration Errors, Computer, IEEE, vol. 37, no. 6, pp. 62–67, Jun. 2004.

## Problem

"*there are no good high-complexity rule sets*"

"*firewalls are (still) poorly configured*"

A. Wool, A Quantitative Study of Firewall Configuration Errors, Computer, IEEE, vol. 37, no. 6, pp. 62–67, Jun. 2004.
A. Wool, Trends in Firewall Configuration Errors: Measuring the Holes in Swiss Cheese, Internet Computing, IEEE, vol. 14, no. 4, pp. 58–65, Jul. 2010.

## Problem

"*there are no good high-complexity rule sets*"

"*firewalls are (still) poorly configured*"

"*tools do not understand real-world firewall rules*"

A. Wool, A Quantitative Study of Firewall Configuration Errors, Computer, IEEE, vol. 37, no. 6, pp. 62–67, Jun. 2004.
A. Wool, Trends in Firewall Configuration Errors: Measuring the Holes in Swiss Cheese, Internet Computing, IEEE, vol. 14, no. 4, pp. 58–65, Jul. 2010.
C. Diekmann, L. Hupel, and G. Carle, Semantics-Preserving Simplification of Real-World Firewall Rule Sets, in Formal Methods (FM). Springer, pp. 195–212. Jun. 2015

# A Tool for Ruleset Verification

## Specification

- Documentation

## Implementation

- Code, tool
- Performance

# A Tool for Ruleset Verification

## Specification

- Documentation
- *What is a correct ruleset?*

## Implementation

- Code, tool
- Performance

# A Tool for Ruleset Verification

## Specification

- Documentation
- *What is a correct ruleset?*
- Goal: Spoofing protection

## Implementation

- Code, tool
- Performance

# A Tool for Ruleset Verification

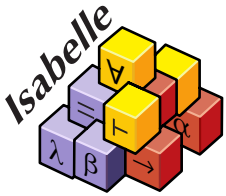
## Specification

- Documentation
- *What is a correct ruleset?*
- Goal: Spoofing protection
- Needs: Model of iptables

## Implementation

- Code, tool
- Performance

# A Tool for Ruleset Verification




## Specification

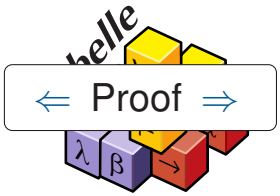
- Documentation
- *What is a correct ruleset?*
- Goal: Spoofing protection
- Needs: Model of iptables

## Implementation

- Code, tool
- Performance

`http://isabelle.in.tum.de/`

# A Tool for Ruleset Verification




## Specification

- Documentation
- *What is a correct ruleset?*
- Goal: Spoofing protection
- Needs: Model of iptables

## Implementation

- Code, tool
- Performance

`http://isabelle.in.tum.de/`

# A Tool for Ruleset Verification


## Specif tation

- Doc ol
- *Wh* nce
- Goa
- Nee

http://isabelle.in.tum.de/

# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

datatype *'a mexpr* = Match *'a* | MatchAny
| MatchNot *'a mexpr* | MatchAnd *'a mexpr* *'a mexpr*

# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

Polymorphic: arbitrary type *'a*

**datatype** *'a mexpr* = Match *'a* | MatchAny
| MatchNot *'a mexpr* | MatchAnd *'a mexpr* *'a mexpr*

# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

Polymorphic: arbitrary type *'a*

datatype *'a mexpr* = Match *'a* | MatchAny
| MatchNot *'a mexpr* | MatchAnd *'a mexpr* *'a mexpr*

# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

Polymorphic: arbitrary type *'a*

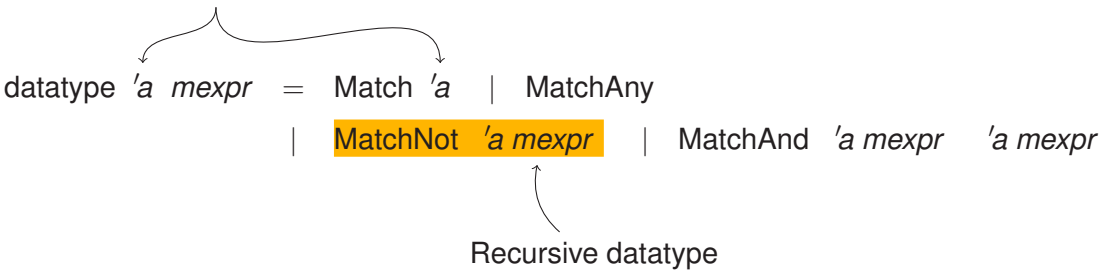
datatype *'a mexpr* = Match *'a* | MatchAny
| MatchNot *'a mexpr* | MatchAnd *'a mexpr* *'a mexpr*

# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

Polymorphic: arbitrary type *'a*

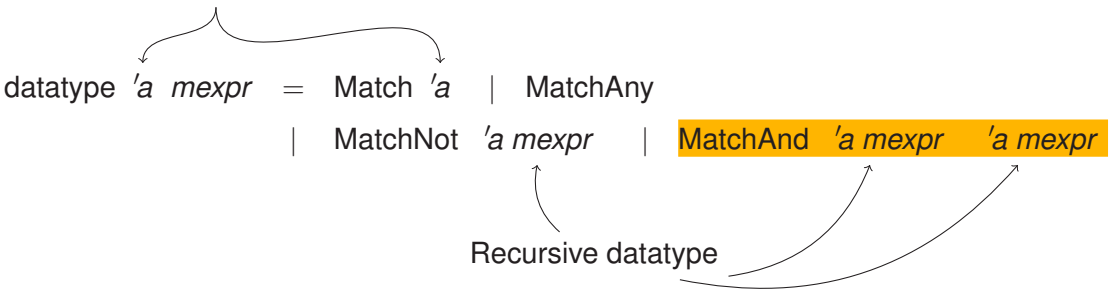


# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

Polymorphic: arbitrary type *'a*



# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

datatype *'a mexpr* = Match *'a* | MatchAny
| MatchNot *'a mexpr* | MatchAnd *'a mexpr* *'a mexpr*

