NS2 Network and System Security Laboratory KAIST SRI International

DELTA: A Security Assessment Framework for Software-Defined Networks

SEUNGSOO LEE†, CHANGHOON YOON†, CHANHEE LEE†, SEUNGWON SHIN†, VINOD YEGNESWARAN‡, PHILLIP PORRAS‡

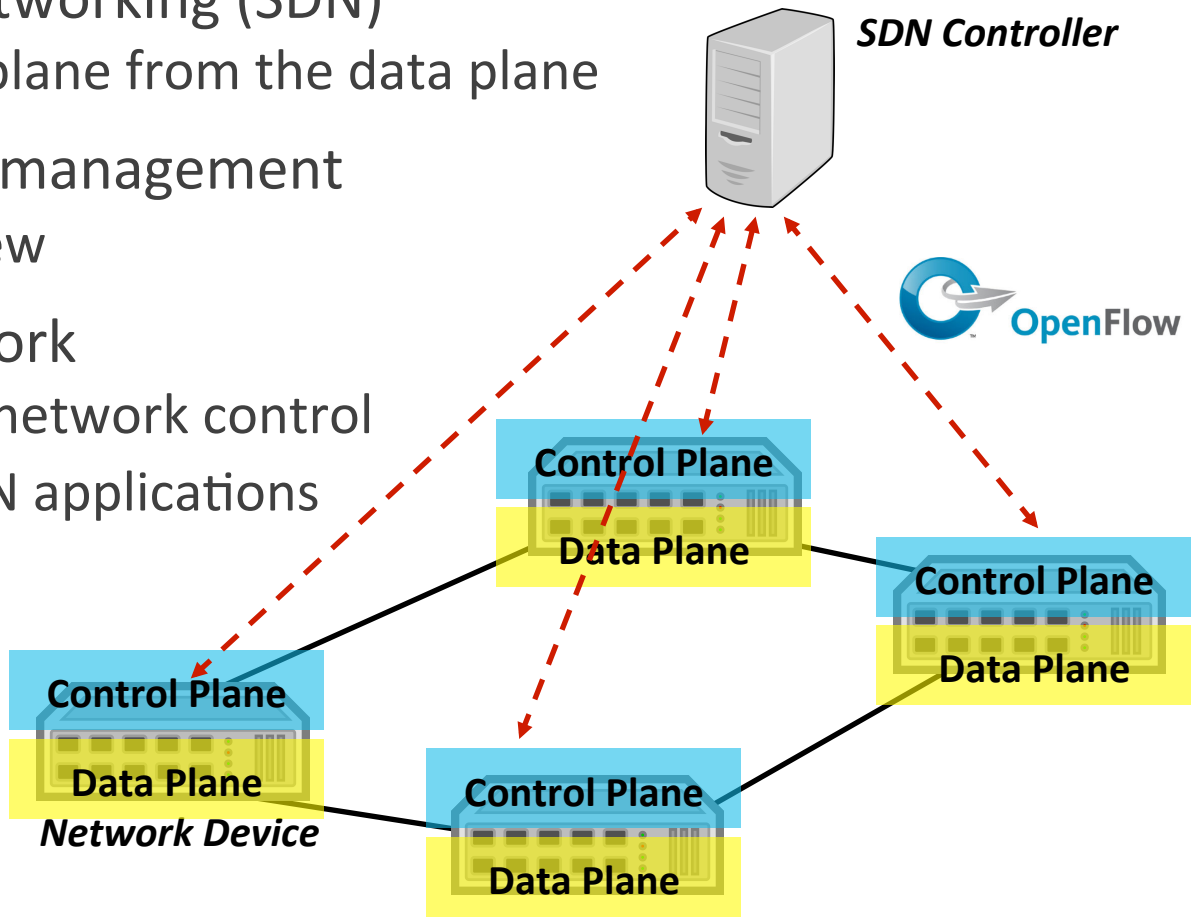
† KAIST ‡SRI INTERNATIONAL

Outline

1. **Background and Motivation**
2. System Design
3. Blackbox Fuzzing
4. Implementation
5. Evaluation
6. Conclusion

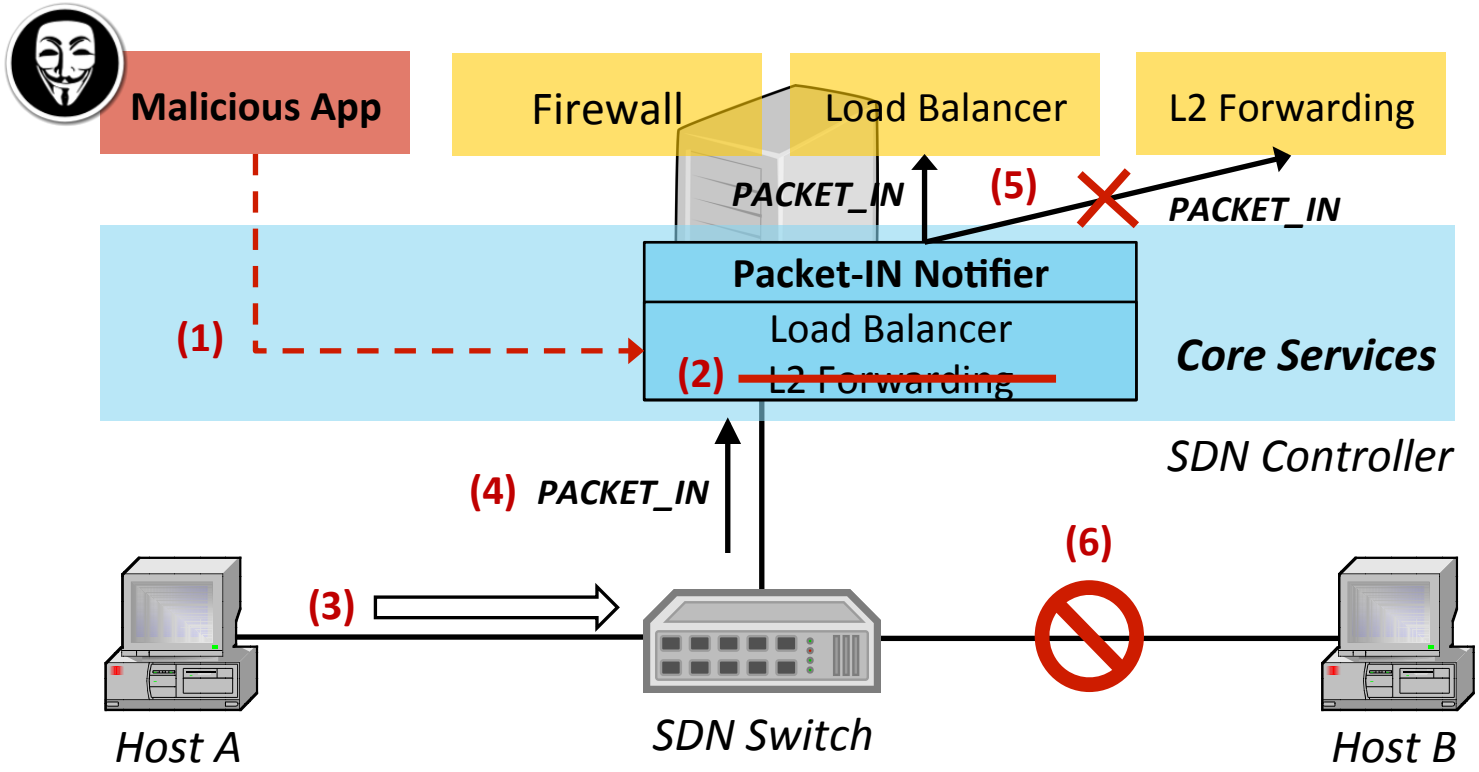
What is Software-defined Networking?

