Safety Verification of Stateful Networks

Sharon Shoham
Collaborators

Kalev Alpernas
Mooly Sagiv

Roman Manevich

Yaron Velner

Aurojit Panda
Scott Shenker

TEL AVIV UNIVERSITY

Amitrachaita

Bloom

THE HEBREW UNIVERSITY OF JERUSALEM

Berkeley UNIVERSITY OF CALIFORNIA
Network Safety Verification

• Show that something bad cannot happen

• Isolation:
  • A packet of type $t$ sent from host A never reaches host B
  • E.g., no packets from Simon to Bob
Stateful Networks

Middleboxes: Local functionality enhancements

- Security (firewalls, IDSs,...)
- Performance (caches, load balancers,...)
- New functionality (proxies,...)
Stateful Networks

Middleboxes: Local functionality enhancements

• Security (firewalls, IDSs,...)
• Performance (caches, load balancers,...)
• New functionality (proxies,...)
Safety with Middleboxes

- For stateless networks
  - Safety can be checked by tracing the forwarding graph

- Middleboxes make everything harder
  - Rewrite packet headers
  - Behave differently over time – need to reason about history
    - Forwarding of a packet depends on previous packets
    - E.g. cache
Challenges in Stateful Network Verification

• Source code complexity
  • Bro Network Intrusion
    • 101,500 lines of C++, Python, Perl, Awk, Lex, Yacc
  • Snort IDS 220,000 C, ...
  • Pfsense 476,438 locs of C,php,scripts,...

• Configuration errors
  • Do the topology and the middlebox configuration satisfy the safety property?
  • Major source of network failures [IMC:RJ13]

Our approach for verifying stateful networks

• Abstraction: over-approximate network behavior
  • If isolation is preserved by the over-approximation, the network is safe
  • If violation is detected, may be a false alarm
## Abstractions for stateful networks

<table>
<thead>
<tr>
<th>Abstraction Description</th>
<th>Complexity or Other Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstracting middleboxes as finite state machines (FSM)</td>
<td>No need to analyze the code</td>
</tr>
<tr>
<td>- Network safety is undecidable</td>
<td></td>
</tr>
<tr>
<td>Abstracting the order of packet arrival</td>
<td>Decidability [TACAS’16]</td>
</tr>
<tr>
<td>- EXPSPACE-complete</td>
<td></td>
</tr>
<tr>
<td>Allowing each middlebox to nondeterministically revert to its initial state</td>
<td>Polynomial complexity for isolation</td>
</tr>
<tr>
<td>- Abstract interpretation</td>
<td></td>
</tr>
</tbody>
</table>

Concrete Network $\approx$ Communicating FSMs

Network state $\in (M \to S) \times (E \to P^*)$

- mbox boxes
- mbox states
- channels
- packets $\in$ Hosts $\times$ Hosts $\times$ T $\times$ $<$src, dst, tpe$>$
Concrete Network ≈ Communicating FSMs

Network state $\in (M \rightarrow S) \times (E \rightarrow P^*)$

- mbox boxes
- mbox states
- channels
- packets $\in \text{Hosts} \times \text{Hosts} \times T$
  $<$src, dst, tpe$>$
Concrete Interpretation

Concrete domain: sets of states

\[ C = \mathbb{P}((M \rightarrow S) \times (E \rightarrow P^*)) \]
Abstract Interpretation + Finite Abstraction

Concrete domain: sets of states

Abstract domain

\[ C = \mathcal{P}((M \rightarrow S) \times (E \rightarrow P^*)) \]
Abstract Interpretation + *Finite* Abstraction

Abstract computation of reachable states

Concrete domain: sets of states

$$C = \mathbb{P}((M \to S) \times (E \to P^*))$$
Abstract Interpretation + \textit{Finite} Abstraction

Concrete domain: sets of states
\[ C = \mathbb{P}((M \to S) \times (E \to P^*)) \]
Network Abstractions