Example:

MatchAnd (Match ( DstIP 8.8.8.8 )) (Match ( Protocol TCP ))

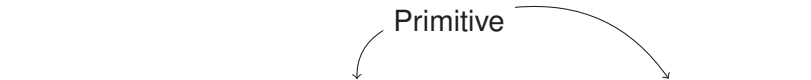
# Match Expressions: Syntax and Semantics

Syntax

- How to represent match expressions?

datatype *'a mexpr* = Match *'a* | MatchAny
| MatchNot *'a mexpr* | MatchAnd *'a mexpr* *'a mexpr*

Example:



# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

$$\begin{aligned}
&\text{matches } \gamma \text{ (Match } a\text{) } p && \longleftrightarrow \gamma\ a\ p \\
&\text{matches } _ \text{ MatchAny } _ && \longleftrightarrow \text{True} \\
&\text{matches } \gamma \text{ (MatchNot } m\text{) } p && \longleftrightarrow \neg\ \text{matches } \gamma\ m\ p \\
&\text{matches } \gamma \text{ (MatchAnd } m_1\ m_2\text{) } p && \longleftrightarrow \text{matches } \gamma\ m_1\ p\ \wedge\ \text{matches } \gamma\ m_2\ p
\end{aligned}$$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: ('a \Rightarrow 'p \Rightarrow \mathbb{B}) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

$$\begin{aligned}
&\text{matches } \gamma \text{ (Match } a\text{) } p && \longleftrightarrow \gamma\ a\ p \\
&\text{matches } _ \text{ MatchAny } _ && \longleftrightarrow \text{True} \\
&\text{matches } \gamma \text{ (MatchNot } m\text{) } p && \longleftrightarrow \neg\ \text{matches } \gamma\ m\ p \\
&\text{matches } \gamma \text{ (MatchAnd } m_1\ m_2\text{) } p && \longleftrightarrow \text{matches } \gamma\ m_1\ p\ \wedge\ \text{matches } \gamma\ m_2\ p
\end{aligned}$$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: (\text{'}a \Rightarrow \text{'}p \Rightarrow \mathbb{B}) \Rightarrow \text{'}a\ mexpr \Rightarrow \text{'}p \Rightarrow \mathbb{B}$$

matches $\gamma$ (Match $a$) $p$ $\longleftrightarrow \gamma\ a\ p$

matches _ MatchAny _ $\longleftrightarrow$ True

matches $\gamma$ (MatchNot $m$) $p$ $\longleftrightarrow \neg$ matches $\gamma\ m\ p$

matches $\gamma$ (MatchAnd $m_1\ m_2$) $p$ $\longleftrightarrow$ matches $\gamma\ m_1\ p\ \wedge$ matches $\gamma\ m_2\ p$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

$$\begin{aligned}
&\text{matches } \gamma \text{ (Match } a\text{) } p && \longleftrightarrow \gamma\ a\ p \\
&\text{matches } _ \text{ MatchAny } _ && \longleftrightarrow \text{True} \\
&\text{matches } \gamma \text{ (MatchNot } m\text{) } p && \longleftrightarrow \neg\ \text{matches } \gamma\ m\ p \\
&\text{matches } \gamma \text{ (MatchAnd } m_1\ m_2\text{) } p && \longleftrightarrow \text{matches } \gamma\ m_1\ p\ \wedge\ \text{matches } \gamma\ m_2\ p
\end{aligned}$$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: (\text{'}a \Rightarrow \text{'}p \Rightarrow \mathbb{B}) \Rightarrow \text{'}a\ mexpr \Rightarrow \text{'}p \Rightarrow \mathbb{B}$$

matches $\gamma$ (Match $a$) $p$ $\longleftrightarrow \gamma\ a\ p$

matches _ MatchAny _ $\longleftrightarrow$ True

matches $\gamma$ (MatchNot $m$) $p$ $\longleftrightarrow \neg$ matches $\gamma\ m\ p$

matches $\gamma$ (MatchAnd $m_1\ m_2$) $p$ $\longleftrightarrow$ matches $\gamma\ m_1\ p\ \wedge$ matches $\gamma\ m_2\ p$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

$$\begin{aligned}
&\text{matches } \gamma \text{ (Match } a\text{) } p && \longleftrightarrow \gamma\ a\ p \\
&\text{matches } _ \text{ MatchAny } _ && \longleftrightarrow \text{True} \\
&\text{matches } \gamma \text{ (MatchNot } m\text{) } p && \longleftrightarrow \neg\ \text{matches } \gamma\ m\ p \\
&\text{matches } \gamma \text{ (MatchAnd } m_1\ m_2\text{) } p && \longleftrightarrow \text{matches } \gamma\ m_1\ p\ \wedge\ \text{matches } \gamma\ m_2\ p
\end{aligned}$$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