- Software Defined Networking (SDN)
  - Separate the control plane from the data plane
- **Centralized** network management
  - Via global network view
- **Programmable** network
  - Flexible and dynamic network control
  - Useful, innovative SDN applications
- **OpenFlow** protocol
  - A de-facto standard



Motivating Example

- Event Listener Unsubscription attack [1]



[1] http://sdnsecurity.org/vulnerability/AttackList.html

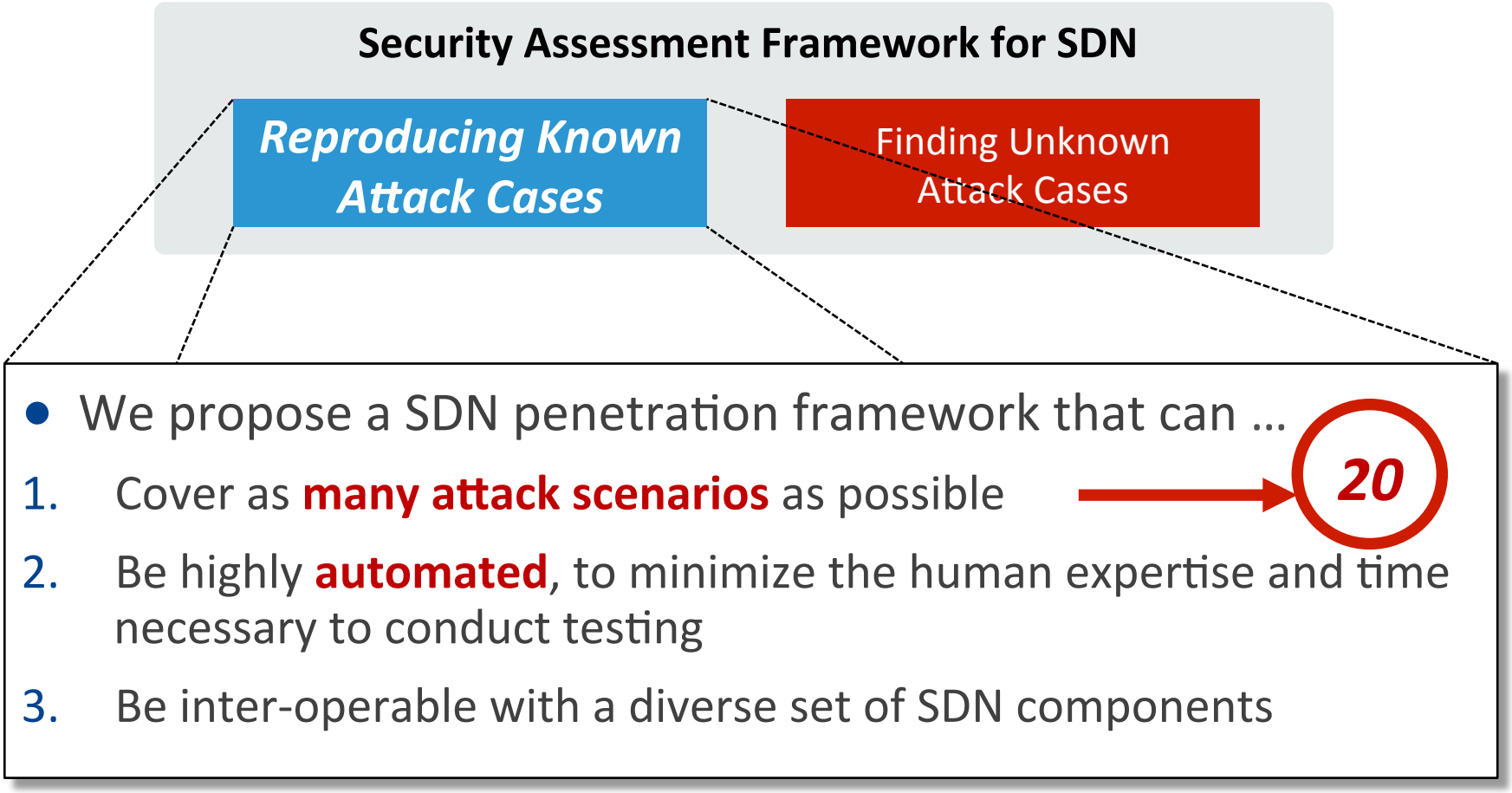
A network operator wants to know …

Is my SDN secure?

