Finding Semantic Bugs in Intent-Based Networking with Fuzzing

Jiwon Kim
Purdue University
Nov 16, 2023
What is Intent-Based Networking?
Facebook Blames Outage on Error During Routine Maintenance

Associated Press | October 6, 2021 11:06 am

South Korean telco KT suffers nationwide outage after routing error

Human error behind network outage, Sure confirms

3 February

“Allow access from marketing team to database via load balancer.”
Translation

“Allow access from marketing team to database via load balancer.”

Network-level Intent(s)

- SRC: MKT
- DST: DB
- Waypoint: LB
Compilation

Network-level Intent(s)

- SRC: MKT
- DST: DB
- Waypoint: LB

Network Object(s)

```plaintext
config {
    service: db.aa.com
    vip: DB
    port: 3306
    backends: [DB1, 2, 3]
}
```

<table>
<thead>
<tr>
<th>IN</th>
<th>SRC</th>
<th>DST</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MKT1</td>
<td>DB</td>
<td>LB</td>
</tr>
<tr>
<td>2</td>
<td>MKT2</td>
<td>DB</td>
<td>LB</td>
</tr>
</tbody>
</table>
Activation

Network Object(s)

Data Plane

Data Plane

Network Devices

Control Plane

Network Devices

config {
  service: db.aa.com
  vip: DB
  port: 3306
  backends: [DB1, 2, 3]
}

<table>
<thead>
<tr>
<th>IN</th>
<th>SRC</th>
<th>DST</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MKT1</td>
<td>DB</td>
<td>LB</td>
</tr>
<tr>
<td>2</td>
<td>MKT2</td>
<td>DB</td>
<td>LB</td>
</tr>
</tbody>
</table>
Telemetry Data

[S1] RX/TX
MKT1: 20mbps/20kbps
MKT2: 980mbps/20kbps
LB:   40kbps/1gbps
Verification

Telemetry Data

INSTALLED, 98% Traffic from MKT2

Result

[S1] RX/TX
MKT1: 20mbps/20kbps
MKT2: 980mbps/20kbps
LB: 40kbps/1gbps
Summary

Result

INSTALLED, 98% Traffic from MKT2

In S1, 98% of traffic to LB comes from MKT2
“Allow access from marketing team to database via load balancer without congestion.”
Intent-Based Networking

- Translation
- Compilation
- Activation
- Optimization
- Verification
- Observation

Intent State Machine

- REQ
- COMPILING
- INSTALLING
- INSTALLED
- FAILED
- REMOVED

INSTALL ➔ DELETE
SDN and IBN

Software-Defined Networking

Intent-Based Networking
SDN and IBN

Software-Defined Networking

Intent-Based Networking
SDN and IBN

Software-Defined Networking

Intent-Based Networking
SDN and IBN

Software-Defined Networking

Intent-Based Networking
SDN and IBN

Software-Defined Networking

Intent-Based Networking
SDN and IBN

Software-Defined Networking

Intent-Based Networking
Vulnerabilities in SDN and IBN

- 1st OpenFlow Controller: NOX
- Google’s B4
- ODL
- ONOS with IBN
- ODL IBN
- Cisco DNAC
- ONAP IBN
- Google’s Orion

Cumulative CVEs

- SDN
- IBN

Timeline:
- 2008: 1st OpenFlow Controller: NOX
- 2013: Google’s B4
- 2014: ODL
- 2016: ONOS with IBN
- 2017: ODL IBN
- 2020: Cisco DNAC
- 2021: ONAP IBN
- 2021: Google’s Orion

Graph showing cumulative CVEs from 2014 to 2023H1.
Can we run a **fuzzer** on Intent-Based Networking?
Fuzzing Programs

Fuzzer → Random Input → Target Program → Feedback

Coverage

Crash
Fuzzing IBN [1/3]

Fuzzer → Random Intent → Intent-Based Networking

Feedback → Coverage → Crash
I. Bug Study in ONOS IBN

Semantic bugs often do not cause program crashes.

→ Domain-specific detection methods are necessary.
Fuzzing IBN [2/3]

Fuzzer

Random Intent

Intent-Based Networking
- Translation
- Optimization
- Compilation
- Verification
- Activation
- Observation

Feedback

Coverage

Many Semantic Bugs
II-1. Limitation in Input Generation

The success of intent (input) depends on network.

→ Topology-aware input generation is needed.
## II-2. Bug Study in ONOS IBN

Table 1: Bug analysis of the ONOS intent subsystem.