$$\begin{aligned}
&\text{matches } \gamma \text{ (Match } a\text{) } p && \longleftrightarrow \gamma\ a\ p \\
&\text{matches } _ \text{ MatchAny } _ && \longleftrightarrow \text{True} \\
&\text{matches } \gamma \text{ (MatchNot } m\text{) } p && \longleftrightarrow \neg\ \text{matches } \gamma\ m\ p \\
&\text{matches } \gamma \text{ (MatchAnd } m_1\ m_2\text{) } p && \longleftrightarrow \text{matches } \gamma\ m_1\ p\ \wedge\ \text{matches } \gamma\ m_2\ p
\end{aligned}$$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

| | |
|---|---|
| matches $\gamma$ (Match $a$) $p$ | $\longleftrightarrow \gamma\ a\ p$ |
| matches _ MatchAny _ | $\longleftrightarrow$ True |
| matches $\gamma$ (MatchNot $m$) $p$ | $\longleftrightarrow \neg$ matches $\gamma\ m\ p$ |
| matches $\gamma$ (MatchAnd $m_1\ m_2$) $p$ | $\longleftrightarrow$ matches $\gamma\ m_1\ p \wedge$ matches $\gamma\ m_2\ p$ |

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

matches $\gamma$ (Match $a$) $p$ $\longleftrightarrow \gamma\ a\ p$

matches _ MatchAny _ $\longleftrightarrow$ True

matches $\gamma$ (MatchNot $m$) $p$ $\longleftrightarrow \neg$ matches $\gamma\ m\ p$

matches $\gamma$ (MatchAnd $m_1\ m_2$) $p \longleftrightarrow$ matches $\gamma\ m_1\ p \wedge$ matches $\gamma\ m_2\ p$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

matches $\gamma$ (Match $a$) $p$ $\longleftrightarrow \gamma\ a\ p$

matches _ MatchAny _ $\longleftrightarrow$ True

matches $\gamma$ (MatchNot $m$) $p$ $\longleftrightarrow \neg$ matches $\gamma\ m\ p$

matches $\gamma$ (MatchAnd $m_1\ m_2$) $p \longleftrightarrow$ matches $\gamma\ m_1\ p\ \wedge$ matches $\gamma\ m_2\ p$

# Match Expressions: Syntax and Semantics

## Semantics

- What do match expressions mean?

$$matches :: \left('a \Rightarrow 'p \Rightarrow \mathbb{B}\right) \Rightarrow 'a\ mexpr \Rightarrow 'p \Rightarrow \mathbb{B}$$

matches $\gamma$ (Match $a$) $p$ $\longleftrightarrow \gamma\ a\ p$

matches _ MatchAny _ $\longleftrightarrow$ True

matches $\gamma$ (MatchNot $m$) $p$ $\longleftrightarrow \neg$ matches $\gamma\ m\ p$

matches $\gamma$ (MatchAnd $m_1\ m_2$) $p \longleftrightarrow$ matches $\gamma\ m_1\ p\ \wedge$ matches $\gamma\ m_2\ p$

# Iptables Semantics: Filtering Behavior

$$\textsc{Skip}\ \frac{}{\gamma, p \vdash \langle [],\ t\rangle \Rightarrow t} \qquad \textsc{Accept}\ \frac{\text{matches}\ \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ?\rangle \Rightarrow \checkmark} \qquad \textsc{Drop}\ \frac{\text{matches}\ \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Drop})],\ ?\rangle \Rightarrow \times}$$

$$\textsc{Reject}\ \frac{\text{matches}\ \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Reject})],\ ?\rangle \Rightarrow \times} \qquad \textsc{NoMatch}\ \frac{\neg\ \text{matches}\ \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ a)],\ ?\rangle \Rightarrow ?} \qquad \textsc{Decision}\ \frac{t \neq ?}{\gamma, p \vdash \langle rs,\ t\rangle \Rightarrow t}$$

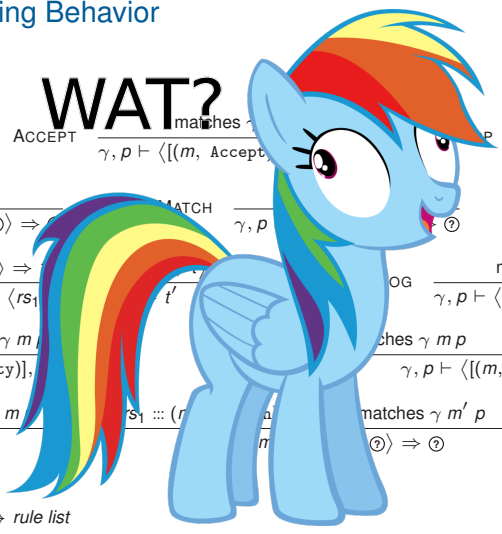
$$\textsc{Seq}\ \frac{\gamma, p \vdash \langle rs_1,\ ?\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 :::: rs_2,\ ?\rangle \Rightarrow t'} \qquad \textsc{Log}\ \frac{\text{matches}\ \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Log})],\ ?\rangle \Rightarrow ?}$$

$$\textsc{Empty}\ \frac{\text{matches}\ \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Empty})],\ ?\rangle \Rightarrow ?} \qquad \textsc{CallResult}\ \frac{\text{matches}\ \gamma\ m\ p \qquad \gamma, p \vdash \langle \Gamma\ c,\ ?\rangle \Rightarrow t}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)],\ ?\rangle \Rightarrow t}$$

$$\textsc{CallReturn}\ \frac{\text{matches}\ \gamma\ m\ p \qquad \Gamma\ c = rs_1 :::: (m',\ \texttt{Return}) :: rs_2 \qquad \text{matches}\ \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ?\rangle \Rightarrow ?}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)],\ ?\rangle \Rightarrow ?}$$