(0) Concrete domain

\[ C = \mathbb{P}((M \to S) \times (E \to P^*)) \]
Network Abstractions

(0) Concrete domain

(1) Unordered channels
   • Channels as multisets of packets

\[ C = \mathbb{P}(M \to S) \times (E \to P^*) \]
\[ \mathbb{P}(M \to S) \times (E \to P \to \mathbb{N}) \]
Network Abstractions

(0) Concrete domain

(1) Unordered channels
- Channels as multisets of packets

Safety verification is decidable [TACAS’16]
- Reduction to/from Petri Net coverability
- EXPSPACE complexity

Network Abstractions

(0) Concrete domain

(1) Unordered channels
   • Channels as multisets of packets

(2) Counter abstraction on channels
   • Channels as sets of packets

\[
C = \mathcal{P}((M \to S) \times (E \to P^*))
\]

\[
\mathcal{P}((M \to S) \times (E \to P \to \mathbb{N}))
\]

\[
\mathcal{P}((M \to S) \times (E \to \mathcal{P}(P)))
\]
Network Abstractions

(0) Concrete domain
(1) Unordered channels
  • Channels as multisets of packets
(2) Counter abstraction on channels
  • Channels as sets of packets
(3) Cartesian Abstraction
  • No correlations between mboxes, channels, packets

\[ C = \mathbb{P}((M \to S) \times (E \to \mathbb{P}(P^*))) \]
\[ \mathbb{P}((M \to S) \times (E \to P \to \mathbb{N})) \]
\[ \mathbb{P}((M \to S) \times (E \to \mathbb{P}(P))) \]
\[ A = (M \to \mathbb{P}(S)) \times (E \to \mathbb{P}(P)) \]
Network Abstractions

(0) Concrete domain

(1) Unordered channels
  • Channels as multisets of packets

(2) Counter abstraction on channels
  • Channels as sets of packets

(3) Cartesian Abstraction
  • No correlations between mboxes, channels, packets

\[ C = \mathbb{P}((M \rightarrow S) \times (E \rightarrow P^*)) \]

\[ \mathbb{P}((M \rightarrow S) \times (E \rightarrow P \rightarrow \mathbb{N})) \]

\[ \mathbb{P}((M \rightarrow S) \times (E \rightarrow \mathbb{P}(P))) \]

\[ \mathcal{A} = (M \rightarrow \mathbb{P}(S)) \times (E \rightarrow \mathbb{P}(P)) \]

Recall: \( \text{height}(\mathbb{P}(X)) = |X| \)

Time(LFP\#) = poly(|M|, |S|, |E|, |P|)
Network Abstractions

(0) Concrete domain

(1) Unordered channels
    • Channels as multisets of packets

(2) Counter abstraction on channels
    • Channels as sets of packets

(3) Cartesian Abstraction
    • No correlations between mboxes, channels, packets

\[ C = \mathbb{P}( (M \to S) \times (E \to P^*) ) \]

\[ \mathbb{P}( (M \to S) \times (E \to P \to \mathbb{N}) ) \]

\[ \mathbb{P}( (M \to S) \times (E \to \mathbb{P}(P)) ) \]

\[ \mathcal{A} = (M \to \mathbb{P}(S)) \times (E \to \mathbb{P}(P)) \]

\[ \text{Time}(\text{LFP}#) = \text{poly}(|M|, |S|, |E|, |P|) \]

Unfortunately \[ |S| = \exp(|\text{Hosts}|) \]
Example: Hole-Punching Firewall

allow packets from $h_{ext}$ to internal network only if internal host already sent packets to $h_{ext}$
Example: Firewall

AMDL: Abstract MBox Def. Lang.
  • Similar to [SIGCOMM’16]
  • States ≈ n-ary relations
  • Topology agnostic
  • Encode FSM compactly
    • For fixed topology finite state

hole_punching_firewall =
port_in ? <src,dst,tpe> =>
  trusted(dst) := true;
  port_ext ! <src,dst,tpe>
|  port_ext ? <src,dst,tpe> =>
    src in trusted =>
      port_in ! <src,dst,tpe>

