
# Formal Methods to the Rescue?
## Experiences Coding in a Theorem Prover

https://github.com/diekmann/ @popitter_net

by

Cornelius Diekmann

April 9, 2018

# About Me

# About Me

- I did this work as PhD student at TUM
  - Which was awesome!
- Now I work at Google
  - Which is awesome, too!
  - This presentation is not related in any way to Google

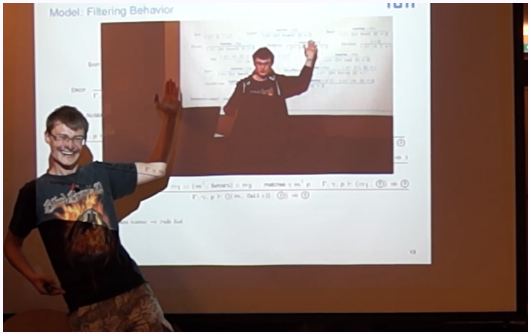



Fig 1.: Me pointing at slides pointing at slides.

Thoughts and opinions are my own, not those of my company.

# Linux iptables by Example

# iptables Example

```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j GOOD~STUFF
-A FORWARD -p tcp -m multiport ! --dports 80,443,6667,6697 -m hashlimit ↩
    --hashlimit-above 10/sec --hashlimit-burst 20 --hashlimit-mode srcip ↩
    --hashlimit-name aflood --hashlimit-srcmask 8 -j LOG
-A FORWARD ! -i lo -s 127.0.0.0/8 -j DROP
-A FORWARD -i internal -s 131.159.21.0/24 -j ACCEPT
-A FORWARD -s 131.159.15.240/28 -d 131.159.21.0/24 -j DROP
-A FORWARD -p tcp -d 131.159.15.240/28 -j ACCEPT
-A FORWARD -i [illegible] -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```

# iptables Example

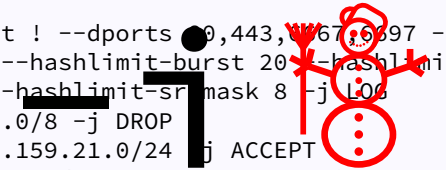
```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j GOOD~STUFF
-A FORWARD -p tcp -m multiport ! --dports 80,443,6667,6697 -m hashlimit ↩
    --hashlimit-above 10/sec --hashlimit-burst 20 --hashlimit-mode srcip ↩
    --hashlimit-name aflood --hashlimit-srcmask 8 -j LOG
-A FORWARD ! -i lo -s 127.0.0.0/8 -j DROP
-A FORWARD -i internal -s 131.159.21.0/24 -j ACCEPT
-A FORWARD -s 131.159.15.240/28 -d 131.159.21.0/24 -j DROP
-A FORWARD -p tcp -d 131.159.15.240/28 -j ACCEPT
-A FORWARD -i  -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```

# iptables Example

```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j GOOD~STUFF
-A FORWARD -p tcp -m multiport ! --dports 80,443,6667,6697 -m hashlimit ↩
    --hashlimit-above 10/sec --hashlimit-burst 20 --hashlimit-mode srcip ↩
    --hashlimit-name aflood --hashlimit-srcmask 8 -j LOG
-A FORWARD ! -i lo -s 127.0.0.0/8 -j DROP
-A FORWARD -i internal -s 131.159.21.0/24 -j ACCEPT
-A FORWARD -s 131.159.15.240/28 -d 131.159.21.0/24 -j DROP
-A FORWARD -p tcp -d 131.159.15.240/28 -j ACCEPT
-A FORWARD -i  -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```

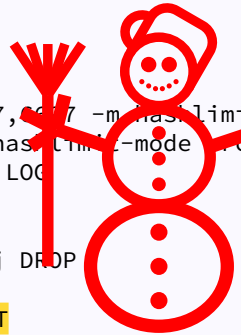
# iptables Example

```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j GOOD~STUFF
-A FORWARD -p tcp -m multiport ! --dports 80,443,[illegible]6697 -m hashlimit ↩
    --hashlimit-above 10/sec --hashlimit-burst 20 --hashlimit-mode srcip ↩
    --hashlimit-name aflood --hashlimit-srcmask 8 -j LOG
-A FORWARD ! -i lo -s 127.0.0.0/8 -j DROP
-A FORWARD -i internal -s 131.159.21.0/24 -j ACCEPT
-A FORWARD -s 131.159.15.240/28 -d 131.159.21.0/24 -j DROP
-A FORWARD -p tcp -d 131.159.15.240/28 -j ACCEPT
-A FORWARD -i [illegible] -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```



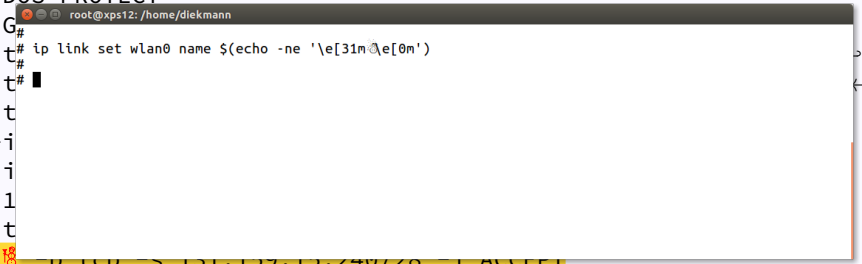
# iptables Example

```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j GOOD~STUFF
-A FORWARD -p tcp -m multiport ! --dports 80,443,6667,6697 -m hashlimit ↩
    --hashlimit-above 10/sec --hashlimit-burst 20 --hashlimit-mode srcip ↩
    --hashlimit-name aflood --hashlimit-srcmask 8 -j LOG
-A FORWARD ! -i l[redacted].0/8 -j DROP
-A FORWARD -i internal -s 131.159.21.0/24 -j ACCEPT
-A FORWARD -s 131.159.15.240/28 -d 131.159.21.0/24 -j DROP
-A FORWARD -p tcp -d 131.159.15.240/28 -j ACCEPT
-A FORWARD -i [redacted] -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```



# iptables Example