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Impact</th>
<th>Detection Mechanism</th>
<th># Bugs Based on Intent Operation of Root Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>submit withdraw purge topo-change Total</td>
</tr>
<tr>
<td>Intent syntactic bug</td>
<td>SYN1</td>
<td>Denied intent request</td>
<td>Input validation</td>
<td>2 - - -</td>
</tr>
<tr>
<td></td>
<td>SYN2</td>
<td>Not found</td>
<td>Input validation</td>
<td>9 - - -</td>
</tr>
<tr>
<td></td>
<td>SYN3</td>
<td>Wrong intent data</td>
<td>Input validation</td>
<td>3 - - -</td>
</tr>
<tr>
<td></td>
<td>SYN4</td>
<td>Corrupt intent</td>
<td>Input validation</td>
<td>1 - - -</td>
</tr>
<tr>
<td>Other syntactic bug</td>
<td>SYN5</td>
<td>Controller shutdown</td>
<td>Application agent</td>
<td>1 - - -</td>
</tr>
<tr>
<td></td>
<td>SYN6</td>
<td>Resource exhaustion</td>
<td>Resource agent</td>
<td>1 2 - -</td>
</tr>
<tr>
<td></td>
<td>SYN7</td>
<td>Topology disband</td>
<td>Application agent</td>
<td>2 1 - -</td>
</tr>
<tr>
<td></td>
<td>SYN8</td>
<td>Throughput drop</td>
<td>Performance test</td>
<td>15 - - -</td>
</tr>
<tr>
<td></td>
<td>SYN9</td>
<td>Exceptions</td>
<td>Log detection</td>
<td>29 6 2 5 42</td>
</tr>
<tr>
<td>Intent semantic bug</td>
<td>SEM1</td>
<td>Inconsistent intent state</td>
<td>Control-Plane (CP) test</td>
<td>17 10 7 16 50</td>
</tr>
<tr>
<td></td>
<td>SEM2</td>
<td>Failure in connectivity</td>
<td>Data-Plane (DP) test</td>
<td>29 2 - 14 45</td>
</tr>
<tr>
<td></td>
<td>SEM3</td>
<td>Impact on existing intent</td>
<td>CP/DP tests</td>
<td>6 2 - -</td>
</tr>
<tr>
<td></td>
<td>SEM4</td>
<td>Garbage flow rules</td>
<td>Flow-Intent mapping</td>
<td>6 4 - 1 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>121 27 9 37 194 (1862)</td>
</tr>
</tbody>
</table>

### Diagram

- ADD
- DEL
- MOD
- LINK
- HOST
- SWITCH
Fuzzing IBN [3/3]

Multiple-Operation Support: C/U/D & Topology Change

Intent-Based Networking
- Translation
- Optimization
- Compilation
- Verification
- Activation
- Observation

Feedback
Coverage

Many Semantic Bugs
III. Limitation in Code-Coverage Guidance

Code coverage cannot differentiate two cases.

The order of operations (intent-state changes) matters.

Case 1: INS-INS-DEL

Case 2: INS-DEL-INS

Install x 2
Delete x 1
Compile x 2
Flow_add x 2
Flow_del x 1

Install x 2
Delete x 1
Compile x 2
Flow_add x 2
Flow_del x 1

Alice
Bob
Limitations in Fuzzing IBN

Topography-Aware Intents

Multiple-Operation Support:
C/U/D & Topology Change

Intent-Based Networking

Translation
Compilation
Activation
Optimization
Verification
Observation

* Black-box Approach

New Coverage Metric for Feedback

Many Semantic Bugs
Intender: Fuzzing IBN

Temperature-Aware Intent Generator (TAIG)

Intent-State Transition Guidance (ISTG)

Detecting Syntactic and Semantic Bugs

Multiple-Operation Support with Intent-Operation Dependency (IOD)

Black-box Approach
Topology-Aware Intent Generator [1/3]

PointToPoint Intent

```
{"intent": {
  "type": "PointToPointIntent",
  "appId": "org.onosproject.null",
  "priority": 55,
  "ingressPoint": {"device": "of:0000000000000002", "port": "2"},
  "egressPoint": {"device": "of:0000000000000001", "port": "2"}
}}
```

HostToHost Intent

```
{"intent": {
  "type": "HostToHostIntent",
  "appId": "org.onosproject.null",
  "priority": 55,
  "one": "12:FB:8D:0B:04:42/None",
  "two": "92:BF:DE:1D:F3:06/None"
}}
```
Topology-Aware Intent Generator (cont’d) [1/3]

- Config Reader
- Fuzz Module
  - Initialize
  - Read, Mutate, and Write

Configurational Topo Store
Operational Topo Store
Topology-Aware Intent Generator (cont’d) [1/3]

- Config Reader
  - Initialize
- Fuzz Module
  - Read, Mutate, and Write
- Configurational Topo Store
- Operational Topo Store
- Run scenario
- Apply actual result (e.g., create-topo, link-del)
- Intender
*Topology-Aware Intent Generator (cont’d) [1/3]*