Example: Firewall

AMDL: Abstract MBox Def. Lang.
  • Similar to [SIGCOMM’16]

• States \(\approx\) n-ary relations
• Topology agnostic
• Encode FSM compactly
  • For fixed topology finite state

hole_punching_firewall =

port_in ? \langle src, dst, tpe \rangle =>

trusted(dst) := true;
port_ext ! \langle src, dst, tpe \rangle |

port_ext ? \langle src, dst, tpe \rangle =>

src in trusted =>

port_in ! \langle src, dst, tpe \rangle

Example: Firewall

AMDL: Abstract MBox Def. Lang.
  • Similar to [SIGCOMM’16]

• States ≈ n-ary relations
• Topology agnostic
• Encode FSM compactly
  • For fixed topology finite state

hole_punching_firewall =

port_in ? <src,dst,tpe> =>
  trusted(dst) := true;
  port_ext ! <src,dst,tpe>

port_ext ? <src,dst,tpe> =>
  src in trusted =>
  port_in ! <src,dst,tpe>

Example: Firewall

AMDL: Abstract MBox Def. Lang.
  • Similar to [SIGCOMM’16]
  
  • States ≈ n-ary relations
  • Topology agnostic
  • Encode FSM compactly
    • For fixed topology finite state

Example: Firewall

Amdl: Abstract

• Similar to [SIGCOMM'16]
• States \( \sim n \)-ary relations
• Topology agnostic
• Encode FSM compactly
  • For fixed topology finite state

\[ \text{trusted: } \mathcal{P}(\text{Hosts}) \]
\[ |S| = 2^{|\text{Hosts}|} \]

\[
\text{hole_punching_firewall} = \\
\text{port_in} ? \langle \text{src}, \text{dst}, \text{tpe} \rangle \Rightarrow \\
\text{trusted}(\text{dst}) := \text{true}; \\
\text{port_ext} ! \langle \text{src}, \text{dst}, \text{tpe} \rangle
\]

\[
\text{port_ext} ? \langle \text{src}, \text{dst}, \text{tpe} \rangle \Rightarrow \\
\text{src in trusted} \Rightarrow \\
\text{port_in} ! \langle \text{src}, \text{dst}, \text{tpe} \rangle
\]

Middlebox-level Abstraction

\[ \text{Time(LFP#)} = \text{poly}(|M|, |S|, |E|, |P|) \]

- Problem: Middlebox state space exponential in number of hosts
Middlebox-level Abstraction

\[
\text{Time}(\text{LFP}^\#) = \text{poly}(|M|, |S|, |E|, |P|)
\]

• Problem: Middlebox state space exponential in number of hosts

• Solution: apply Cartesian abstraction
  • Ignore some correlations \textit{within} a middlebox state
Middlebox-level Abstraction

\[ \text{Time(LFP#)} = \text{poly(|M|, |S|, |E|, |P|)} \]

- Problem: Middlebox state space exponential in number of hosts
- Solution: apply Cartesian abstraction
  - Ignore some correlations within a middlebox state
- How to decompose a state into sub-states?
Packet state

- Alternative (isomorphic) state representation

- Depends on AMDL: the restricted way in which middleboxes query and update their state
Packet state example

\[(src, dst, type) \mapsto \{\text{queries which hold}\}\]

\[(1,_,_,) \mapsto {}\]
\[(2,_,_,) \mapsto {}\]

```
hole_punching_firewall = // hosts ∈ \{1, 2\} 
  port_in ? <src,dst,tpe> => 
    trusted(dst) := true; port_ext ! <src,dst,tpe>
  | 
  port_ext ? <src,dst,tpe> => 
    srcT: src in trusted => port_in ! <src,dst,tpe>
```
Packet state example

\[(src, dst, type) \mapsto \{\text{queries which hold}\}\]