Background ruleset $\Gamma$ : *chain name* $\rightarrow$ *rule list*

# Iptables Semantics: Filtering Behavior




$$\textsc{Skip}\ \frac{}{\gamma, p \vdash \langle [],\ t\rangle \Rightarrow t}$$

$$\textsc{Accept}\ \frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ \textcircled{?}\rangle \Rightarrow \checkmark}$$

$$\textsc{Drop}\ \frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Drop})],\ \textcircled{?}\rangle \Rightarrow \text{✗}}$$

$$\textsc{Reject}\ \frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Reject})],\ \textcircled{?}\rangle \Rightarrow \text{✗}}$$

$$\textsc{Match}\ \frac{\gamma, p \vdash \langle rs,\ \textcircled{?}\rangle \Rightarrow \textcircled{?}}{\gamma, p \vdash \langle [(m,\ \texttt{Match}\ ms)] @ rs,\ \textcircled{?}\rangle \Rightarrow \textcircled{?}}$$

$$\textsc{Decision}\ \frac{t \neq \textcircled{?}}{\gamma, p \vdash \langle rs,\ t\rangle \Rightarrow t}$$

$$\textsc{Seq}\ \frac{\gamma, p \vdash \langle rs_1,\ \textcircled{?}\rangle \Rightarrow t' \qquad t' \neq \textcircled{?}}{\gamma, p \vdash \langle rs_1 @ rs_2,\ \textcircled{?}\rangle \Rightarrow t'}$$

$$\textsc{Log}\ \frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Log})],\ \textcircled{?}\rangle \Rightarrow \textcircled{?}}$$

$$\textsc{Empty}\ \frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Empty})],\ \textcircled{?}\rangle \Rightarrow \textcircled{?}}$$

$$\textsc{Call}\ \frac{\text{matches } \gamma\ m\ p \qquad \gamma, p \vdash \langle \Gamma\ c,\ \textcircled{?}\rangle \Rightarrow t}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)],\ \textcircled{?}\rangle \Rightarrow t}$$

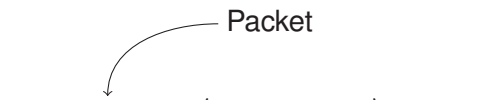
$$\textsc{CallReturn}\ \frac{\text{matches } \gamma\ m\ p \qquad \gamma, p \vdash \langle rs_1,\ \textcircled{?}\rangle \Rightarrow \textcircled{?} \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ \textcircled{?}\rangle \Rightarrow \textcircled{?}}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)] @ rs_1 @ [(m',\ \texttt{Return})] @ rs_2,\ \textcircled{?}\rangle \Rightarrow \textcircled{?}}$$

Background ruleset $\Gamma$ : *chain name* $\rightarrow$ *rule list*

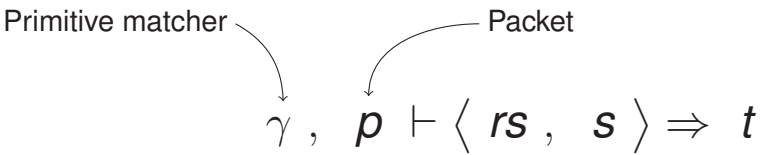
# Semantics Explained

$$\gamma\ ,\quad p \vdash \left\langle rs\ ,\quad s \right\rangle \Rightarrow\ t$$