```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j G
-A FORWARD -p t
    --hashlimit
    --hashlimit
-A FORWARD ! -i
-A FORWARD -i i
-A FORWARD -s 1
-A FORWARD -p t
-A FORWARD -i [redacted] -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```



# iptables Example

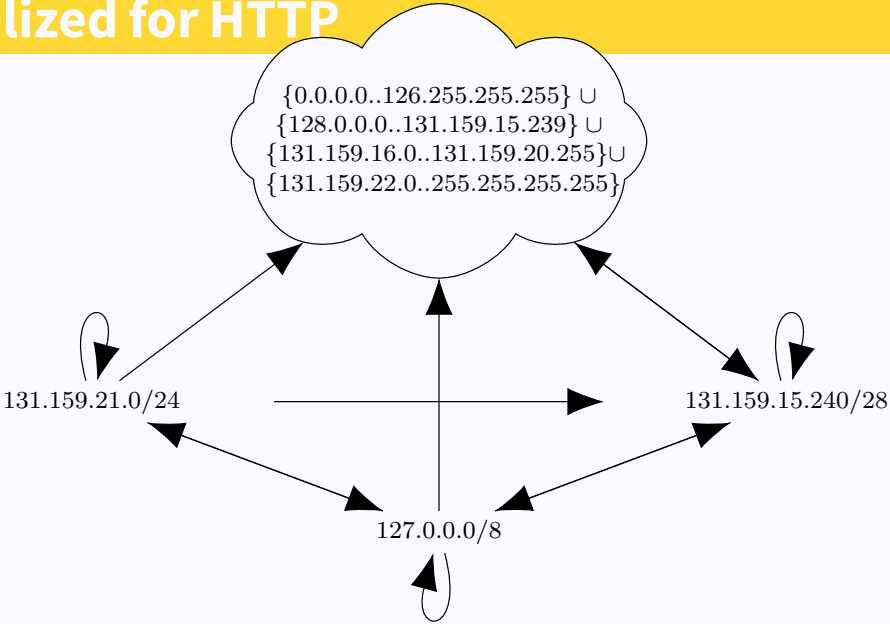
```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j G
-A FORWARD -p t
    --hashlimit
    --hashlimit
-A FORWARD ! -i
-A FORWARD -i i
-A FORWARD -s 1
-A FORWARD -p t
-A FORWARD -i                -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```

```
root@xps12: /home/diekmann
#
# ip link set wlan0 name $(echo -ne '\e[31m\e[0m')
#
# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2:  : <BROADCAST,MULTICAST> mtu 1500 qdisc mq state DOWN mode DEFAULT group default qlen 1000
    link/ether e8:2a:ea:44:1a:c4 brd ff:ff:ff:ff:ff:ff
#
```

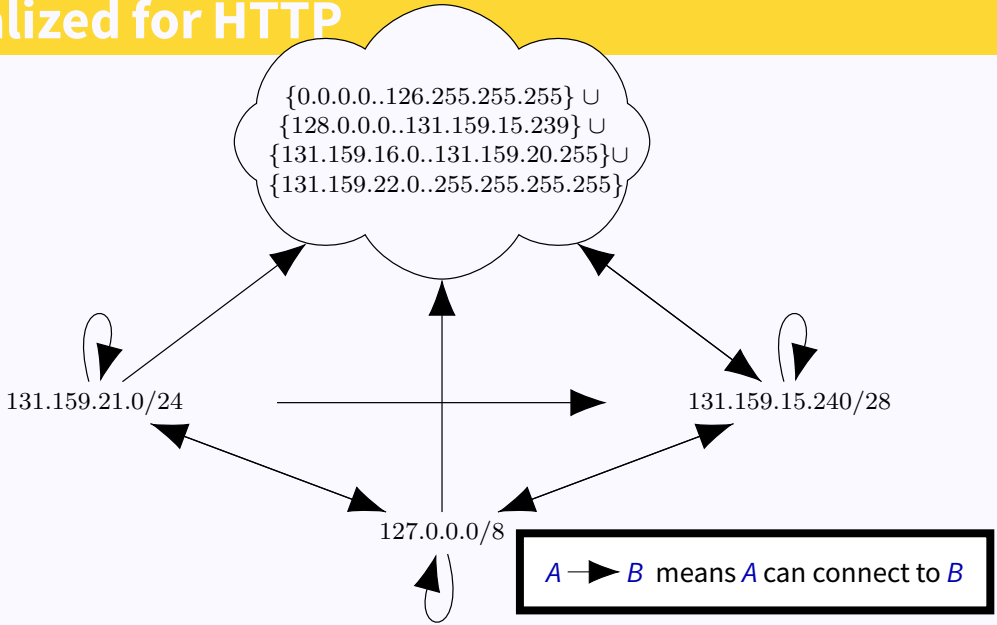
# iptables Example