A Security Assessment Framework for Software-Defined Networks

- **Which vulnerabilities exist now?**
- **How to reproduce each test case?**
- **Any more vulnerabilities?**
- **…**

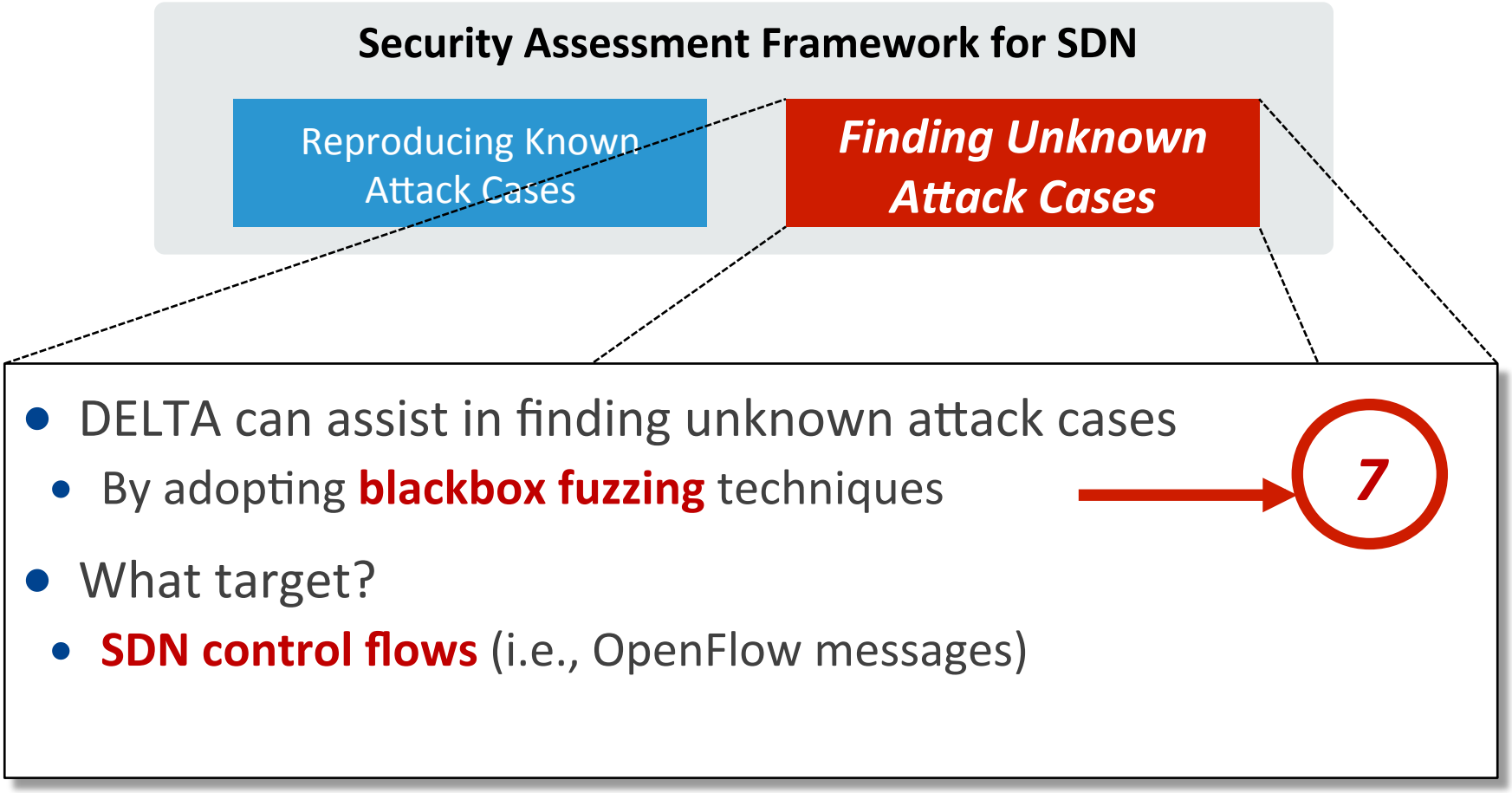
DELTA: A Security Assessment Framework for SDN



- We propose a SDN penetration framework that can …

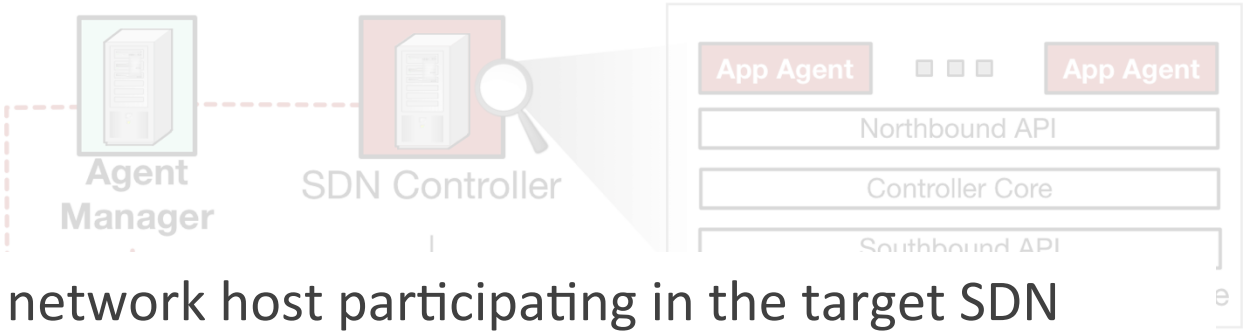
1. Cover as **many attack scenarios** as possible → **20**
2. Be highly **automated**, to minimize the human expertise and time necessary to conduct testing
3. Be inter-operable with a diverse set of SDN components