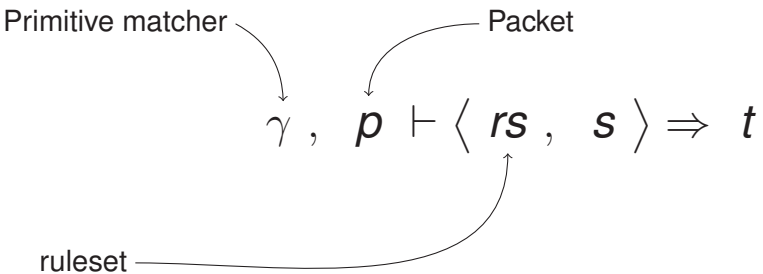
# Semantics Explained



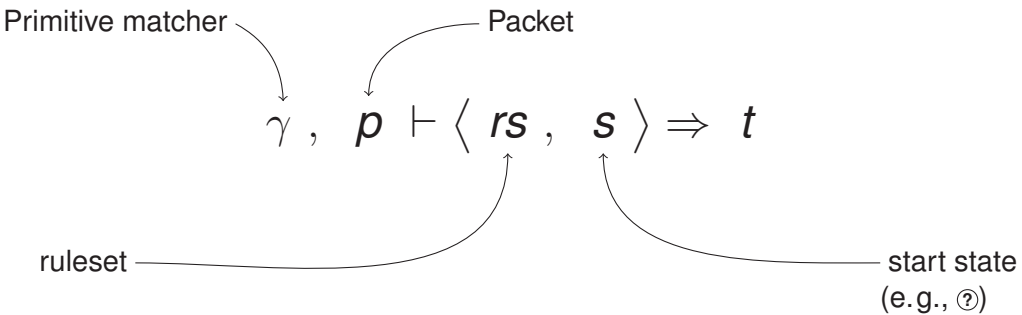
# Semantics Explained



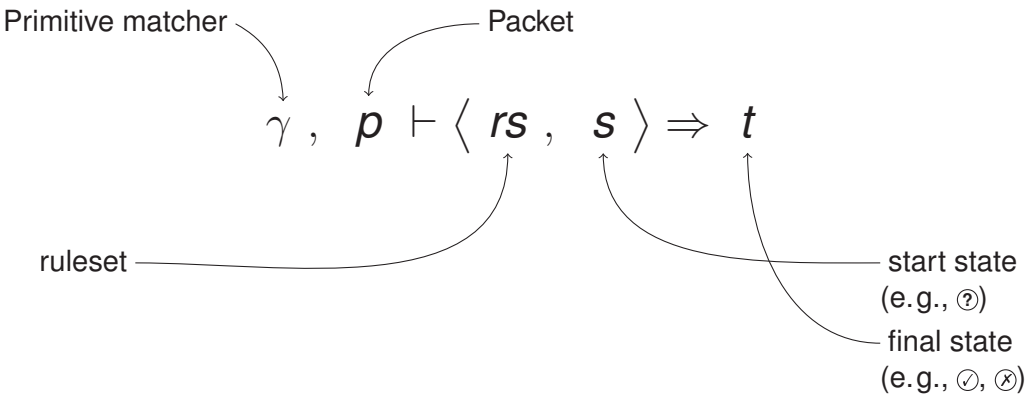
# Semantics Explained



# Semantics Explained



# Semantics Explained



# Semantics Explained: SKIP

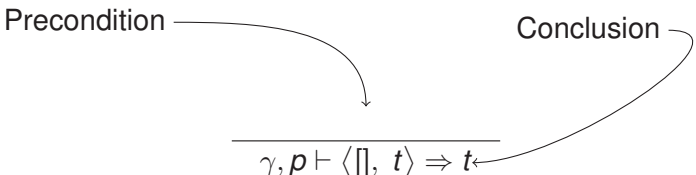
$$\frac{}{\gamma, p \vdash \langle [],\ t \rangle \Rightarrow t}$$

# Semantics Explained: SKIP

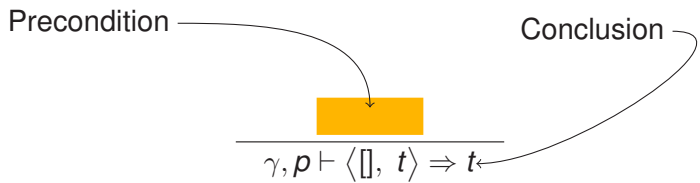
Precondition

$$\frac{}{\gamma, p \vdash \langle [],\ t \rangle \Rightarrow t}$$

# Semantics Explained: SKIP

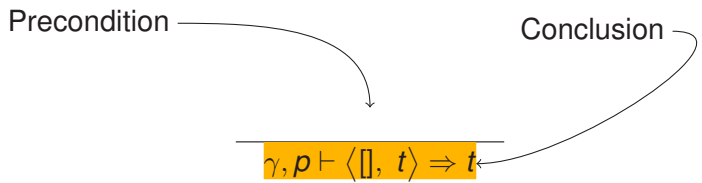


# Semantics Explained: SKIP




- no precondition
  - Holds unconditionally

# Semantics Explained: SKIP




- no precondition
  - Holds unconditionally
  - IF TRUE then $\gamma, p \vdash \langle [],\ t \rangle \Rightarrow t$

# Semantics Explained: SKIP

$$\frac{}{\gamma, p \vdash \langle [],\ t \rangle \Rightarrow t}$$

# Semantics Explained: SKIP

$$\gamma, p \vdash \langle [], t \rangle \Rightarrow t$$

- Empty Ruleset

# Semantics Explained: SKIP

$$\frac{}{\gamma, p \vdash \langle [],\ t \rangle \Rightarrow t}$$

- Empty Ruleset
- Start state equals final state

# Semantics Explained: SKIP

$$\frac{}{\gamma, p \vdash \langle [],\ t \rangle \Rightarrow t}$$

- Empty Ruleset
- Start state equals final state
- For the empty ruleset, the firewall does nothing

# Semantics Explained: ACCEPT

$$\frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ? \rangle \Rightarrow \checkmark}$$

# Semantics Explained: ACCEPT

$$\frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ?\rangle \Rightarrow \checkmark}$$

- Ruleset: single rule

# Semantics Explained: ACCEPT

$$\frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ? \rangle \Rightarrow \checkmark}$$

- Ruleset: single rule
- matches

# Semantics Explained: ACCEPT

$$\frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ? \rangle \Rightarrow \checkmark}$$

- Ruleset: single rule
- matches
- The action of the rule is `Accept` rule

# Semantics Explained: ACCEPT

$$\frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ?\rangle \Rightarrow \checkmark}$$

- Ruleset: single rule
- matches
- The action of the rule is `Accept` rule
- The firewall does not have a decision yet

# Semantics Explained: ACCEPT

$$\frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ? \rangle \Rightarrow \checkmark}$$

- Ruleset: single rule
- matches
- The action of the rule is `Accept` rule
- The firewall does not have a decision yet
- It will accept the packet

# Semantics Explained: ACCEPT

$$\frac{\text{matches } \gamma\ m\ p}{\gamma, p \vdash \langle [(m,\ \texttt{Accept})],\ ? \rangle \Rightarrow \checkmark}$$