```
*filter
:FORWARD DROP [0:0]
:DOS_PROTECT - [0:0]
:GOOD~STUFF - [0:0]
-A FORWARD -j DOS_PROTECT
-A FORWARD -j GOOD~STUFF
-A FORWARD -p tcp -m multiport ! --dports 80,443,6667,6697 -m hashlimit ↩
    --hashlimit-above 10/sec --hashlimit-burst 20 --hashlimit-mode srcip ↩
    --hashlimit-name aflood --hashlimit-srcmask 8 -j LOG
-A FORWARD ! -i lo -s 127.0.0.0/8 -j DROP
-A FORWARD -i internal -s 131.159.21.0/24 -j ACCEPT
-A FORWARD -s 131.159.15.240/28 -d 131.159.21.0/24 -j DROP
-A FORWARD -p tcp -d 131.159.15.240/28 -j ACCEPT
-A FORWARD -i [illegible] -p tcp -s 131.159.15.240/28 -j ACCEPT
-A GOOD~STUFF -i lo -j ACCEPT
-A GOOD~STUFF -m state --state ESTABLISHED -j ACCEPT
-A GOOD~STUFF -p icmp -m state --state RELATED -j ACCEPT
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 ... --limit 1/sec -j RETURN
-A DOS_PROTECT -i eth1 -p icmp -m icmp --icmp-type 8 -j DROP
COMMIT
```

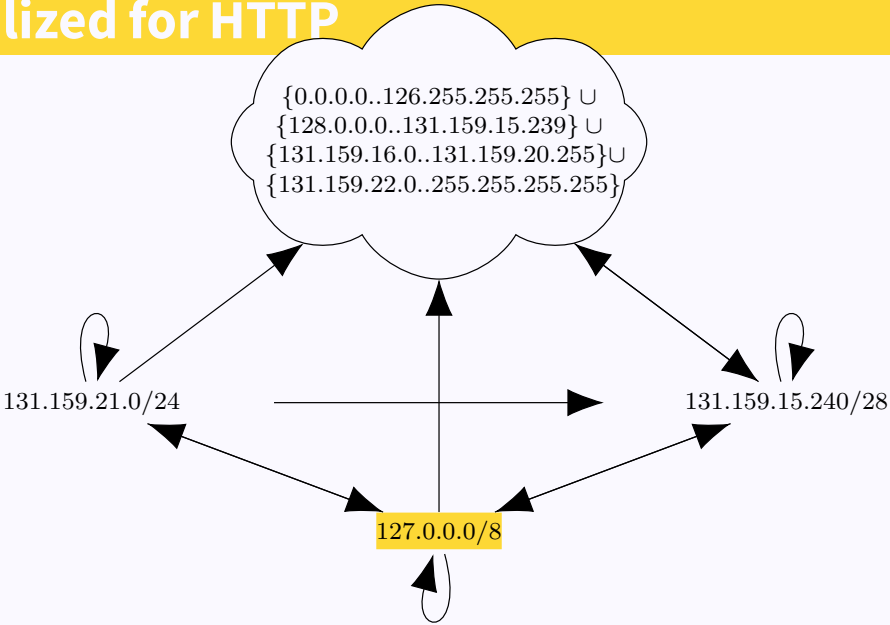
# Visualized for HTTP



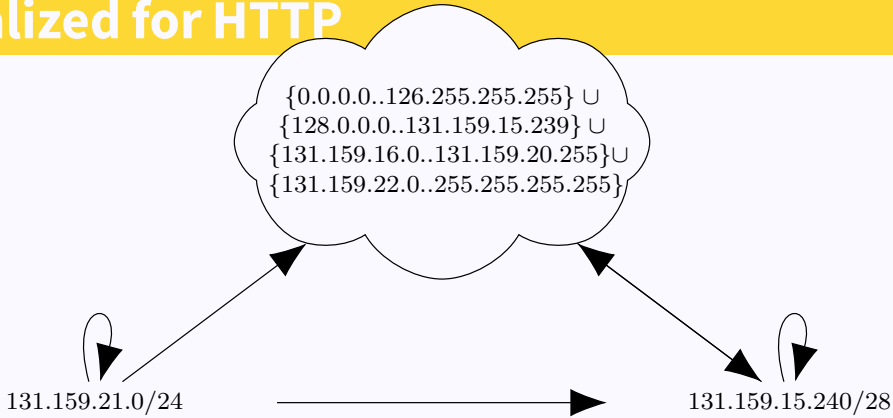
# Visualized for HTTP



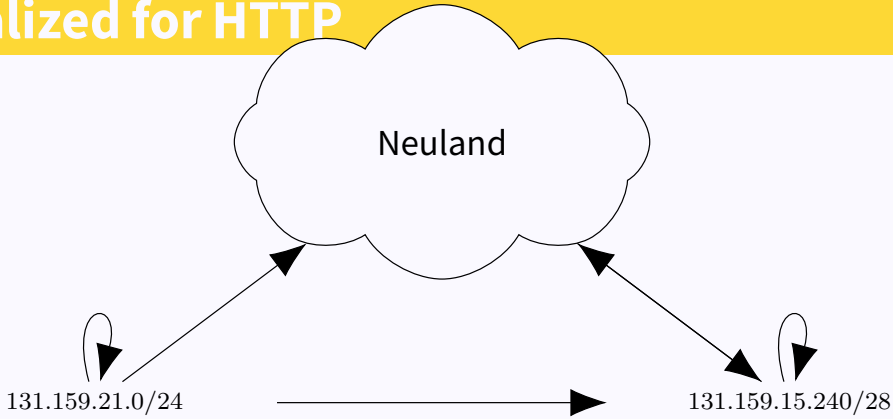
# Visualized for HTTP



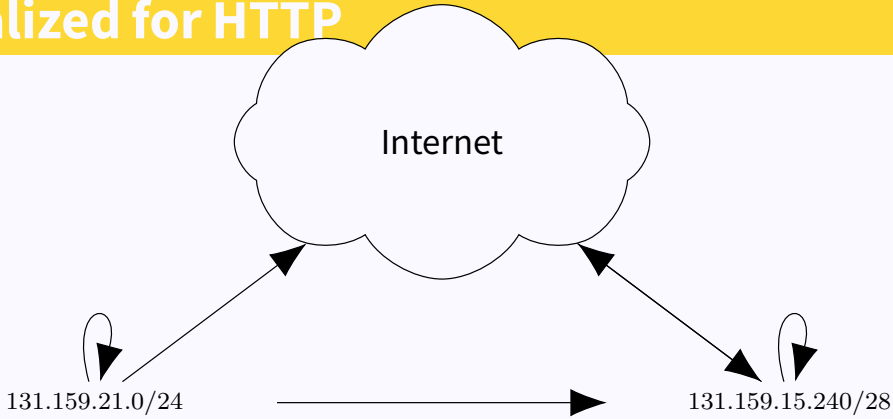
# Visualized for HTTP



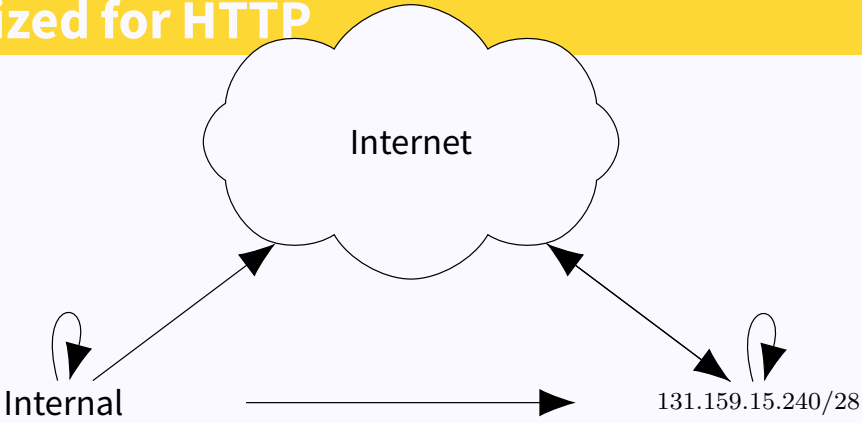
# Visualized for HTTP



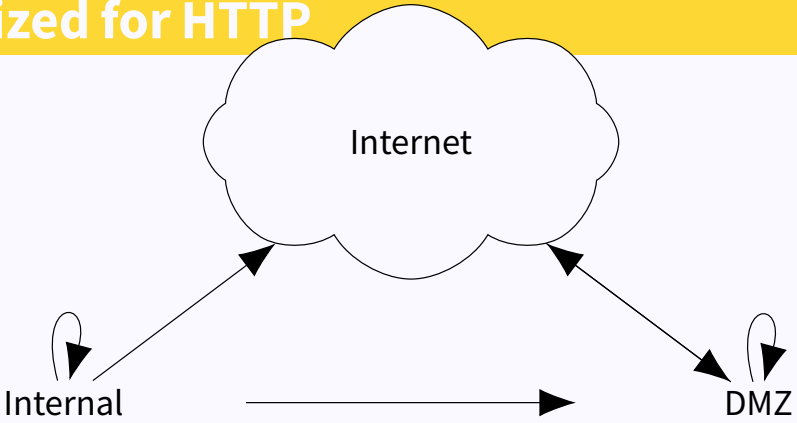
# Visualized for HTTP



# Visualized for HTTP



# Visualized for HTTP



# About this Tool

# Bold Claim

- I coded a tool to print this visualization

# Bold Claim

- I coded a tool to print this visualization
  - Okay, TikZ and Graphviz draw the image, but my tool computes the graph

# Bold Claim

- I coded a tool to print this visualization
  - Okay, TikZ and Graphviz draw the image, but my tool computes the graph

If the graph looks good to you, your firewall configuration is secure!

# How to Specify Correctness?

# Correctness Theorem!

Assumes

- Unfolded $\mathrm{rs}$ for $\Gamma$
- $\mathrm{p}$ is NEW
- $\Gamma, \gamma, p \vdash \langle \mathrm{rs},\ ⍰ \rangle \Rightarrow ✓$

- Let $(V, E) = \mathsf{compute\_graph}\ (\mathsf{prot}\ p, \mathsf{sport}\ p, \mathsf{dport}\ p)\ (\mathsf{simplify}\ \mathrm{rs})$

Shows

$$\begin{aligned}
&\exists \mathrm{s_{repr}}\ \mathrm{d_{repr}}\ \mathrm{s_{range}}\ \mathrm{d_{range}}.\ (\mathrm{s_{repr}}, \mathrm{d_{repr}}) \in \mathrm{E} \wedge \\
&\quad (\mathsf{map_of}\ \mathrm{V})\ \mathrm{s_{repr}} = \mathsf{Some}\ \mathrm{s_{range}} \wedge (\mathsf{src}\ \mathrm{p}) \in \mathrm{s_{range}} \wedge \\
&\quad (\mathsf{map_of}\ \mathrm{V})\ \mathrm{d_{repr}} = \mathsf{Some}\ \mathrm{d_{range}} \wedge (\mathsf{dst}\ \mathrm{p}) \in \mathrm{d_{range}}
\end{aligned}$$

# Correctness Theorem!

Assumes

- $\Gamma, \gamma, p \vdash \langle \mathbf{rs},\ ? \rangle \Rightarrow \checkmark$
- Let $(V, E) = \mathsf{compute\_graph}\ (\mathsf{prot}\ p, \mathsf{sport}\ p, \mathsf{dport}\ p)\ (\mathsf{simplify}\ \mathbf{rs})$

Shows

$$\begin{aligned}
&\exists s_{\mathrm{repr}}\ d_{\mathrm{repr}}\ s_{\mathrm{range}}\ d_{\mathrm{range}}.\ (s_{\mathrm{repr}}, d_{\mathrm{repr}}) \in \mathbf{E} \wedge \\
&\quad (\mathsf{map_of}\ \mathbf{V})\ s_{\mathrm{repr}} = \mathsf{Some}\ s_{\mathrm{range}} \ \wedge\ (\mathsf{src}\ \mathrm{p}) \in s_{\mathrm{range}} \wedge \\