- **Config Reader**
  - Initialize
  - Read, Mutate, and Write

- **Configurational Topo Store**
  - Compare
  - Run scenario

- **Fuzz Module**
  - Apply actual result (e.g., create-topo, link-del)

- **Operational Topo Store**

- **Intender**
Intent-State Transition Guidance (ISTG) [2/3]

1. Get seed
2. Mutate based on IOD/TAIG
3. Execute action
4. Read intent state
5. Record all transitions
6. Store scenario with new transitions

Seed Scenario

Intender

Mutant Scenario

Intent Transition History

<table>
<thead>
<tr>
<th>Scenario</th>
<th>INSTALL</th>
<th>MODIFY</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTALL</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>INSTALL</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>add-link</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Intent-Based Networking

- Translation
- Optimization
- Compilation
- Verification
- Activation
- Observation
Detecting Connection Failure [3/3]

How to send and sniff packets on link ports?
Detecting Connection Failure [3/3]
1. Point-interleaving switches deliver packets.
2. Point-interleaving hosts sends or sniffs packets.
Intender as a Framework

IBN-Fuzzing Framework

- AFL
- Jazzer
- Zest (JQF)
- PAZZ (SDN)

Detecting Syntactic and Semantic Bugs

- (1) TAIG & IOD
- (2) Random
- (3) Grammar
- (4) Packet Generator

(1) ISTG
(2) CCG
Intender as a Framework: Multi-Layer Fuzzing
Evaluation [1/4]

• Environment Setup
  • Google Cloud VM: 4 vCPU, 16GB MEM, 60GB SSD
  • ONOS v2.5.1

• Found 12 new bugs (11 security-critical CVEs)
  • 9 semantic bugs
  • Security impacts: network-wide denial of service & tampering

• Compare 4 existing fuzzers (AFL, Jazzer, Zest, PAZZ)
  • Up to 2.2× better in branch coverage
  • Up to 82.6× more number of unique errors
Evaluation [2/4]

- Improve fuzzing performance compared to baselines
  - Topology-Aware Input Generation (TAIG) can produce $78.7\times$ more valid intents
  - Intent-Operation Dependency (IOD) can reduce 73.02% of redundant operations
  - Intent-State Transition Guidance (ISTG) leads to $1.8\times$ more intent-state transitions than code coverage guidance (CCG)
Evaluation [3/4]

(a) Number of operations.
(b) Number of intent states.
(c) Branch coverage.
(d) Number of unique errors.
(e) Cumulative sum of bugs found.
(f) Mean time to find each bug.

The chart illustrates the time-to-find bugs and the reliability of finding them using different tools: AFL, Jazzer, Zest, and Intender. The y-axis represents the time to find bugs in seconds, and the x-axis represents the bug ID. The chart shows that some bugs are found only by Intender, indicating a higher efficiency in finding semantic bugs. The reliability percentage is also shown on the right side of the chart.
Case Study: CVE-2022-24035

(1) Eve requests PURGE on INSTALLED intent

Intent Manager
API Handlers
Batch & Process

Objective Tracker
Topology Change Handler

Pending Requests
Current Intents

① Wrong request remains in Pending because of bug.

Eve

ONOS Controller
Data Plane Network

API Handlers
Batch & Process
Case Study: CVE-2022-24035

(2) Eve exploits link-flooding attack

Intent Manager
- API Handlers
- Batch & Process
- Topology Change Handler

Intent Store
- Pending Requests
- Current Intents

ONOS Controller
Data Plane Network

Intent should be recompiled if the path goes invalid.
Case Study: CVE-2022-24035

(3) Intent **DOES NOT** respond to topology event any more → DoS⚠️

![Diagram of Intent Manager and Intent Store](image)

- **Intent Manager**
  - API Handlers
  - Batch & Process
- **Objective Tracker**
  - Topology Change Handler

**Intent Store**
- Pending Requests
- Current Intents

③ Topology event is ignored to wait **Pending** request.

**Data Plane Network**

![Eve](image)
Case Study: CVE-2022-24035

(3) Intent **DOES NOT** respond to topology event any more

- **To discover this bug efficiently…**

**Intent Transition History**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTALL</td>
<td>!</td>
</tr>
<tr>
<td>PURGE</td>
<td>!</td>
</tr>
<tr>
<td>del-link</td>
<td>!</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Intender**
  - Transitions:
    - INSTALL
    - PURGE
    - Topology change (del-link)
  - Intent Manager
  - API Handlers
  - Batch & Process
  - Objective Tracker
  - Topology Change Handler
  - Data Plane Network
  - ONOS Controller

- **Eve**
  - LINK DOWN

**Objective:**

- To discover this bug efficiently…
Conclusions of Intender

- Analyzed **186 bugs** in ONOS IBN
- Designed new fuzzing techniques for IBN
  - Topology-Aware Intent Generation (**TAIG**)  
  - Intent-Operation Dependency (**IOD**)  
  - Intent-State Transition Guidance (**ISTG**)  
- Developed Intender architecture with 4 fuzzers
  - AFL, Jazzer, Zest, PAZZ  
- Found **12 new bugs (11 CVEs)** in ONOS IBN
Future Research on IBN Security

The marketing team can access the database via a load balancer.