- Ruleset: single rule
- matches
- The action of the rule is `Accept` rule
- The firewall does not have a decision yet
- It will accept the packet
- A matching `Accept` rule accepts packets

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ⑦\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ⑦\rangle \Rightarrow t'}$$

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ? \rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t \rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ? \rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ⑦\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ⑦\rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially
- First $rs_1$

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ⑦\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ⑦\rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially
- First $rs_1$
- Then $rs_2$

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ⓪\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 :::: rs_2,\ ⓪\rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially
- First $rs_1$
- Then $rs_2$
- To do so, first process $rs_1$, starting with ⓪ resulting in state $t$

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ⑦\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ⑦\rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially
- First $rs_1$
- Then $rs_2$
- To do so, first process $rs_1$, starting with ⑦ resulting in state $t$
- Then, use this state $t$ to continue processing with $rs_2$

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ?\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ?\rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially
- First $rs_1$
- Then $rs_2$
- To do so, first process $rs_1$, starting with ? resulting in state $t$
- Then, use this state $t$ to continue processing with $rs_2$

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ⑦\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ⑦\rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially
- First $rs_1$
- Then $rs_2$
- To do so, first process $rs_1$, starting with ⑦ resulting in state $t$
- Then, use this state $t$ to continue processing with $rs_2$
- Let this result in state $t'$

# Semantics Explained: SEQ

$$\frac{\gamma, p \vdash \langle rs_1,\ ?\rangle \Rightarrow t \qquad \gamma, p \vdash \langle rs_2,\ t\rangle \Rightarrow t'}{\gamma, p \vdash \langle rs_1 ::: rs_2,\ ?\rangle \Rightarrow t'}$$

- We want to process two rule lists sequentially
- First $rs_1$
- Then $rs_2$
- To do so, first process $rs_1$, starting with ? resulting in state $t$
- Then, use this state $t$ to continue processing with $rs_2$
- Let this result in state $t'$
- Ultimately, we have started with ? and processed $rs_1$ followed by $rs_2$ resulting in state $t'$

# Semantics Explained: CallReturn

$$\frac{\text{matches } \gamma\ m\ p \qquad \Gamma\ c = rs_1 :::: (m',\ \texttt{Return}) :: rs_2 \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ? \rangle \Rightarrow ?}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)],\ ? \rangle \Rightarrow ?}$$

# Semantics Explained: CallReturn

$$\frac{\text{matches } \gamma\ m\ p \qquad \Gamma\ c = rs_1 :::: (m',\ \texttt{Return}) :: rs_2 \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ⑦\rangle \Rightarrow ⑦}{\gamma, p \vdash \langle[(m,\ \texttt{Call}\ c)],\ ⑦\rangle \Rightarrow ⑦}$$

- matches

# Semantics Explained: CallReturn

$$\frac{\text{matches } \gamma\ m\ p \qquad \Gamma\ c = rs_1 :::: (m',\ \texttt{Return}) :: rs_2 \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ? \rangle \Rightarrow ?}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)],\ ? \rangle \Rightarrow ?}$$

- matches
- The called chain $c$ in the background ruleset $\Gamma$ is defined as $rs_1 :::: (m',\ \texttt{Return}) :: rs_2$

# Semantics Explained: CallReturn

$$\frac{\text{matches } \gamma\ m\ p \qquad \Gamma\ c = rs_1 \mathbin{:::} (m',\ \texttt{Return}) :: rs_2 \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ?\rangle \Rightarrow ?}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)],\ ?\rangle \Rightarrow ?}$$

- matches
- The called chain $c$ in the background ruleset $\Gamma$ is defined as $rs_1 \mathbin{:::} (m',\ \texttt{Return}) :: rs_2$
- First part $rs_1$ is processed without result

# Semantics Explained: CallReturn

$$\frac{\text{matches } \gamma\ m\ p \qquad \Gamma\ c = rs_1 :::: (m', \texttt{Return}) :: rs_2 \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ? \rangle \Rightarrow ?}{\gamma, p \vdash \langle [(m, \texttt{Call}\ c)],\ ? \rangle \Rightarrow ?}$$

- matches
- The called chain $c$ in the background ruleset $\Gamma$ is defined as $rs_1 :::: (m', \texttt{Return}) :: rs_2$
- First part $rs_1$ is processed without result
- Then there is a matching `Return`

# Semantics Explained: CallReturn

$$\frac{\text{matches } \gamma\ m\ p \qquad \Gamma\ c = rs_1 ::: (m',\ \texttt{Return}) :: rs_2 \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ? \rangle \Rightarrow ?}{\gamma, p \vdash \langle [(m,\ \texttt{Call}\ c)],\ ? \rangle \Rightarrow ?}$$

- matches
- The called chain $c$ in the background ruleset $\Gamma$ is defined as $rs_1 ::: (m',\ \texttt{Return}) :: rs_2$
- First part $rs_1$ is processed without result
- Then there is a matching `Return`
- Calling to user-defined chain and return without result

## Semantics Explained: CallReturn


$$\frac{\text{matches } \gamma\ m\ p \qquad \Gamma\ c = rs_1 :::\ (\ldots) :: rs_2 \qquad \text{matches } \gamma\ m'\ p \qquad \gamma, p \vdash \langle rs_1,\ ?\rangle \Rightarrow ?}{\ldots\ \texttt{all}\ c)],\ ?\rangle \Rightarrow ?}$$