&\quad (\mathsf{map_of}\ \mathbf{V})\ d_{\mathrm{repr}} = \mathsf{Some}\ d_{\mathrm{range}} \ \wedge\ (\mathsf{dst}\ \mathrm{p}) \in d_{\mathrm{range}}
\end{aligned}$$

# Correctness Theorem!

Assumes

- $\Gamma, \gamma, p \vdash \langle \mathbf{rs},\ ? \rangle \Rightarrow \checkmark$
- Let $(V, E) = \text{compute\_graph}\ (\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)\ (\text{simplify}\ \mathbf{rs})$

Shows

$$\begin{aligned}
&\exists s_{\text{repr}}\ d_{\text{repr}}\ s_{\text{range}}\ d_{\text{range}}.\ (s_{\text{repr}}, d_{\text{repr}}) \in \mathbf{E} \wedge \\
&\quad (\text{map_of}\ \mathbf{V})\ s_{\text{repr}} = \text{Some}\ s_{\text{range}} \wedge (\text{src}\ \mathbf{p}) \in s_{\text{range}} \wedge \\
&\quad (\text{map_of}\ \mathbf{V})\ d_{\text{repr}} = \text{Some}\ d_{\text{range}} \wedge (\text{dst}\ \mathbf{p}) \in d_{\text{range}}
\end{aligned}$$

# Correctness Theorem!

Assumes

- $\Gamma, \gamma, p \vdash \langle \mathrm{rs},\ ? \rangle \Rightarrow ✓$ “Firewall with ruleset $\mathrm{rs}$ accepts packet $p$”
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$

Shows

$$\exists \mathrm{s_{repr}}\ \mathrm{d_{repr}}\ \mathrm{s_{range}}\ \mathrm{d_{range}}.\ (\mathrm{s_{repr}}, \mathrm{d_{repr}}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{s_{repr}} = \mathrm{Some}\ \mathrm{s_{range}} \wedge (\mathrm{src}\ \mathrm{p}) \in \mathrm{s_{range}} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{d_{repr}} = \mathrm{Some}\ \mathrm{d_{range}} \wedge (\mathrm{dst}\ \mathrm{p}) \in \mathrm{d_{range}}$$

# Correctness Theorem!

Assumes

- $\Gamma, \gamma, p \vdash \langle \mathrm{rs}, ? \rangle \Rightarrow \checkmark$ “Firewall with ruleset $\mathrm{rs}$ accepts packet $p$”
  - $\gamma$: arbitrary function
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$

Shows

$$\begin{aligned}
&\exists s_{repr}\ d_{repr}\ s_{range}\ d_{range}.\ (s_{repr}, d_{repr}) \in \mathrm{E} \wedge \\
&\quad (\mathrm{map_of}\ \mathrm{V})\ s_{repr} = \mathrm{Some}\ s_{range} \wedge (\mathrm{src}\ \mathrm{p}) \in s_{range} \wedge \\
&\quad (\mathrm{map_of}\ \mathrm{V})\ d_{repr} = \mathrm{Some}\ d_{range} \wedge (\mathrm{dst}\ \mathrm{p}) \in d_{range}
\end{aligned}$$

# Correctness Theorem!

Assumes

- $\Gamma, \gamma, p \vdash \langle \mathrm{rs},\ ? \rangle \Rightarrow \checkmark$ “Firewall with ruleset $\mathrm{rs}$ accepts packet $p$”
  - $\gamma$: arbitrary function For *all* iptables matching features
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$

Shows

$$\exists s_{repr}\ d_{repr}\ s_{range}\ d_{range}.\ (s_{repr}, d_{repr}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ s_{repr} = \mathrm{Some}\ s_{range} \wedge (\mathrm{src}\ \mathrm{p}) \in s_{range} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ d_{repr} = \mathrm{Some}\ d_{range} \wedge (\mathrm{dst}\ \mathrm{p}) \in d_{range}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \text{compute\_graph}\ (\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)\ (\text{simplify}\ \mathrm{rs})$

Shows

$$\exists s_{\text{repr}}\ d_{\text{repr}}\ s_{\text{range}}\ d_{\text{range}}.\ (s_{\text{repr}}, d_{\text{repr}}) \in \mathrm{E} \wedge$$
$$(\text{map\_of}\ \mathrm{V})\ s_{\text{repr}} = \text{Some}\ s_{\text{range}} \wedge (\text{src}\ \mathrm{p}) \in s_{\text{range}} \wedge$$