DELTA: A Security Assessment Framework for SDN

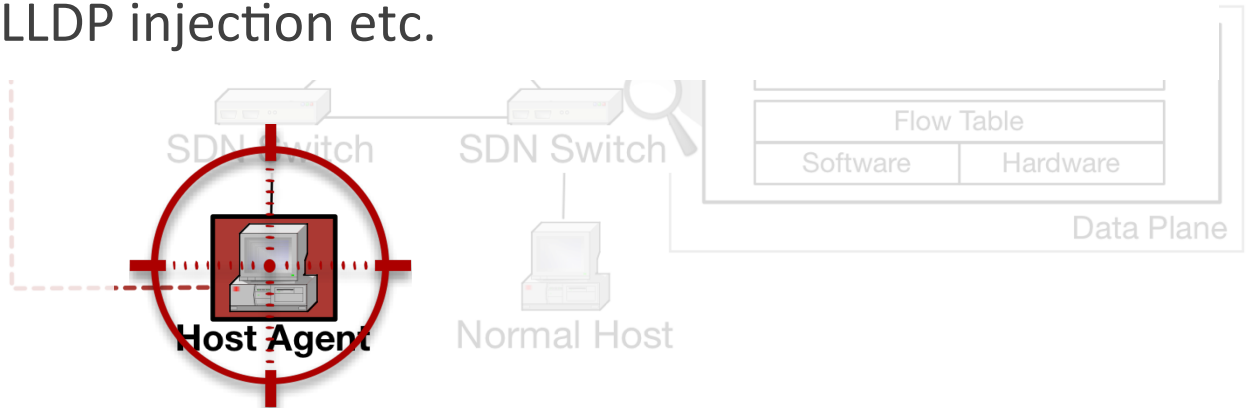


System Design

- Host agent

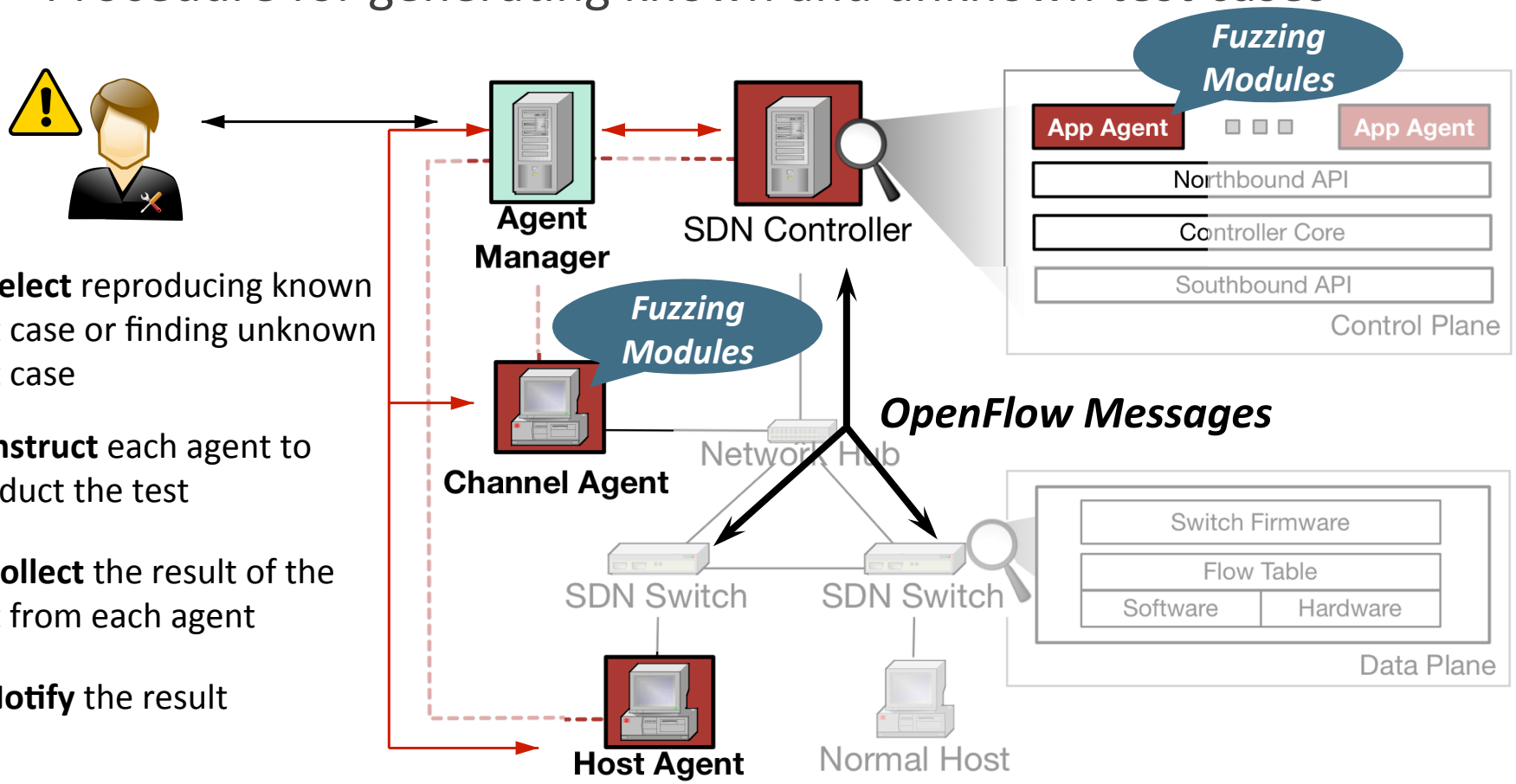


  - A legitimate network host participating in the target SDN
  - Generates **network traffic** as instructed by the agent manager
    - e.g. DDoS, LLDP injection etc.



Basic Operation

- Procedure for generating known and unknown test cases



1. **Select** reproducing known test case or finding unknown test case

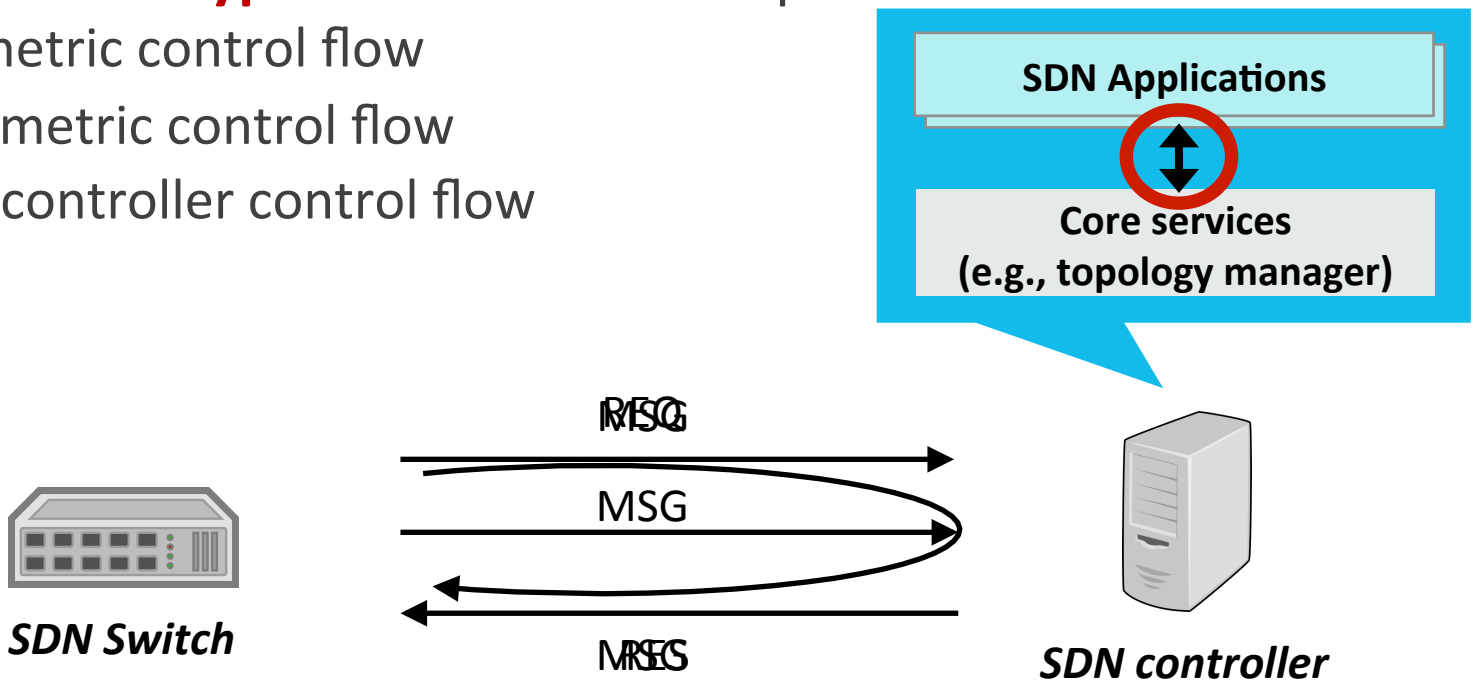
2. **Instruct** each agent to conduct the test