- matches
- The called chain $c$ in the ba… d ruleset $\Gamma$ is defined as $rs_1 ::: (m', \texttt{Return}) :: rs_2$
- First part $rs_1$ is processe…
- Then there is a matching …
- Calling to user-defined chai… ut result

## Semantics-Preserving Simplification

$$\gamma, p \vdash \langle rs,\ s \rangle \Rightarrow t \quad \textit{iff} \quad \gamma, p \vdash \langle \mathrm{f}\ rs,\ s \rangle \Rightarrow t$$

# Semantics-Preserving Simplification

$$\gamma, p \vdash \langle rs,\ s \rangle \Rightarrow t \quad \textit{iff} \quad \gamma, p \vdash \langle \mathsf{f}\ rs,\ s \rangle \Rightarrow t$$

- $f :: 'a\ ruleset \Rightarrow 'a\ ruleset$

# Semantics-Preserving Simplification

$$\gamma, p \vdash \langle rs,\ s \rangle \Rightarrow t \quad \textit{iff} \quad \gamma, p \vdash \langle \mathsf{f}\ rs,\ s \rangle \Rightarrow t$$

- $f :: 'a\ ruleset \Rightarrow 'a\ ruleset$
- $f$ does not change filtering behavior of firewall

# Semantics-Preserving Simplification

$$\gamma, p \vdash \langle rs,\ s \rangle \Rightarrow t \quad \textit{iff} \quad \gamma, p \vdash \langle \mathsf{f}\ rs,\ s \rangle \Rightarrow t$$

- $f :: 'a\ ruleset \Rightarrow 'a\ ruleset$
- $f$ does not change filtering behavior of firewall
- Removing `Log` rules

# Semantics-Preserving Simplification

$$\gamma, p \vdash \langle rs,\ s \rangle \Rightarrow t \quad \textit{iff} \quad \gamma, p \vdash \langle \mathsf{f}\ rs,\ s \rangle \Rightarrow t$$

- $f :: 'a\ ruleset \Rightarrow 'a\ ruleset$
- $f$ does not change filtering behavior of firewall

- Removing `Log` rules
- Unfolding of user-defined chains

# Semantics-Preserving Simplification

$$\gamma, p \vdash \langle rs,\ s\rangle \Rightarrow t \quad \textit{iff} \quad \gamma, p \vdash \langle \mathsf{f}\ rs,\ s\rangle \Rightarrow t$$

- $f :: 'a\ ruleset \Rightarrow 'a\ ruleset$
- $f$ does not change filtering behavior of firewall

- Removing `Log` rules
- Unfolding of user-defined chains
- Normalizing match expressions, ...

# Embedding in Ternary Logic

$\mathbb{B} = \{\mathit{True}, \mathit{False}\}$ $\quad$ $\mathit{Ternary} = \{\mathit{True}, \mathit{False}, \mathit{Unknown}\}$

$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ stricter } \mathit{rs} = \text{✓}\Big\}$$
$$\subseteq$$
$$\{p \mid \gamma, p \vdash \langle \mathit{rs}, \text{?} \rangle \Rightarrow \text{✓}\}$$
$$\subseteq$$
$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ permissive } \mathit{rs} = \text{✓}\Big\}$$

# Embedding in Ternary Logic

$\mathbb{B} = \{\mathit{True}, \mathit{False}\}$ $\quad$ $\mathit{Ternary} = \{\mathit{True}, \mathit{False}, \mathit{Unknown}\}$

$$\left\{p \mid \text{approx\_firewall } \gamma \text{ stricter } rs = \textcircled{✓}\right\}$$
$$\subseteq$$
$$\{p \mid \gamma, p \vdash \langle rs, \textcircled{?}\rangle \Rightarrow \textcircled{✓}\}$$
$$\subseteq$$
$$\left\{p \mid \text{approx\_firewall } \gamma \text{ permissive } rs = \textcircled{✓}\right\}$$

- Set of packets accepted by the firewall

# Embedding in Ternary Logic

$\mathbb{B} = \{\mathit{True}, \mathit{False}\}$ $\quad$ $\mathit{Ternary} = \{\mathit{True}, \mathit{False}, \mathit{Unknown}\}$

$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ stricter } \mathit{rs} = \text{✓}\Big\}$$
$$\subseteq$$
$$\{p \mid \gamma, p \vdash \langle \mathit{rs}, \text{?} \rangle \Rightarrow \text{✓}\}$$
$$\subseteq$$
$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ permissive } \mathit{rs} = \text{✓}\Big\}$$

- Set of packets accepted by the firewall

# Embedding in Ternary Logic

$\mathbb{B} = \{\mathit{True}, \mathit{False}\}$ $\quad$ $\mathit{Ternary} = \{\mathit{True}, \mathit{False}, \mathit{Unknown}\}$

$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ stricter } rs = \textcircled{\checkmark}\Big\}$$
$$\subseteq$$
$$\{p \mid \gamma, p \vdash \langle rs, \textcircled{?}\rangle \Rightarrow \textcircled{\checkmark}\}$$
$$\subseteq$$
$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ permissive } rs = \textcircled{\checkmark}\Big\}$$

- Set of packets accepted by the firewall

# Embedding in Ternary Logic

$\mathbb{B} = \{\mathit{True}, \mathit{False}\}$ $\quad$ $\mathit{Ternary} = \{\mathit{True}, \mathit{False}, \mathit{Unknown}\}$

$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ stricter } \mathit{rs} = \text{✓}\Big\}$$
$$\subseteq$$
$$\{p \mid \gamma, p \vdash \langle \mathit{rs}, \text{?} \rangle \Rightarrow \text{✓}\} \longleftarrow \text{not executable}$$
$$\subseteq$$
$$\Big\{p \mid \text{approx\_firewall } \gamma \text{ permissive } \mathit{rs} = \text{✓}\Big\}$$

- Set of packets accepted by the firewall
- We can specify a lot . . .