\((1, _, _) \mapsto \{\}\)
\((2, _, _) \mapsto \{\}\)

\[\text{hole_punching_firewall} = \quad // \text{hosts} \in \{1, 2\}
\quad \text{port}\_\text{in} ? <src, dst, tpe> =>
\quad \quad \text{trusted}(dst) := \text{true}; \text{port}\_\text{ext} ! <src, dst, tpe>
\quad |\]
\quad \text{port}\_\text{ext} ? <src, dst, tpe> =>
\quad \quad \text{srcT: src in trusted} \Rightarrow \text{port}\_\text{in} ! <src, dst, tpe>\]
Packet state example

Let $(\text{src}, \text{dst}, \text{type}) \mapsto \text{\{queries which hold\}}$

- $(1,\_\_,\_\_) \mapsto \{\}$
- $(2,\_\_,\_\_) \mapsto \{\}$

```plaintext
hole_punching_firewall = // hosts ∈ {1, 2}
  port_in ? <src,dst,tpe> =>
    trusted(dst) := true; port_ext ! <src,dst,tpe>
  |
  port_ext ? <src,dst,tpe> =>
    srcT: src in trusted => port_in ! <src,dst,tpe>
```
Packet state example

\[(src, dst, type) \mapsto \{\text{queries which hold}\}\]

\[ (1,_,_) \mapsto \emptyset \]
\[ (2,_,_) \mapsto \emptyset \]

```
hole_punching_firewall = // hosts ∈ \{1, 2\}
port_in ? <src,dst,tpe> =>
    trusted(dst) := true; port_ext ! <src,dst,tpe>
| port_ext ? <src,dst,tpe> =>
    srcT: src in trusted => port_in ! <src,dst,tpe>
```
Packet state example

\[(\text{src, dst, type}) \mapsto \{\text{queries which hold}\}\]

\[(1,\_\_\_) \mapsto \{\} \quad (1,\_\_\_) \mapsto \{\text{srcT}\} \quad (2,\_\_\_) \mapsto \{\} \quad (2,\_\_\_) \mapsto \{\}\]

\[\text{hole_punching_firewall} = \quad // \text{hosts } \in \{1, 2\}
\quad \text{port_in ? } <\text{src, dst, tpe}> \Rightarrow
\quad \quad \text{trusted(dst)} := \text{true}; \text{port_ext ! } <\text{src, dst, tpe}>
\quad | \quad \text{port_ext ? } <\text{src, dst, tpe}> \Rightarrow
\quad \quad \text{srcT: src in trusted} \Rightarrow \text{port_in ! } <\text{src, dst, tpe}>\]
Packet state example

\((\text{src, dst, type}) \mapsto \{\text{queries which hold}\}\)

\( (1, _, _) \mapsto \{\} \)
\( (2, _, _) \mapsto \{\} \)

\( (1, _, _) \mapsto \{\text{srcT}\} \)
\( (2, _, _) \mapsto \{\} \)

\[
\text{hole_punching_firewall} = \quad // \text{hosts} \in \{1, 2\} \\
\text{port_in} ? <\text{src, dst, tpe}> \Rightarrow \\
\quad \text{trusted}(\text{dst}) := \text{true}; \text{port_ext} ! <\text{src, dst, tpe}> \\
\quad | \\
\quad \text{port_ext} ? <\text{src, dst, tpe}> \Rightarrow \\
\quad \text{srcT: src in trusted} \Rightarrow \text{port_in} ! <\text{src, dst, tpe}>
\]
Packet state example

\[(src, dst, type) \mapsto \{\text{queries which hold}\}\]

\[
\begin{align*}
(1,\_\_\_) & \mapsto \{} \\
(2,\_\_\_) & \mapsto \{}
\end{align*}
\]

\[
\begin{align*}
(1,\_\_\_) & \mapsto \{\text{srcT}\} \\
(2,\_\_\_) & \mapsto \{}
\end{align*}
\]

```
hole_punching_firewall = // hosts \in \{1, 2\}
  port_in ? <src,dst,tpe> =>
    trusted(dst) := true; port_ext ! <src,dst,tpe>
  |
  port_ext ? <src,dst,tpe> =>
    srcT: src in trusted => port_in ! <src,dst,tpe>
```

