VeriCon: Towards Verifying Controller Programs in SDNs

Thomas Ball, Nikolaj Bjorner, **Aaron Gember**, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky

Microsoft Research
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON
TEL AVIV UNIVERSITY
The Hebrew University of Jerusalem
TECHNISCHE UNIVERSITÄT MÜNCHEN

Guaranteeing network invariants

- Network should always satisfy some invariants

- Difficult to write an SDN application that always guarantees such invariants

Limitations of existing approaches

1. Establish existence, but not absence, of bugs
   – **NICE** *(finite-state model checking*): unexplored topologies may cause bugs to be missed
   – **HSA** (*check network snapshots*): snapshots may not capture situations in which bugs exist
2. Runtime overhead
   – **VeriFlow** & **NetPlumber** (*check in real-time*): bugs only identified when app is actually running

VeriCon

Verifies network-wide invariants for ***any*** event sequence and ***all*** admissible topologies

SDN application in *Core SDN*

+

Topology constraints & invariants in first order logic

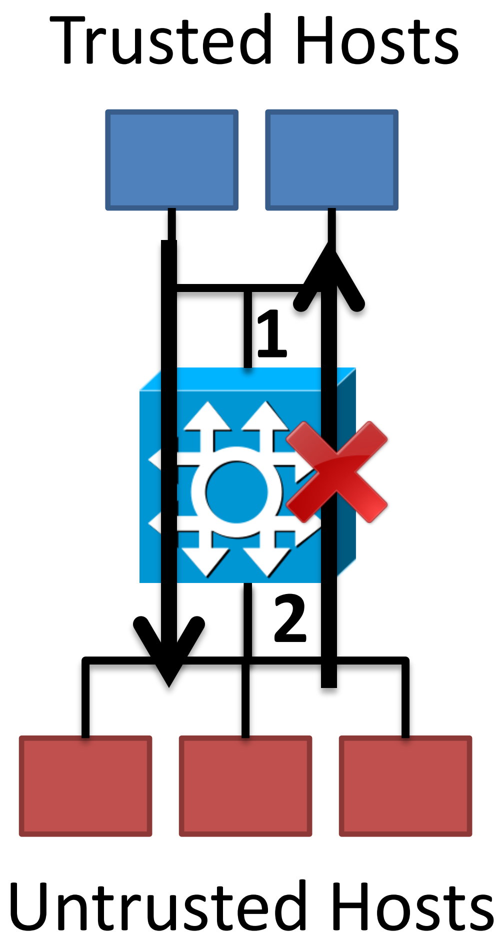
Verify conditions using the Z3 theorem prover

Guarantee invariants are satisfied

OR

Concrete counter-example

Example: stateful firewall

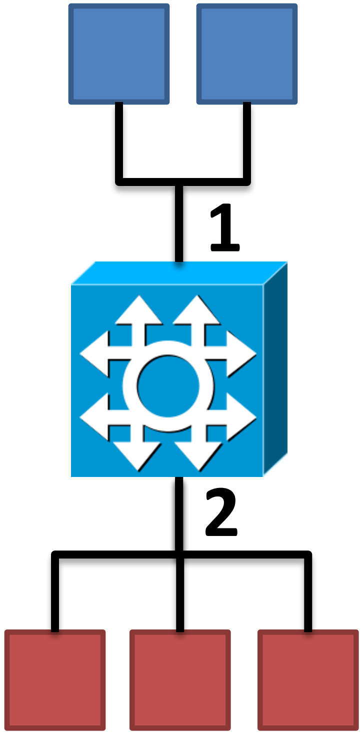


- Always forward from trusted to untrusted hosts
- Only forward from untrusted to trusted hosts if a trusted host previously sent a packet to the untrusted host

Core SDN (CSDN) language

- Define and initialize relations
  - Topology: $link\ (S, O, H)$ $link(S_1, I_1, I_2, S_2)$
  - Forwarding: $S.ft(Src \rightarrow Dst, I \rightarrow O)$
    $S.sent(Src \rightarrow Dst, I \rightarrow O)$
- Write event handlers: $\mathbf{pktIn}(S, Pkt, I)$
  - Update relation
  - Install rule (insert into *ft*)
  - Forward packet (insert into *sent*)
  - If-then-else

Stateful firewall in CSDN



$\mathbf{rel}\ \mathrm{tr}(SW, HO) = \{\}$

$\mathbf{pktIn}(s, pkt, prt(1)) \rightarrow$

$\quad s.\mathbf{forward}(pkt, prt(1), prt(2))$

$\quad \mathrm{tr}.insert(s, pkt.dst)$

$\quad s.\mathbf{install}(pkt.src \rightarrow pkt.dst, prt(1), prt(2))$

$\mathbf{pktIn}(s, pkt, prt(2)) \rightarrow$

$\quad \mathbf{if}\ \mathrm{tr}(s, pkt.src)\ \mathbf{then}$

$\quad\quad s.\mathbf{forward}(pkt, prt(2), prt(1))$

$\quad\quad s.\mathbf{install}(pkt.src \rightarrow pkt.dst, prt(2), prt(1))$

Invariants

- **Topology**: define admissible topologies

assumed to hold initially

- **Safety**: define the required consistency of network-wide states
- **Transition**: define the effect of executing event handlers