## Embedding in Ternary Logic

$\mathbb{B} = \{\mathit{True}, \mathit{False}\}$ $\quad$ $\mathit{Ternary} = \{\mathit{True}, \mathit{False}, \mathit{Unknown}\}$

executable

$$\left\{p \mid \text{approx\_firewall } \gamma \text{ stricter } rs = \text{✓}\right\}$$

$$\subseteq$$

$$\{p \mid \gamma, p \vdash \langle rs, \text{?}\rangle \Rightarrow \text{✓}\}$$ ← not executable

$$\subseteq$$

$$\left\{p \mid \text{approx\_firewall } \gamma \text{ permissive } rs = \text{✓}\right\}$$

executable

- Set of packets accepted by the firewall
- We can specify a lot . . . but we also believe in running code

## Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth0} \wedge \gamma, p \vdash \langle rs, ⍰ \rangle \Rightarrow ✓\} \subseteq 192.168.0.0/24$$

# Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth0} \wedge \gamma, p \vdash \langle rs, ? \rangle \Rightarrow \checkmark\} \subseteq 192.168.0.0/24$$

# Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth0} \wedge \gamma, p \vdash \langle rs, ② \rangle \Rightarrow ✓\} \subseteq 192.168.0.0/24$$

# Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth0} \wedge \gamma, p \vdash \langle rs, ⍰ \rangle \Rightarrow ✓\} \subseteq 192.168.0.0/24$$

# Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth0} \wedge \gamma, p \vdash \langle rs, ② \rangle \Rightarrow ✓\} \subseteq 192.168.0.0/24$$

# Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth0} \wedge \gamma, p \vdash \langle rs, ⍰ \rangle \Rightarrow ✓\} \subseteq 192.168.0.0/24$$

## Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\forall \texttt{eth} \in ipassmt.\text{keys}$$
$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth} \wedge \gamma, p \vdash \langle rs,\ ? \rangle \Rightarrow \checkmark\} \subseteq ipassmt.\text{get}(\texttt{eth})$$

## Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

Spoofing Protection:

$$\text{check\_spoofing\_protection } ipassmt\ rs \longrightarrow$$
$$\forall \texttt{eth} \in ipassmt.\text{keys}$$
$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth} \wedge \gamma, p \vdash \langle rs,\ ? \rangle \Rightarrow \checkmark\} \subseteq ipassmt.\text{get}(\texttt{eth})$$

## Spoofing Protection

*ipassmt* :: *interface* $\Rightarrow$ *IP set*
Example:

$$ipassmt = [\texttt{eth0} \mapsto 192.168.0.0/24]$$

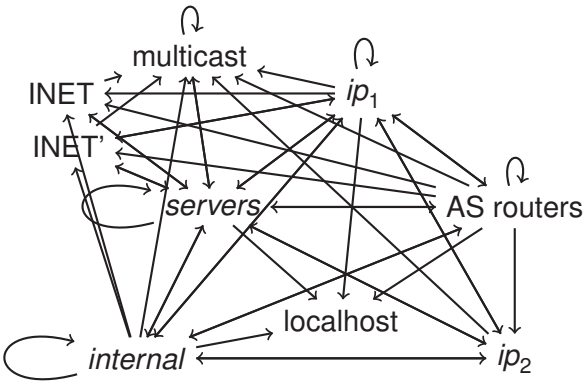
Spoofing Protection:

$$\text{check\_spoofing\_protection } ipassmt\ rs \longrightarrow$$
$$\forall \texttt{eth} \in ipassmt.\text{keys}$$
$$\{p.\text{src\_ip} \mid p.\text{in\_iface} = \texttt{eth} \wedge \gamma, p \vdash \langle rs,\ ? \rangle \Rightarrow \checkmark\} \subseteq ipassmt.\text{get}(\texttt{eth})$$

diekmann@xps12: ~/git/iptables_Semantics/beta_haskell_tool

```
$ iptables-save | ./check_spoofing_protection ipassmt_tumi8
loading ipassmt from `ipassmt_tumi8'
Parsed IpAssmt
sanity_wf_ruleset passed
Parsed 90 chains in table filter, a total of 4813 rules
Table `nat' : `Reading ruleset failed! sanity_wf_ruleset check failed.'.
Table `raw' : `Reading ruleset failed! sanity_wf_ruleset check failed.'.
sanity_wf_ruleset passed
(eth0,True)
(foo,False)
(eth1.96,True)
(eth1.108,True)
(eth1.109,True)
(eth1.110,False)
(eth1.116,True)
(eth1.152,True)
(eth1.171,True)
(eth1.173,True)
(eth1.1010,True)
(eth1.1011,True)
```

# Service Matrix




- Partitions complete IPv4 space
- All IP addresses in each group have same access rights
- Cannot be compressed any further

# Sources

Firewall Rulesets

`https://github.com/diekmann/net-network`

Isabelle Theories + Haskell Tool:

`https://github.com/diekmann/Iptables_Semantics`