$$(\text{map\_of}\ \mathrm{V})\ d_{\text{repr}} = \text{Some}\ d_{\text{range}} \wedge (\text{dst}\ \mathrm{p}) \in d_{\text{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) =$ compute_graph $(\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)\ (\text{simplify}\ \mathrm{rs})$

Shows

$$\begin{aligned}
&\exists s_{\text{repr}}\ d_{\text{repr}}\ s_{\text{range}}\ d_{\text{range}}.\ (s_{\text{repr}}, d_{\text{repr}}) \in \mathrm{E} \wedge \\
&\quad (\text{map_of}\ \mathrm{V})\ s_{\text{repr}} = \text{Some}\ s_{\text{range}} \wedge (\text{src}\ \mathrm{p}) \in s_{\text{range}} \wedge \\
&\quad (\text{map_of}\ \mathrm{V})\ d_{\text{repr}} = \text{Some}\ d_{\text{range}} \wedge (\text{dst}\ \mathrm{p}) \in d_{\text{range}}
\end{aligned}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) =$ compute_graph $(\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)$ (simplify $\mathrm{rs}$)

Shows

$$\exists \mathrm{s_{repr}}\ \mathrm{d_{repr}}\ \mathrm{s_{range}}\ \mathrm{d_{range}}.\ (\mathrm{s_{repr}}, \mathrm{d_{repr}}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{s_{repr}} = \mathrm{Some}\ \mathrm{s_{range}} \wedge (\mathrm{src}\ \mathrm{p}) \in \mathrm{s_{range}} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{d_{repr}} = \mathrm{Some}\ \mathrm{d_{range}} \wedge (\mathrm{dst}\ \mathrm{p}) \in \mathrm{d_{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$

Shows

$$\exists \mathrm{s_{repr}}\ \mathrm{d_{repr}}\ \mathrm{s_{range}}\ \mathrm{d_{range}}.\ (\mathrm{s_{repr}}, \mathrm{d_{repr}}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{s_{repr}} = \mathrm{Some}\ \mathrm{s_{range}} \wedge (\mathrm{src}\ \mathrm{p}) \in \mathrm{s_{range}} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{d_{repr}} = \mathrm{Some}\ \mathrm{d_{range}} \wedge (\mathrm{dst}\ \mathrm{p}) \in \mathrm{d_{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) =$ compute_graph (prot $p$, sport $p$, dport $p$) (simplify $\mathrm{rs}$)
  - (prot $p$, sport $p$, dport $p$): Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

$$\exists \mathrm{s_{repr}}\ \mathrm{d_{repr}}\ \mathrm{s_{range}}\ \mathrm{d_{range}}.\ (\mathrm{s_{repr}}, \mathrm{d_{repr}}) \in \mathrm{E} \wedge$$
$$(\text{map\_of}\ \mathrm{V})\ \mathrm{s_{repr}} = \text{Some}\ \mathrm{s_{range}} \wedge (\text{src}\ \mathrm{p}) \in \mathrm{s_{range}} \wedge$$
$$(\text{map\_of}\ \mathrm{V})\ \mathrm{d_{repr}} = \text{Some}\ \mathrm{d_{range}} \wedge (\text{dst}\ \mathrm{p}) \in \mathrm{d_{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$
  - $(\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)$: Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

$$\exists \mathrm{s_{repr}}\ \mathrm{d_{repr}}\ \mathrm{s_{range}}\ \mathrm{d_{range}}.\ (\mathrm{s_{repr}}, \mathrm{d_{repr}}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{s_{repr}} = \mathrm{Some}\ \mathrm{s_{range}} \wedge (\mathrm{src}\ \mathrm{p}) \in \mathrm{s_{range}} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{d_{repr}} = \mathrm{Some}\ \mathrm{d_{range}} \wedge (\mathrm{dst}\ \mathrm{p}) \in \mathrm{d_{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with r
  - For *all* iptab
- Let $(V, E) =$ co
  - (prot $p$, spo

Shows

