Model Checking Dynamic Datapaths

Aurojit Panda, Katerina Argyraki, Scott Shenker
UC Berkeley, ICSI, EPFL
Networks: Not Just for Delivery

- Enforce a variety of invariants:
  - Packet Isolation: Packets from A can not reach B
  - Content Isolation: Content X never accessible by A.
  - Rate Limiting: B limited to M requests per second.
Invariants are Global

- Existing work on verifying global invariants
  - Anteater, HSA, VeriFlow.

- Key assumption
  - Forwarding state dictated by control plane.
Many Datapaths are "Dynamic"

- **Dynamic**: Forwarding State affected by traffic.

- Examples
  - Middle boxes
  - Learning switches
  - Loose Source Record Route IP option.
Dynamic Behavior can Violate Invariants

10.0.0.1 > 10.0.1.1

10.0.0.1

Firewall

deny: 10.0.0.1-> 10.0.1.1

10.0.0.2

10.0.1.1

10.0.1.2
Dynamic Behavior can Violate Invariants

deny: 10.0.0.1 -> 10.0.1.1
Dynamic Behavior can Violate Invariants
Dynamic Behavior can Violate Invariants

10.0.0.1 > 10.0.1.1

10.0.0.1

Proxy
10.1.0.1

10.0.0.2

Firewall
deny: 10.0.0.1-> 10.0.1.1

10.0.1.1

10.0.1.2
Dynamic Behavior can Violate Invariants

10.0.0.1

10.1.0.1 > 10.0.1.1

Proxy
10.1.0.1

Firewall

deny: 10.0.0.1-> 10.0.1.1

10.0.0.2

10.0.1.1

10.0.1.2
Another Example

10.0.0.1 > 10.0.1.1
BAD

10.0.0.1
Compression Middlebox

10.0.0.2

IDS
if BAD send to 10.0.1.2

10.0.1.1

10.0.1.2
Another Example

10.0.0.1 > 10.0.1.1
gzip(BAD)

Compression Middlebox

IDS

if BAD send to 10.0.1.2

10.0.0.1

10.0.0.2

10.0.1.1

10.0.1.2
Another Example

10.0.0.1 > 10.0.1.1
gzip(BAD)

if BAD send to 10.0.1.2
Why is this a Problem in the Real World?

- Networks are complex and enforce many invariants.
- Hard for administrators to keep global image in head.
- NFV: Easier to make changes that violate invariants.

Goal:

- Check invariants for networks with dynamic elements.
Focusing on Middleboxes for this talk.
High-Level Solution

- Treat network as a large program.
  - Middleboxes are functions in this program.
- Use model checking to check the network.
- Naive implementation intractable

Challenge:
- Network → Program so model checking is tractable.
Scaling through Modularity

- Three techniques
  - Middlebox models (what to model?)
  - Leverage service chaining.
  - Policy choices that speed up analysis.
Consider a DPI Middlebox

Real processing pipeline

Expensive to combine: Exponential growth
Consider a DPI Middlebox

Receive packet → Lookup flow state → ... → Labeled Harmful, Labeled Benign → Send packet

Many steps to analyze traffic

Important for Global Properties
The DPI Model in Math

\[ \forall send(d, e, p) \quad \implies \exists e' : recv(e', d, p) \]
\[ \land (d.label(p) = \text{harmful}) \land (d.label(p) = \text{benign}) \]
\[ \land rtime(d, p) < stime(d, p) \]

For any packet p sent by DPI box d

p was received by d
p was marked harmful or benign
also p was received before being sent
Model Globally Significant Behavior

- Checking model accuracy?
- Verify code against model.
- Enforce model.
Simple Models not Enough

Networks with 25 middleboxes take 32.2 seconds.
• Modeling middleboxes
• Leverage service chaining
• Policies for scalability
Networks of Middleboxes

- Load Balancer
- Proxy
- IDS
- WAN Opt
- Web Server
- Web Server
- Web Server
- Web Server
- Web Server
Networks of Middleboxes

Network

Load Balancer
Proxy
IDS
WAN Opt
Web Server
Web Server
Web Server
Web Server
Web Server

Also provide annotations on when paths are taken.
Networks of Middleboxes

Also provide annotations on when paths are taken.
Networks of Middleboxes

Also provide annotations on when paths are taken.
Service Chaining

- Solutions to implement such chaining.
- Limits middlebox states to be checked.
  - Middlebox state depends on past traffic.
- Chaining policy defines sources of traffic.
- Network path: set of middleboxes traversed.
• Modeling middleboxes

• Leverage service chaining

• Policies for scalability
How much of the Network to Consider?

Prove A isolated from B.

Network Path is set of Middleboxes Traversed.
How much of the Network to Consider?

Prove A isolated from B.

Network Path is set of Middleboxes Traversed.
How much of the Network to Consider?

Prove A isolated from B.

Network Path is set of Middleboxes Traversed.
Consider Only Network Path

- **Pro:** Scales with path length not size of network
- **Con:** Not generally applicable
- Allows scaling to 10000s of nodes.
- Trivial test (2 endhosts, 1 firewall, no ACLs)
  - With pruning 0.11 seconds (with 25000 mboxes)
  - Without pruning 32.2 seconds (with 25 mboxes)
When can we Prune Part of the Network?

- **Path Independence**
  - Model checking behavior: Enables pruning.
- **Robustness**: Network changes remain local
Achieving Path Independence

- Solution depends on invariant and network.

10.0.0.1 → Application Firewall → Proxy → Application Firewall → Web Server
Achieving Path Independence

- Solution depends on invariant and network.
- Add a firewall before the proxy.
Achieving Path Independence

- Solution depends on invariant and network.
- Add a firewall before the proxy.
- Change proxy to enforce access invariants.
Achieving Path Independence

- Solution depends on invariant and network.
- Add a firewall before the proxy.
- Change proxy to enforce access invariants.
- Can automatically check path independence.
Achieving Path Independence

- Solution depends on invariant and network.
- Add a firewall before the proxy.
- Change proxy to enforce access invariants.
- Can automatically check path independence.
Tools for Checking Invariants

- We have implemented a tool with these optimizations.
- Leverages Z3, a SMT solver from Microsoft.
- Implemented in about 3700 lines of Python code
  - The models themselves are less than 1500 lines.
- Models about 10 different middlebox kinds.
- Much of the space: Expressing math in Python.
Early Results from Tools

- Use Loose Source Routing to circumvent firewall
- Invariant: No packets from 10.0.0.1 to 10.0.1.1
- We can verify this in 0.39 seconds.
Early Results from Tools

- Use Loose Source Routing to circumvent firewall
- Invariant: No packets from 10.0.0.1 to 10.0.1.1
- We can verify this in 0.39 seconds.
Summary

- Path independent policies and invariants:
  - Easy to maintain, easy to check.
- Check if policies + invariants are path independent.
- Check if those invariants hold.
- Offline but quick verification possible.