3. **Collect** the result of the test from each agent

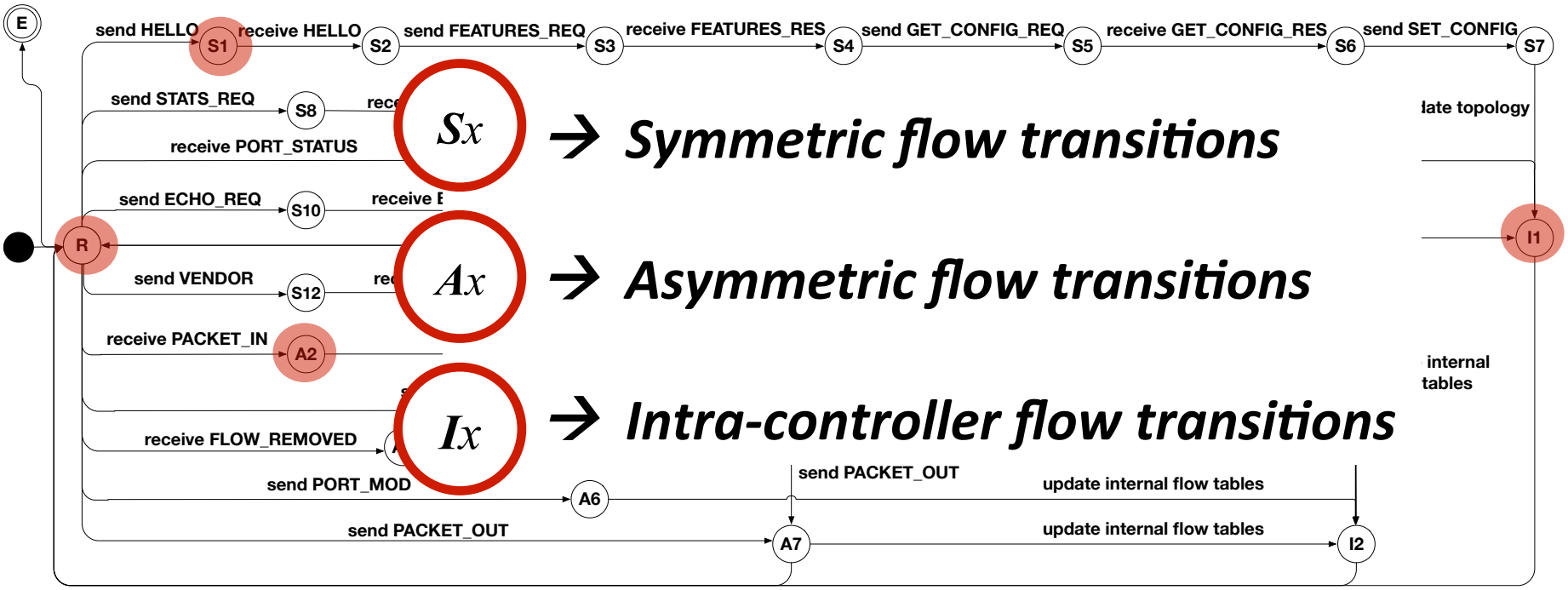
4. **Notify** the result

Blackbox Fuzzing

- To more **efficiently** and **systematically** randomize control flows (i.e., OpenFlow messages)
- Define **three types** of control flow operations
  - Symmetric control flow
  - Asymmetric control flow
  - Intra-controller control flow