```
\begin{tikzpicture}
 \node (a) at (-4,-4) {$131.159.21.0/24$};
 \node (b) at (4,-4) {$131.159.15.240/28$};
 \node (c) at (0,-6) {$127.0.0.0/8$};
...

 \draw (a) to (a);
 \draw (a) to (b);
 \draw (a) to (c);
...
\end{tikzpicture}
```

$$\exists s_{repr}\ d_{repr}\ s_{range}\ d_{range}.\ (s_{repr}, d_{repr}) \in E \wedge$$
$$(\text{map\_of}\ V)\ s_{repr} = \text{Some}\ s_{range} \wedge (\text{src}\ p) \in s_{range} \wedge$$
$$(\text{map\_of}\ V)\ d_{repr} = \text{Some}\ d_{range} \wedge (\text{dst}\ p) \in d_{range}$$

# Correctness Theorem!

Assumes

- “Firewall with r
  - For *all* iptab
- Let $(V, E) =$ co
  - (prot $p$, spo

Shows

```
\begin{tikzpicture}
 \node (a) at (-4,-4) {$131.159.21.0/24$};
 \node (b) at (4,-4) {$131.159.15.240/28$};
 \node (c) at (0,-6) {$127.0.0.0/8$};
...

 \draw (a) to (a);
 \draw (a) to (b);
 \draw (a) to (c);
...
\end{tikzpicture}
```

$$\exists s_{repr}\ d_{repr}\ s_{range}\ d_{range}.\ (s_{repr}, d_{repr}) \in E \wedge$$
$$(\text{map\_of}\ V)\ s_{repr} = \text{Some}\ s_{range} \wedge (\text{src}\ p) \in s_{range} \wedge$$
$$(\text{map\_of}\ V)\ d_{repr} = \text{Some}\ d_{range} \wedge (\text{dst}\ p) \in d_{range}$$

# Correctness Theorem!

Assumes

- "Firewall with r
  - For *all* iptab
- Let $(V, E) =$ co
  - (prot $p$, spo

```
\begin{tikzpicture}
 \node (a) at (-4,-4) {$131.159.21.0/24$};
 \node (b) at (4,-4) {$131.159.15.240/28$};
 \node (c) at (0,-6) {$127.0.0.0/8$};
...

 \draw (a) to (a);
 \draw (a) to (b);
 \draw (a) to (c);
...
\end{tikzpicture}
```

Shows

$$\exists s_{repr}\ d_{repr}\ s_{range}\ d_{range}.\ (s_{repr}, d_{repr}) \in E \wedge$$
$$(\text{map\_of}\ V)\ s_{repr} = \text{Some}\ s_{range} \wedge (\text{src}\ p) \in s_{range} \wedge$$
$$(\text{map\_of}\ V)\ d_{repr} = \text{Some}\ d_{range} \wedge (\text{dst}\ p) \in d_{range}$$

# Correctness Theorem!

Assumes

- “Firewall with r
  - For *all* iptab
- Let $(V, E) = co$
  - $(\text{prot}\ p, \text{spo}$

```
\begin{tikzpicture}
 \node (a) at (-4,-4) {$131.159.21.0/24$};
 \node (b) at (4,-4) {$131.159.15.240/28$};
 \node (c) at (0,-6) {$127.0.0.0/8$};
...