Translation → Compilation → Activation → Observation → Verification → Optimization

Network Intent

- SRC: MKT
- DST: DB
- Waypoint: LB

Group Table

<table>
<thead>
<tr>
<th>Group</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKT</td>
<td>A, B, C</td>
</tr>
<tr>
<td>DB</td>
<td>D, E, F</td>
</tr>
</tbody>
</table>

Topology

- SRC: MKT
- DST: DB
- Waypoint: LB

Group Table

<table>
<thead>
<tr>
<th>Group</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKT</td>
<td>A, B, C</td>
</tr>
<tr>
<td>DB</td>
<td>D, E, F</td>
</tr>
</tbody>
</table>

S1 Flow Rules

<table>
<thead>
<tr>
<th>IN SRC</th>
<th>DST</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>DB</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>DB</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>DB</td>
</tr>
</tbody>
</table>

LB Config

```config
service: db.aa.com
vip: DB
port: 3306
backends: [D, E, F]
```

S2 Flow Rules

<table>
<thead>
<tr>
<th>IN SRC</th>
<th>DST</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>LB</td>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>LB</td>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>

Monitoring Data

[S1] RX/TX
1: 100mbps/20kbps
2: 100mbps/20kbps
3: 800mbps/20kbps
LB: 60kbps/1gbps

In S1, 80% of traffic to LB comes from Host C.

Report

The marketing team can access the database via a load balancer, 300 mbps bandwidth for each host.
Thank you!

Questions: kim1685@purdue.edu
# Found Bugs

<table>
<thead>
<tr>
<th>#</th>
<th>CVE ID</th>
<th>Type</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVE-2021-38363</td>
<td>SYN2</td>
<td>add-intent</td>
<td>PointToPoint intent with invalid point field causes NullPointerException</td>
</tr>
<tr>
<td>2</td>
<td>CVE-2022-29604</td>
<td>SYN4</td>
<td>add-intent</td>
<td>PointToPoint intent which has an upper-case letter in a device ID shows CORRUPT</td>
</tr>
<tr>
<td>3</td>
<td>CVE-2022-29606</td>
<td>SYN4</td>
<td>add-intent</td>
<td>PointToPoint intent which has a large switch port number shows CORRUPT</td>
</tr>
<tr>
<td>4</td>
<td>CVE-2022-29609</td>
<td>SEM1</td>
<td>add-intent</td>
<td>HostToHost intent with the same source and destination shows INSTALLING</td>
</tr>
<tr>
<td>5</td>
<td>CVE-2022-29608</td>
<td>SEM2</td>
<td>add-intent</td>
<td>PointToPoint intent installs an invalid flow rule causing network loop</td>
</tr>
<tr>
<td>6</td>
<td>CVE-2022-29605</td>
<td>SEM2</td>
<td>add-intent</td>
<td>Intent tries to install IPv6 flow rules into OF10 switches</td>
</tr>
<tr>
<td>7</td>
<td>CVE-2022-29944</td>
<td>SEM3</td>
<td>add-intent</td>
<td>Intent cannot bypass intents with higher priority</td>
</tr>
<tr>
<td>8</td>
<td>CVE-2021-38364</td>
<td>SEM3</td>
<td>add-intent</td>
<td>Intent can delete or modify flow rules of previous intents which share the path</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>SEM4</td>
<td>add-intent</td>
<td>PointToPoint intent with switch port 0 installs useless flow rules</td>
</tr>
<tr>
<td>10</td>
<td>CVE-2022-24109</td>
<td>SEM3</td>
<td>withdraw-intent</td>
<td>Deletion of one of the duplicate intents removes all flow rules</td>
</tr>
<tr>
<td>11</td>
<td>CVE-2022-29607</td>
<td>SEM1</td>
<td>mod-intent</td>
<td>HostToHost intent modified to have same source and destination shows INSTALLED without any flow rules</td>
</tr>
<tr>
<td>12</td>
<td>CVE-2022-24035</td>
<td>SEM1</td>
<td>purge-intent &amp; topology-change</td>
<td>After requesting purge on installed PointToPoint intent, the state of intent does not change to FAILED with link failure</td>
</tr>
</tbody>
</table>