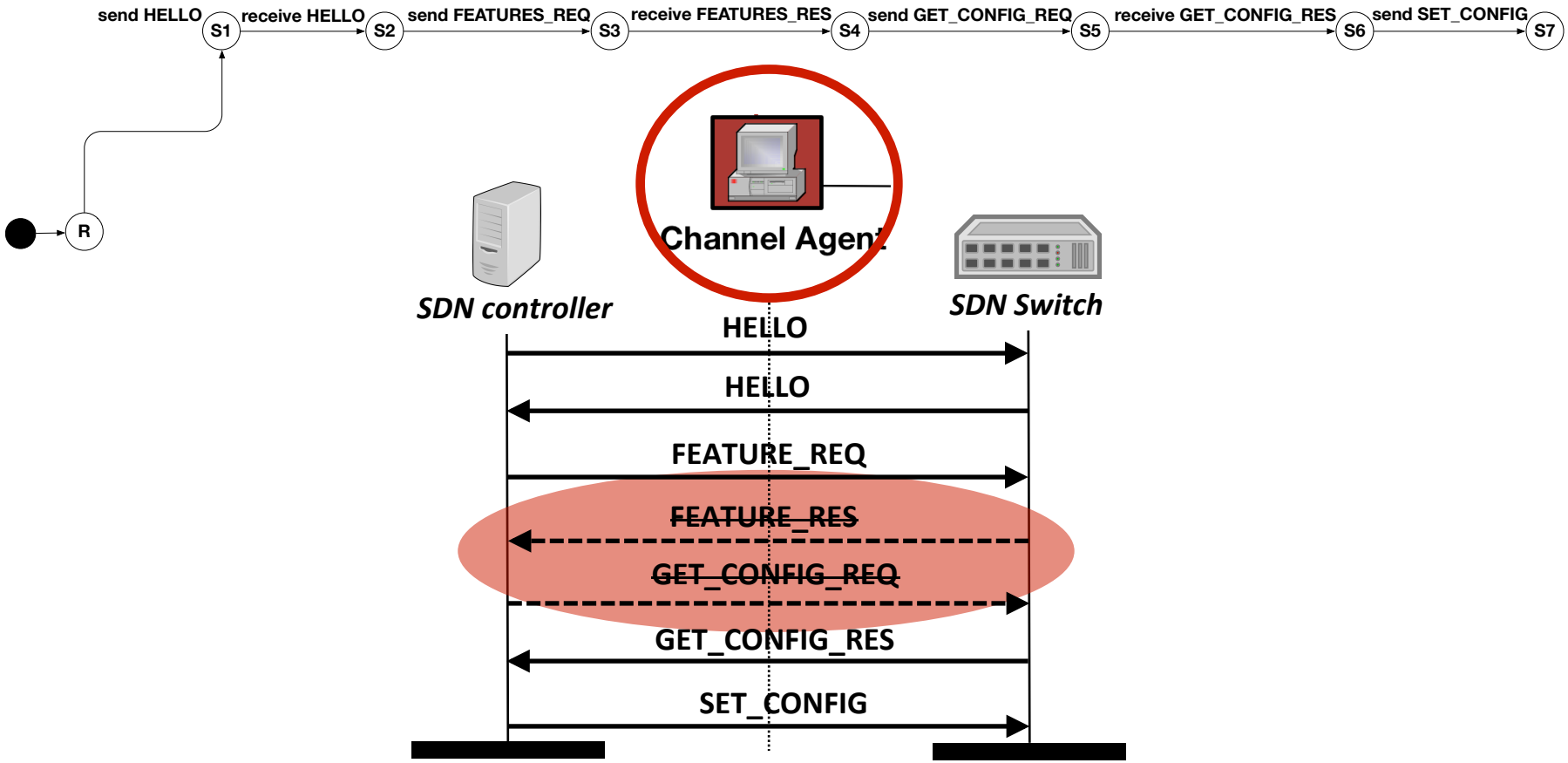
Operational State Diagram



1. Inferring current state
2. Manipulating the control flow sequence or input values

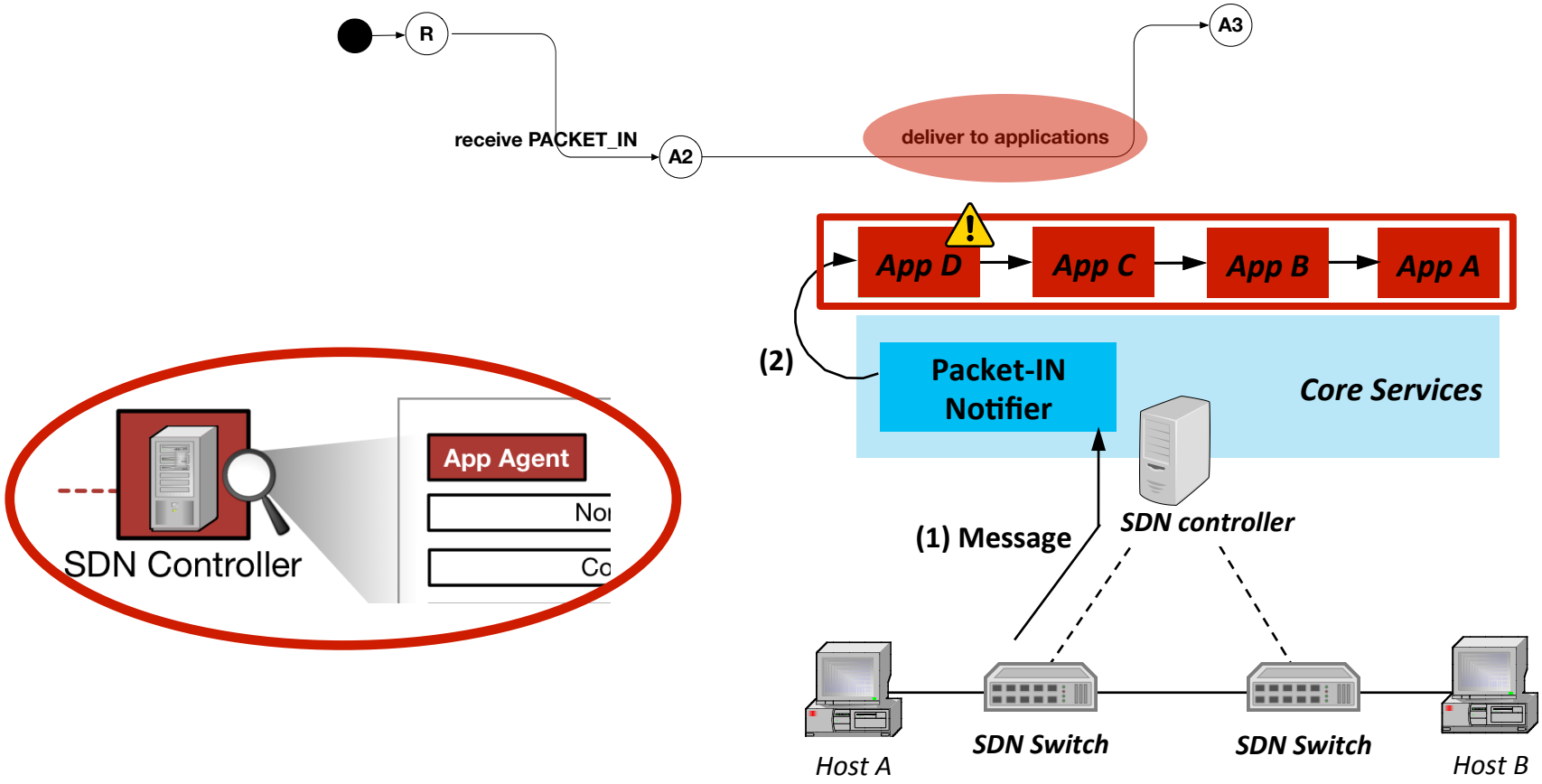
Randomizing Control Flow Sequence

- In the case of **symmetric** control flows



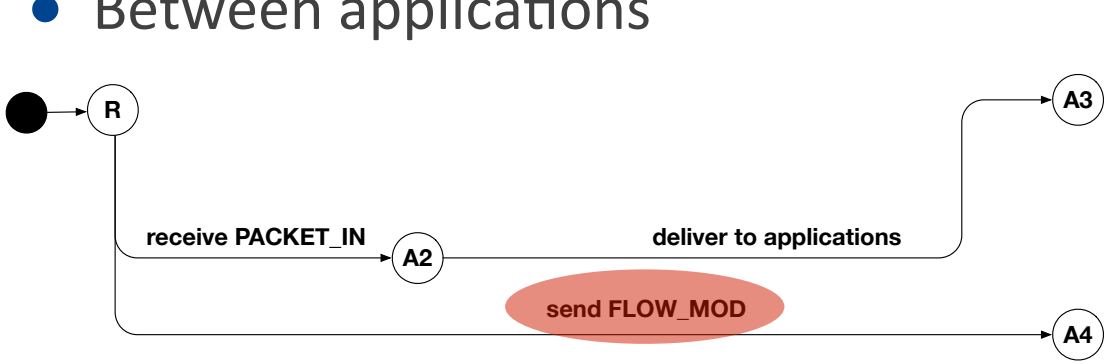
Randomizing Control Flow Sequence

- In the case of **asymmetric** control flows

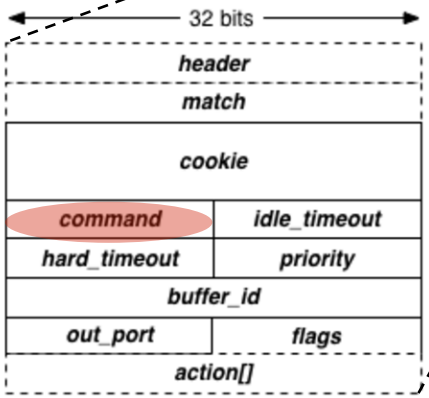


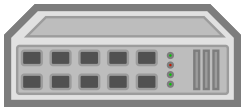
Randomizing Input Values

- ***Between an SDN controller and an SDN switch***
- Between applications



e.g.) ADD (0x0000)
→ ***(Undefined) (0x0005)***





Implementation

- Supports four different SDN controllers
  - 3 open source controllers (**ONOS, OpenDaylight, and Floodlight**)
  - 1 commercial controller
- OpenFlow v1.0 and v1.3 supported

< Supported application agents >

	ONOS				*OpenDaylight*				*Floodlight*				*A commercial one*
Version	1.2	1.3	1.4	1.5	Hydrogen	Helium	Lithium	Beryllium	0.91	1.0	1.1	1.2	2.3.0
Release Date	6/5/15	9/18/15	12/16/15	3/10/16	2/4/14	9/29/14	6/29/15	2/22/16	12/8/14	12/30/14	4/17/15	2/7/16	2016
