An Assertion Language for Debugging SDN Applications

Ryan Beckett
with
X. Kelvin Zou, Shuyuan Zhang, Sharad Malik, Jennifer Rexford, David Walker

Princeton University
Data Plane Verification

- Find common misconfigurations
- Operate in real time
- Check fixed network properties
- Can report false positives during transitions
Stateful Firewall

State Table

<table>
<thead>
<tr>
<th>client</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stateful Firewall

A server can only reach a client if the client has communicated with the server

Firewall Property:
Stateful Firewall

“A server can only reach a client if the client has communicated with the server”

Firewall Property:
“A server can only reach a client if the client has communicated with the server”
Stateful Firewall

State Table

<table>
<thead>
<tr>
<th>client</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>S₁</td>
</tr>
</tbody>
</table>

Firewall Property:
reachable(s₁, c₁)
Stateful Firewall

State Table

<table>
<thead>
<tr>
<th>client</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>S₁</td>
</tr>
<tr>
<td>C₂</td>
<td>S₁</td>
</tr>
</tbody>
</table>

Firewall Property:

reachable(s₁, c₁) \land reachable(s₁, c₂)
Stateful Firewall

State Table

<table>
<thead>
<tr>
<th>client</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>S₁</td>
</tr>
<tr>
<td>C₂</td>
<td>S₁</td>
</tr>
</tbody>
</table>

Firewall Property:
∀(c,s)∈table, reachable(s, c)
Stateful Firewall

State Table

Firewall Property:

∀c∈clients, ∀s∈servers,
reachable(s, c) ↔ (c, s)∈table
Stateful Firewall

![Diagram of network setup with state table and controller code]

Controller Code:

```python
assert_continuously(f)

def packet_in(event):
    pkt = event.parsed
    if pkt.typ != eth.IP_TYP:
        return
    ...
Stateful Firewall

<table>
<thead>
<tr>
<th>client</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Controller Code:

```python
assert_continuously(f)
```
Stateful Firewall

State Table

<table>
<thead>
<tr>
<th>client</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>S₁</td>
</tr>
</tbody>
</table>

Controller Code:

```
assert_continuously(f)
```
Stateful Firewall

State Table

Controller Code:

```
assert_continuously(f)
stop(f)
install(r1)
install(r2)
assert_continuously(f)
```
Design Overview
Design Overview
Design Overview

Controller

Assertion Library

Property Generator

Data Plane Verifier
Design Overview

Controller

Assertion Library

Property Generator

Data Plane Verifier
Design Overview

- Controller
- Assertion Library
- Property Generator
- Data Plane Verifier
Design Overview

Controller

Assertion Library

Property Generator

Data Plane Verifier

event

assertion

state update

verification conditions
Design Overview

Controller

Assertion Library

Property Generator

Data Plane Verifier

event

exception

verification results
Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
  - Incrementally generate new verification conditions
Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
  - **Incrementally** generate new verification conditions

Firewall Property:
reachable(s₁, c₁)
Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
  - Incrementally generate new verification conditions

Firewall Property:
reachable(s₁, c₁) ∧ reachable(s₁, c₂)
Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
  - Incrementally generate new verification conditions
  - Precompute and cache intermediate results

**Firewall Property:**
reachable(s₁, c₁)
Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
  - Incrementally generate new verification conditions
  - Precompute and cache intermediate results

Firewall Property:

\[
\text{reachable}(s_1, c_1) \land \text{reachable}(s_1, c_2)
\]
Incremental Data Structure

Firewall Property:
\[ \forall c \in \text{clients}, \forall s \in \text{servers}, \]
\[ \text{reachable}(s, c) \leftrightarrow (c, s) \in \text{table} \]
Incremental Data Structure

Firewall Property:
∀c ∈ clients, ∀s ∈ servers,
reachable(s, c) ⇔ (c, s) ∈ table
Incremental Data Structure

Firewall Property:
∀ c ∈ clients, ∀ s ∈ servers,
reachable(s, c) ↔ (c, s) ∈ table
Incremental Data Structure

Firewall Property:
\[ \forall c \in \text{clients}, \forall s \in \text{servers}, \]
\[ \text{reachable}(s,c) \leftrightarrow (c,s) \in \text{table} \]
Incremental Data Structure

Firewall Property:
∀c ∈ clients, ∀s ∈ servers, reachable(s, c) ↔ (c, s) ∈ table
Incremental Data Structure

Firewall Property:
\[ \forall c \in \text{clients}, \forall s \in \text{servers}, \]
\[ \text{reachable}(s,c) \leftrightarrow (c,s) \in \text{table} \]
Incremental Data Structure

Firewall Property:
\[ \forall c \in \text{clients}, \forall s \in \text{servers}, \]
\[ \text{reachable}(s, c) \leftrightarrow (c, s) \in \text{table} \]
Prototype Implementation

- Python assertion debugging library
- Support for Pyretic, Pox, Ryu
- Uses the VeriFlow verification tool
- Initial performance is promising

Stateful Firewall (incremental vs naive)
Conclusion

- Assertions to verify *dynamic* properties
- Programmatic control over verification *timing*
- Incremental algorithm to verify dynamic assertion properties
- Prototype with reasonable performance