Query name: hole_punching_firewall

Query:
```
hole_punching_firewall = // hosts \in \{1, 2\}
  port_in ? <src,dst,tpe> =>
    trusted(dst) := true; port_ext ! <src,dst,tpe>
  |
  port_ext ? <src,dst,tpe> =>
    srcT: src in trusted => port_in ! <src,dst,tpe>
```
Cartesian packet state example

\[(1, _, _) \mapsto \{\} \{\text{srcT}\}\]
\[(2, _, _) \mapsto \{\} \{\text{srcT}\}\]

\text{hole_punching_firewall} = \quad // \text{hosts} \in \{1, 2\}
\quad \text{port_in} ? \langle \text{src, dst, tpe} \rangle \Rightarrow
\quad \quad \text{trusted}(\text{dst}) := \text{true}; \text{port_ext} ! \langle \text{src, dst, tpe} \rangle
\quad |\n\quad \text{port_ext} ? \langle \text{src, dst, tpe} \rangle \Rightarrow
\quad \quad \text{srcT: src in trusted} \Rightarrow \text{port_in} ! \langle \text{src, dst, tpe} \rangle
Summary: Network Abstractions

(1) Unordered channels
(2) Counter abstraction on channels
(3) Network-level Cartesian Abstraction
Summary: Network Abstractions

(1) Unordered channels

(2) Counter abstraction on channels

(3) Network-level Cartesian Abstraction
Summary: Network Abstractions

(1) Unordered channels

(2) Counter abstraction on channels

(3) Network-level Cartesian Abstraction

(4) Middlebox-level Cartesian abstraction
   • No correlations between packet states
   • But keep correlations between queries

\[ \mathcal{A} = (M \rightarrow P \rightarrow \mathbb{P}(\mathbb{P}(Q)) \times (E \rightarrow \mathbb{P}(P))) \]

Time(LFP\#) = poly(|M|, |P|, 2^{|Q|}, |E|)
When is this precise?
Reverting Middlebox Abstraction

Cartesian abstraction ≈ Reverting middlebox abstraction

• Let middleboxes independently revert to their initial state
Example: Firewall

a is trusted
Reverting Middlebox Abstraction

**Theorem**: If the network is correct in the presence of packet reordering and middlebox reverts then our analysis is precise.

- Common wisdom: Network resets make verification harder
  - Reachability for Petri nets with resets is undecidable

- But: Simplifies the task of automatic verification of networks
  - The analysis is precise for isolation
  - No false alarms
Initial Experimental Results

Network Configuration & Topology

Middlebox Model

Compiler

Datalog Rules

LogicBlox

Packet Space States
Scalability Testing - Hosts

• Enterprise network with 3 subnets
  • Each with a different security policy
• Isolation between *quarantined* and *Internet*
Scalability Testing - Middleboxes

• Servers with parallel middlebox chains
• Scaled the number of chains
• Isolation – packets from $h_1$ never reach bottom flow

![Diagram of Scalability Testing - Middleboxes](image_url)
Summary

• Abstract interpretation of stateful networks
  • Unordered + Counter + Cartesian X2

• AMDL – Abstract Middlebox Definition Language

• Packet effect semantics for middleboxes
  • Enables middlebox-level Cartesian abstraction

• Precise for unordered channels + reverting middleboxes
Further Work

- Correlated middlebox states
- Temporal properties
- Parameterized case

Seeking students and postdocs

[erc logo] Supervised Verification of Infinite-State Systems