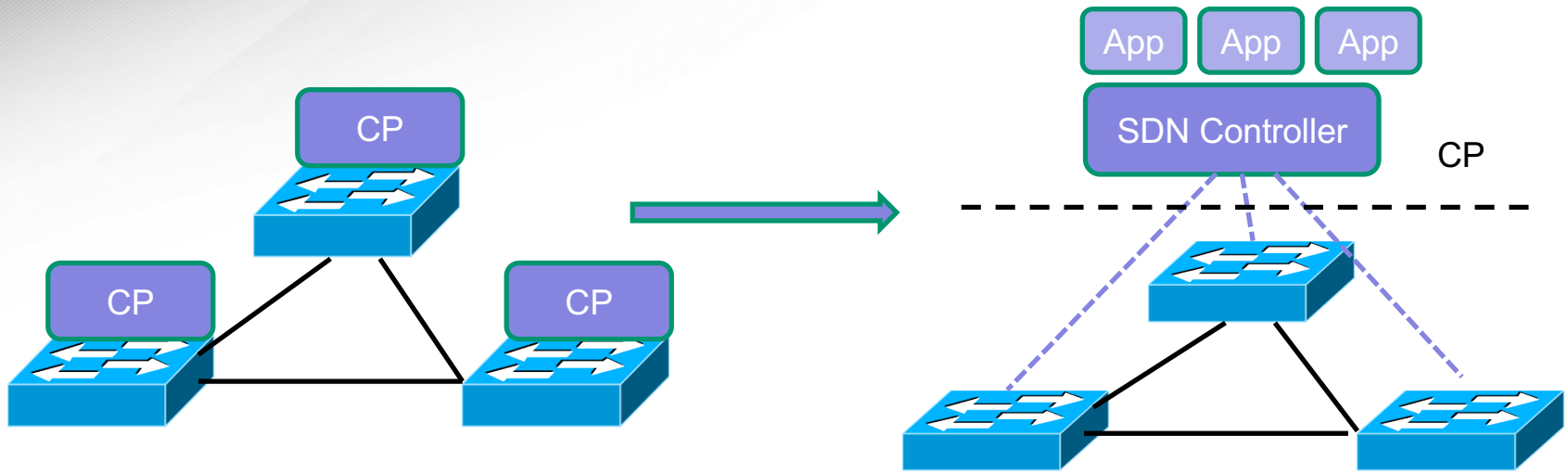
COMPUTER SCIENCE & ENGINEERING
TEXAS A&M UNIVERSITY

Attacking the Brain: Races in the SDN Control Plane

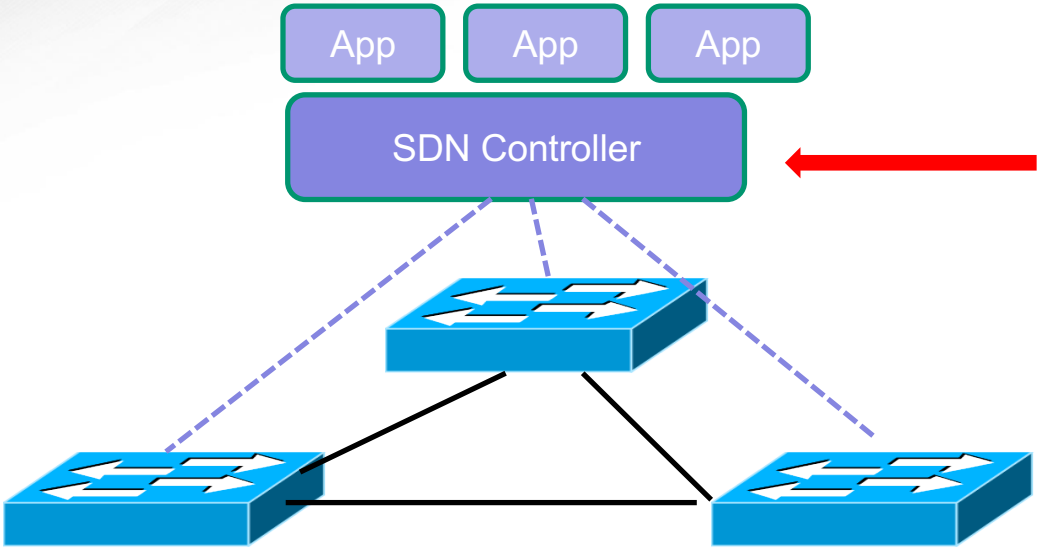
Lei Xu, Jeff Huang, Sungmin Hong, Jialong Zhang, Guofei Gu
*Secure Communication and Computer Systems Lab
Texas A&M University

SDN Overview



- Software-Defined Networking (SDN) is a novel programmable network paradigm that separates the control logic from the data plane.