Supported	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓

Evaluation

1. **Fuzz-testing Effectiveness**
   (Finding unknown attacks)

2. **Test Coverage and Flexibility**
   (Reproducing known attacks)

Use Case 1: Finding Unknown Attacks

- How to detect a vulnerability
  - Based on defined test criteria

- **Effectiveness** of fuzz testing
  - **7 unknown attack cases** found

1. A controller crash
2. An application crash
3. Internal-storage poisoning
4. A switch disconnection
5. Switch-performance downgrade
6. Error-packet generation
7. Inter-host communication disconnection

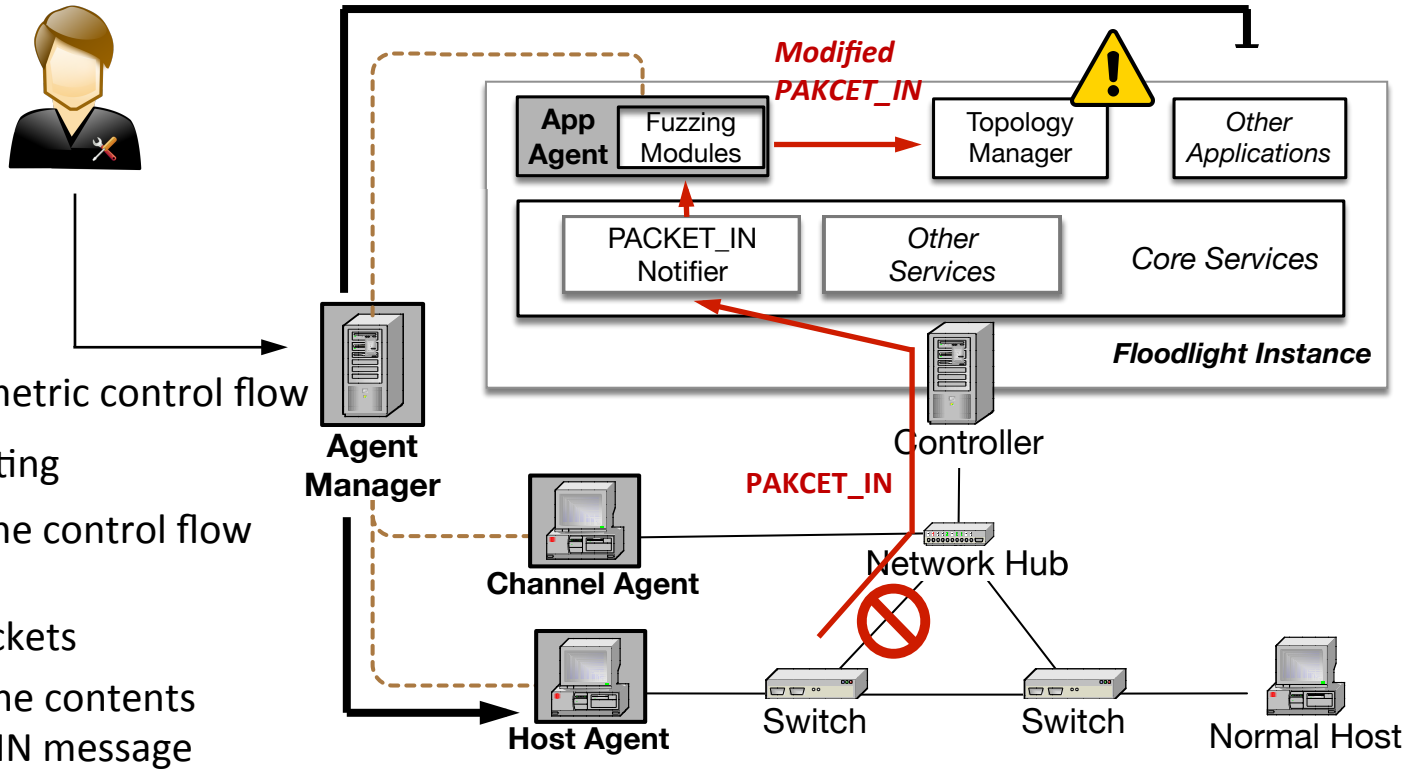
< Test Criteria >

Unknown Attack Name	***Flow***	***Target***
Sequence and Data-Forge	Asymmetric	Floodlight
Stats-Payload-Manipulation	Symmetric	Floodlight, OpenDaylight
Echo-Reply-Payload-Manipulation	Symmetric	OpenDaylight
Service-Unregistration	Intro-controller	OpenDaylight
Flow-Rule-Obstruction	Intro-controller	ONOS
Host-Tracking-Neutralization	Intro-controller	ONOS
Link-Discovery-Neutralization	Intro-controller	Floodlight