 \draw (a) to (a);
 \draw (a) to (b);
 \draw (a) to (c);
...
\end{tikzpicture}
```

Shows

$$\exists s_{repr}\ d_{repr}\ s_{range}\ d_{range}.\ (s_{repr}, d_{repr}) \in E \wedge$$
$$(\text{map\_of}\ V)\ s_{repr} = \text{Some}\ s_{range} \wedge (\text{src}\ p) \in s_{range} \wedge$$
$$(\text{map\_of}\ V)\ d_{repr} = \text{Some}\ d_{range} \wedge (\text{dst}\ p) \in d_{range}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$
  - $(\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)$: Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

$$\exists \mathrm{s_{repr}}\ \mathrm{d_{repr}}\ \mathrm{s_{range}}\ \mathrm{d_{range}}.\ (\mathrm{s_{repr}}, \mathrm{d_{repr}}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{s_{repr}} = \mathrm{Some}\ \mathrm{s_{range}} \wedge (\mathrm{src}\ \mathrm{p}) \in \mathrm{s_{range}} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ \mathrm{d_{repr}} = \mathrm{Some}\ \mathrm{d_{range}} \wedge (\mathrm{dst}\ \mathrm{p}) \in \mathrm{d_{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$
  - $(\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)$: Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

$$\exists s_{\mathrm{repr}}\ d_{\mathrm{repr}}\ s_{\mathrm{range}}\ d_{\mathrm{range}}.\ (s_{\mathrm{repr}}, d_{\mathrm{repr}}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ s_{\mathrm{repr}} = \mathrm{Some}\ s_{\mathrm{range}} \wedge (\mathrm{src}\ \mathrm{p}) \in s_{\mathrm{range}} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ d_{\mathrm{repr}} = \mathrm{Some}\ d_{\mathrm{range}} \wedge (\mathrm{dst}\ \mathrm{p}) \in d_{\mathrm{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \mathrm{compute\_graph}\ (\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)\ (\mathrm{simplify}\ \mathrm{rs})$
  - $(\mathrm{prot}\ p, \mathrm{sport}\ p, \mathrm{dport}\ p)$: Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

$$\exists s_{\mathrm{repr}}\ d_{\mathrm{repr}}\ s_{\mathrm{range}}\ d_{\mathrm{range}}.\ (s_{\mathrm{repr}}, d_{\mathrm{repr}}) \in \mathrm{E} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ s_{\mathrm{repr}} = \mathrm{Some}\ s_{\mathrm{range}} \wedge (\mathrm{src}\ \mathrm{p}) \in s_{\mathrm{range}} \wedge$$
$$(\mathrm{map\_of}\ \mathrm{V})\ d_{\mathrm{repr}} = \mathrm{Some}\ d_{\mathrm{range}} \wedge (\mathrm{dst}\ \mathrm{p}) \in d_{\mathrm{range}}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \text{compute\_graph}\ (\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)\ (\text{simplify}\ \mathrm{rs})$
  - $(\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)$: Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

$$\exists s_{repr}\ d_{repr}\ s_{range}\ d_{range}.\ (s_{repr}, d_{repr}) \in \mathrm{E} \wedge$$
$$(\text{map\_of}\ \mathrm{V})\ s_{repr} = \text{Some}\ s_{range} \wedge (\text{src}\ \mathrm{p}) \in s_{range} \wedge$$
$$(\text{map\_of}\ \mathrm{V})\ d_{repr} = \text{Some}\ d_{range} \wedge (\text{dst}\ \mathrm{p}) \in d_{range}$$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \text{compute\_graph}\ (\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)\ (\text{simplify}\ \mathrm{rs})$
  - $(\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)$: Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

In the graph, we can find an arrow from $\text{src}\ \mathrm{p} \longrightarrow \text{dst}\ \mathrm{p}$

# Correctness Theorem!

Assumes

- "Firewall with ruleset $\mathrm{rs}$ accepts packet $p$"
  - For *all* iptables matching features
- Let $(V, E) = \text{compute\_graph}\ (\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)\ (\text{simplify}\ \mathrm{rs})$
  - $(\text{prot}\ p, \text{sport}\ p, \text{dport}\ p)$: Example $(\mathrm{tcp},\ 42242,\ 80)$

Shows

In the graph, we can find an arrow from $\text{src}\ \mathrm{p} \longrightarrow \text{dst}\ \mathrm{p}$

Warning: The other direction may not hold! Connectivity not guaranteed.

# Stats

# Tool Implementation

- Functions compute_graph and simplify

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated
- How long would it take you to review 5k of machine-generated Haskell?

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated
- How long would it take you to review 5k of unreadable Haskell?

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated
- How long would it take you to review 5k of unreadable Haskell?
  - Answer: 0s

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated
- How long would it take you to review 5k of unreadable Haskell?
  - Answer: 0s
  - Let your computer do this!

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated
- How long would it take you to review 5k of unreadable Haskell?
  - Answer: 0s
  - Let your computer do this!
  - . . . once you reviewed the correctness theorem.

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated
- How long would it take you to review 5k of unreadable Haskell?
  - Answer: 0s
  - Let your computer do this!
  - . . . once you reviewed the correctness theorem.
  - . . . and a formal (= machine-verifiable) proof is shipped with the code.

# Tool Implementation

- Functions compute_graph and simplify
- Around 5k LoC Haskell
  - Machine-Generated
- How long would it take you to review 5k of unreadable Haskell?
  - Answer: 0s
  - Let your computer do this!
  - . . . once you reviewed the correctness theorem.
  - . . . and a formal (= machine-verifiable) proof is shipped with the code.
- Suddenly:
  - Correctness of code taken for granted.
  - Discuss and review the assumptions of the code.

# Stats (approx.)

- 5k LoC
- 5 Lines of Theorem

# Stats (approx.)

- 5k LoC
- 5 Lines of Theorem
  - “Just having to review 5 Lines of Theorem to get confidence in 5k LoC. Awesome!”

# Stats (approx.)

- 5k LoC
- 5 Lines of Theorem
  - “Just having to review 5 Lines of Theorem to get confidence in 5k LoC. Awesome!”
- So what’s the catch?

# Stats (approx.)

- 5k LoC
- 5 Lines of Theorem
  - “Just having to review 5 Lines of Theorem to get confidence in 5k LoC. Awesome!”
- So what’s the catch?
  - 50k Lines of Formalization
  - Over 3 years.

# More Stats

- Another Example: seL4 (verified microkernel)