checked initially & after each event

Stateful firewall invariants

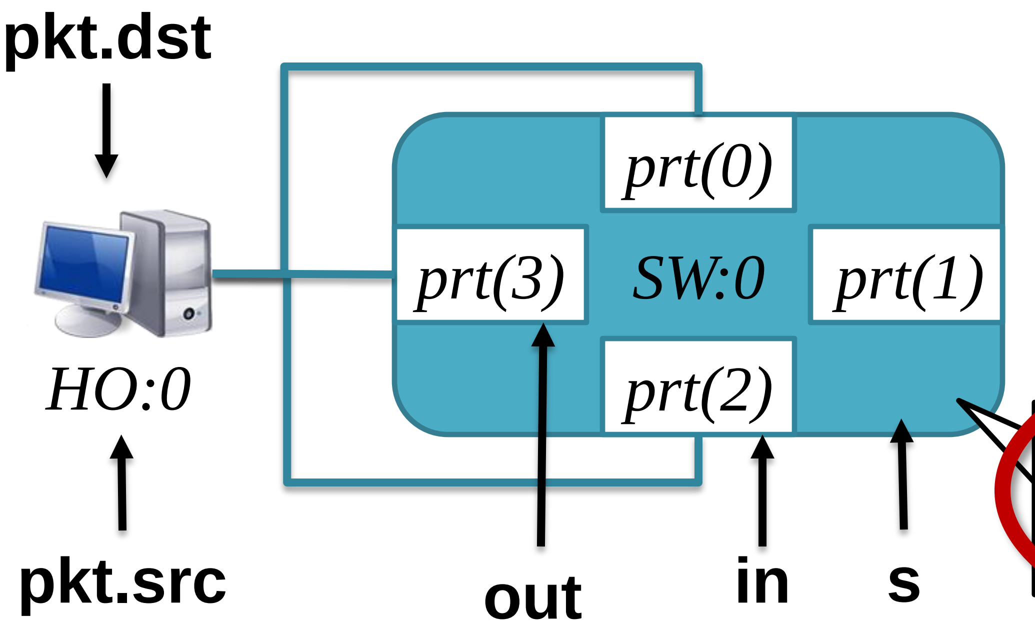
- **Topology**: At least one switch with two ports, $prt(1)$ & $prt(2)$; a packet P is forwarded from an untrusted host U to a trusted host T

$$\begin{aligned} &\exists U, T : HO, S : SW, P : PK. \\ &\quad link(S, prt(2), U) \wedge link(S, prt(1), T) \wedge \\ &\quad P.src = U \wedge P.dst = T \wedge S.sent(P, prt(2), prt(1)) \end{aligned}$$

- **Safety**: For every packet sent from a host U to a host T there exists a packet sent to T' from U

$$I_1 = \begin{array}{c} S.sent(P, prt(2), prt(1)) \Rightarrow \\ \exists P' : PK . P'.dst = P.src \wedge S.sent(P', prt(1), prt(2)) \end{array}$$

Counterexample



flow-table

Src	Dst	In	Out
HO:0	*HO:0*	*	*

I_1 is not inductive—not all executions starting from an arbitrary state satisfy the invariant

Additional firewall invariants

- Flow table entries only contain forwarding rules from trusted hosts

$$I_2 = \begin{array}{c} S.ft(Src \rightarrow Dst, prt(2), prt(1)) \Rightarrow \\ \exists P': PK.P'.dst = Src \wedge S.sent(P', prt(1), prt(2)) \end{array}$$

- Controller relation tr records the correct hosts

$$I_3 = \begin{array}{c} tr(S, H) \Rightarrow \\ \exists P: PK.P.dst = H \wedge S.sent(P, prt(1), prt(2)) \end{array}$$

- $I_1 \wedge I_2 \wedge I_3$ is inductive

Non-buggy verification examples

Program
Firewall
Stateless Firewall
Firewall + Host Migration
Learning Switch
Learning Switch + Auth
Resonance (simplified)
Stratos (simplified)

Buggy verification examples

Benchmark	Counterex Host + Sw
Auth: Rules for unauth host not removed	3 + 2
Firewall: Forgot part of consistency inv	5 + 3
Firewall: No check if host is trusted	6 + 4
Firewall: No inv defining trusted host	6 + 4
Learning: Packets not forwarded	1 + 1
Resonance: No inv for host to have one state	11 + 4
StatelessFW: Rule allowing all port 2 traffic	4 + 2

Future work

- Assume events are executed atomically
  - Enforceable using barriers, with performance hit
  - Consider out-of-order rule installs
- Rule timeouts
  - App handles timeout events to update its *ft* relation and check invariants
  - Need to reason about event ordering

Summary of VeriCon

- Verifies network-wide invariants for ***any*** event sequence and ***all*** admissible topologies
- Guarantees invariants are satisfied, or provides a concrete counterexample
- Application with 93 LOC and 13 invariants is verified in 0.21s

http://agember.com/go/vericon