< Unknown attack classification >

Use Case 1: Finding Unknown Attacks

- Sequence and Data-Forge Attack
  - Target: asymmetric control flow and Floodlight v1.2



1. **Select** Asymmetric control flow
2. **Start** fuzz testing
3. **Randomize** the control flow sequence first
4. **Generate** packets
5. **Randomize** the contents of the PACKET_IN message

Use Case 1: Finding Unknown Attacks

- Results of the Sequence and Data-Forge attack experiment (Floodlight v1.2)

Before

```
[appagent] Packet-In listener as follows:
[appagent] 1 [linkdiscovery]  application
[appagent] 2 [topology]  application
[appagent] 3 [devicemanager]  application
[appagent] 4 [loadbalancer]  application
[appagent] 5 [firewall]  application
[appagent] 6 [forwarding]  application
[appagent] 7 [appagent]  application
```

After

```
[appagent] Packet-In listener as follows:
[appagent] 1 [appagent]  application
[appagent] 2 [topology]  application
[appagent] 3 [devicemanager]  application
[appagent] 4 [loadbalancer]  application
[appagent] 5 [firewall]  application
[appagent] 6 [forwarding]  application
[appagent] 7 [linkdiscovery]  application
```

1. A controller crash
2. An application crash
3. Internal-storage poisoning
4. ***A switch disconnection***
5. Switch-performance downgrade
6. Inter-host communication disconnection
7. Error-packet generation

```
rException: null
lightcontroller.topology.TopologyManager.processPacketInMessage(
lightcontroller.topology.TopologyManager.receive(TopologyManager
tification.main] Switch 00:0a:f0:92:1c:21:3d:c0 disconnected.
nelHandler:New I/O server worker #2-1] [00:0a:f0:92:1c:21:3d:c0
```

< Test Criteria >

Use Case 2: Reproducing Known Attacks [1]

Flow Type	*Attack Code*	*Attack Name*	*Controller*		
			ONOS	OpenDaylight	Floodlight
Symmetric Flows	SF-1	Switch Table Flooding	X	X	O
	SF-2	Switch Identification Spoofing	X	O	O
	SF-3	Malformed Control Message	X	O	O
	SF-4	Control Message Manipulation	O	O	O
Asymmetric Flows	AF-1	Control Message Drop	O	O	O
	AF-2	Control Message Infinite Loop	O	O	O
	AF-3	PACKET_IN Flooding	O	O	O
	AF-4	Flow Rule Flooding	O	O	O
	AF-5	Flow Rule Modification	O	O	O
	AF-6	Switch Firmware Misuse	O	O	O
	AF-7	Flow Table Clearance	O	O	O
	AF-8	Eavesdrop	O	O	O
	AF-9	Man-In-The-Middle	O	O	O
Intra-controller Flows	CF-1	Internal Storage Misuse	O	O	O
	CF-2	Application Eviction	O	O	N/A
	CF-3	Event Listener Unsubscription	N/A	O	O
Non Flow Operations	NF-1	System Command Execution	O	X	O
	NF-2	Memory Exhaustion	X	O	O
	NF-3	CPU Exhaustion	X	O	O
	NF-4	System Variable Manipulation	O	O	O

O: Successful
X: Unsuccessful
N/A: Not available

[1] http://sdnsecurity.org/vulnerability/AttackList.html

Use Case 2: Reproducing Known Attacks

- Flexibility of DELTA
  - **3 open source controllers and 1 commercial controller**
  - For example: Application Eviction Attack

Conclusion

- We categorize known vulnerabilities that can mislead network operations into **three control flow types** and non flow operations
- We propose **an automated security assessment framework** for SDN capable of reproducing those vulnerabilities
- We incorporate blackbox fuzzing techniques into our framework **to detect new unknown attack scenarios**
- We show the **flexibility** of system design by evaluating it against three popular open-source SDN controllers and the commercial controller
- DELTA is now available as on OFFICIAL ONF Sponsored Open Source Project **https://github.com/OpenNetworkingFoundation/delta**

Q&A

Appendix: Performance

Control Flow Type	*Average Running Time*
Asymmetric Control Flow	82.5 sec
Symmetric Control Flow	80.4 sec
Intra-controller Control Flow	75.2 sec

Finding unknown attack microbenchmark

Reproducing known attacks microbenchmark

Attack Name	*Controller*		
	ONOS	**ODL**	**Floodlight**
Switch Table Flooding	-	-	5400 sec
Switch Identification Spoofing	16.09 sec	16.34 sec	15.96 sec
Malformed Control Message	21.50 sec	12.33 sec	11.09 sec
Control Message Manipulation	28.10 sec	19.27 sec	18.60 sec
Control Message Drop	12.55 sec	8.47 sec	3.13 sec
Control Message Infinite Loop	3.38 sec	8.12 sec	3.21 sec
PACKET_IN Flooding	12.59 sec	17.79 sec	11.96 sec
Flow Rule Flooding	43.65 sec	23.28 sec	43.20 sec
Flow Rule Modification	40.43 sec	40.24 sec	20.35 sec
Switch Firmware Misuse	20.52 sec	20.25 sec	20.20 sec
Flow Table Clearance	20.60 sec	20.32 sec	20.17 sec
Eavesdrop	33.62 sec	33.18 sec	33.14 sec
Man-In-The-Middle	17.80 sec	17.19 sec	7.88 sec
Internal Storage Misuse	2.60 sec	3.14 sec	2.14 sec
Application Eviction	22.57 sec	13.33 sec	N/A
Event Listener Unsubscription	N/A	13.22 sec	13.11 sec
System Comma…			0.127 sec
Memory Exhaus…			23.16 sec
CPU Exhaustion	23.43 s…	23.36 sec	23.35 sec
System Variable Manipulation	3.39 sec	4.86 sec	3.17 sec
Total	**346.38 sec**	**317.98 sec**	**274.84 sec**

About 5 minutes