# More Stats

- Another Example: seL4 (verified microkernel)
- by June Andronick, Callum Bannister, Joel Beeren, Nelson Billing, Bernard Blackham, Timothy Bourke, Andrew Boyton, Matthew Brassil, Aleksander Budzynowski, Manuel Chakravarty, Xi Ma Chen, Nahida Chowdhury, Peter Chubb, David Cock, Adrian Danis, Matthias Daum, Jeremy Dawson, Philip Derrin, Dhammika Elkaduwe, Kevin Elphinstone, Kai Engelhardt, Matthew Fernandez, Peter Gammie, Xin Gao, Dean Garden, Gianpaolo Gioiosa, David Greenaway, Matthew Grosvenor, Lukas Haenel, Gernot Heiser, Rohan Jacob-Rao, Benjamin Kalman, Justin King-Lacroix, Gerwin Klein, Rafal Kolanski, Alexander Kroh, Etienne Le Sueur, Corey Lewis, Japheth Lim, Anna Lyons, Tran Ma, Daniel Matichuk, Stephanie McArthur, Sam McNally, Jia Meng, Catherin Menon, Toby Murray, Magnus Myreen, Michael Norrish, Liam O'Connor, Ameya Palande, Sean Peters, Simon Rodgers, Sean Seefried, Thomas Sewell, Rupert Shuttleworth, Vernon Tang, Michael von Tessin, David Tsai, Harvey Tuch, Adam Walker, James Wilmot, Simon Winwood, and Jiawei Xie

# More Stats

- Another Example: seL4 (verified microkernel)
- by June Andronick, Callum Bannister, Joel Beeren, Nelson Billing, Bernard Blackham, Timothy Bourke, Andrew Boyton, Matthew Brassil, Aleksander Budzynowski, Manuel Chakravarty, Xi Ma Chen, Nahida Chowdhury, Peter Chubb, David Cock, Adrian Danis, Matthias Daum, Jeremy Dawson, Philip Derrin, Dhammika Elkaduwe, Kevin Elphinstone, Kai Engelhardt, Matthew Fernandez, Peter Gammie, Xin Gao, Dean Garden, Gianpaolo Gioiosa, David Greenaway, Matthew Grosvenor, Lukas Haenel, Gernot Heiser, Rohan Jacob-Rao, Benjamin Kalman, Justin King-Lacroix, Gerwin Klein, Rafal Kolanski, Alexander Kroh, Etienne Le Sueur, Corey Lewis, Japheth Lim, Anna Lyons, Tran Ma, Daniel Matichuk, Stephanie McArthur, Sam McNally, Jia Meng, Catherin Menon, Toby Murray, Magnus Myreen, Michael Norrish, Liam O'Connor, Ameya Palande, Sean Peters, Simon Rodgers, Sean Seefried, Thomas Sewell, Rupert Shuttleworth, Vernon Tang, Michael von Tessin, David Tsai, Harvey Tuch, Adam Walker, James Wilmot, Simon Winwood, and Jiawei Xie
  - 8k Lines of C Code

# More Stats

- Another Example: seL4 (verified microkernel)
- by June Andronick, Callum Bannister, Joel Beeren, Nelson Billing, Bernard Blackham, Timothy Bourke, Andrew Boyton, Matthew Brassil, Aleksander Budzynowski, Manuel Chakravarty, Xi Ma Chen, Nahida Chowdhury, Peter Chubb, David Cock, Adrian Danis, Matthias Daum, Jeremy Dawson, Philip Derrin, Dhammika Elkaduwe, Kevin Elphinstone, Kai Engelhardt, Matthew Fernandez, Peter Gammie, Xin Gao, Dean Garden, Gianpaolo Gioiosa, David Greenaway, Matthew Grosvenor, Lukas Haenel, Gernot Heiser, Rohan Jacob-Rao, Benjamin Kalman, Justin King-Lacroix, Gerwin Klein, Rafal Kolanski, Alexander Kroh, Etienne Le Sueur, Corey Lewis, Japheth Lim, Anna Lyons, Tran Ma, Daniel Matichuk, Stephanie McArthur, Sam McNally, Jia Meng, Catherin Menon, Toby Murray, Magnus Myreen, Michael Norrish, Liam O'Connor, Ameya Palande, Sean Peters, Simon Rodgers, Sean Seefried, Thomas Sewell, Rupert Shuttleworth, Vernon Tang, Michael von Tessin, David Tsai, Harvey Tuch, Adam Walker, James Wilmot, Simon Winwood, and Jiawei Xie
  - 8k Lines of C Code
  - 25 person-years

# More Stats

- Another Example: seL4 (verified microkernel)
- by June Andronick, Callum Bannister, Joel Beeren, Nelson Billing, Bernard Blackham, Timothy Bourke, Andrew Boyton, Matthew Brassil, Aleksander Budzynowski, Manuel Chakravarty, Xi Ma Chen, Nahida Chowdhury, Peter Chubb, David Cock, Adrian Danis, Matthias Daum, Jeremy Dawson, Philip Derrin, Dhammika Elkaduwe, Kevin Elphinstone, Kai Engelhardt, Matthew Fernandez, Peter Gammie, Xin Gao, Dean Garden, Gianpaolo Gioiosa, David Greenaway, Matthew Grosvenor, Lukas Haenel, Gernot Heiser, Rohan Jacob-Rao, Benjamin Kalman, Justin King-Lacroix, Gerwin Klein, Rafal Kolanski, Alexander Kroh, Etienne Le Sueur, Corey Lewis, Japheth Lim, Anna Lyons, Tran Ma, Daniel Matichuk, Stephanie McArthur, Sam McNally, Jia Meng, Catherin Menon, Toby Murray, Magnus Myreen, Michael Norrish, Liam O'Connor, Ameya Palande, Sean Peters, Simon Rodgers, Sean Seefried, Thomas Sewell, Rupert Shuttleworth, Vernon Tang, Michael von Tessin, David Tsai, Harvey Tuch, Adam Walker, James Wilmot, Simon Winwood, and Jiawei Xie
  - 8k Lines of C Code
  - 25 person-years
  - 10 person-months to add integrity and authority confinement.
  - 51 person-months to add IFS.

# Takeaway

# Takeaway

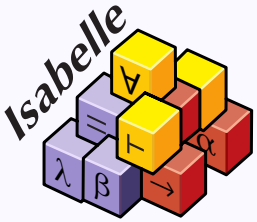
- Tool to verify iptables
  - `https://github.com/diekmann/Iptables_Semantics`

# Takeaway

- Tool to verify iptables
  - `https://github.com/diekmann/Iptables_Semantics`
- Verifying the verifier was great fun
  - `Yo_Dawg.jpg`

# Give Isabelle/HOL a Try!

- Write correct code
- Only correct code can be secure
- Reason about the correctness of your code




https://isabelle.in.tum.de/

# TL;DR

# Talk Summary

It took me over 3 years to code 5k LoC. Not very impressive. However, that code is is correct! According to an independent self-conducted study (pun intended), it is also the best open-source iptables analyzer out there. The value is not the 5k LoC, but rather 5 Lines of Theorem which show that the 5k LoC are correct.

In this talk, I don't want to walk you over the 5k lines of machine-generated Haskell code. Nor do I want to show you the 50k lines of formalization needed. I just want to show you the 5 Lines of the Theorem. Is this enough to convince you about the correctness of the iptables analyzer? Neighbors, feel free to question anybody who does formalism for the sake of pretending to be a scientist. Formalism has a clear value: It is more expressive than our programming languages. And it is a nice language to precisely state the high-level requirements of our software. And here lies the true value: Computers understand rigorous formalism, too. Computers can check proofs, so you don't have to. Just inspect 5 Lines of Theorem to get confidence in the correctness of 5k LoC. This approach seems to scale.*

*) given an infinite supply of PhD students who do the proofs for you.