SDN Control Plane – New Achilles' Heel



Control Plane Saturation (CCS'13, DSN'15)

Topology Poisoning (NDSS'15)

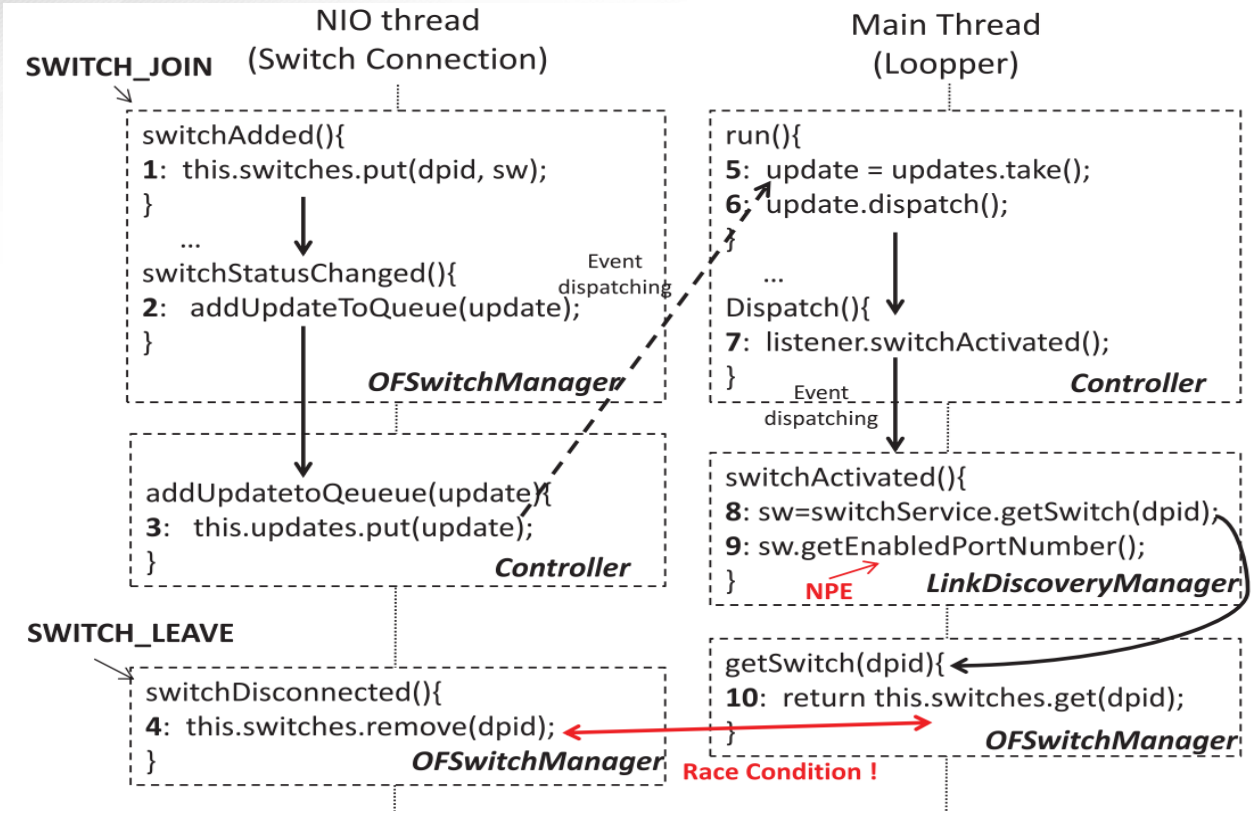
State Manipulation (In this paper!)

Harmful Race Conditions in the Brain

The network states maintained in the SDN control plane is subject to harmful race conditions.

- Non-adversarial causality: asynchronous network events and non-determinist schedules.
- Adversarial causality: an attacker can intentionally inject *right* network events to exploit vulnerabilities -- *State Manipulation Attacks*

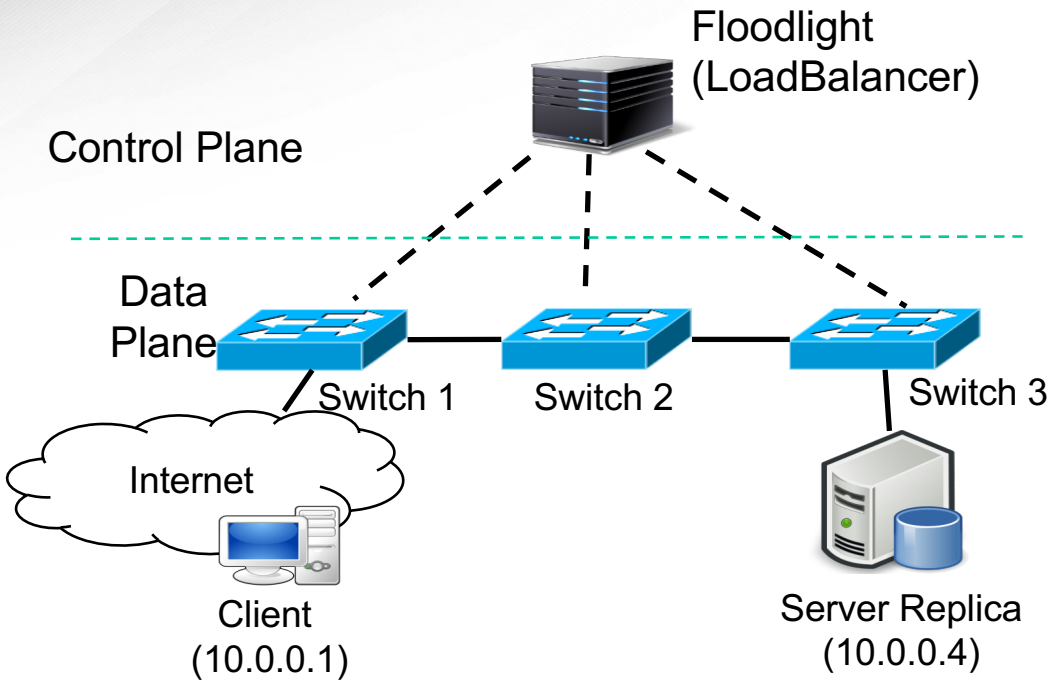
How Does It Look like?



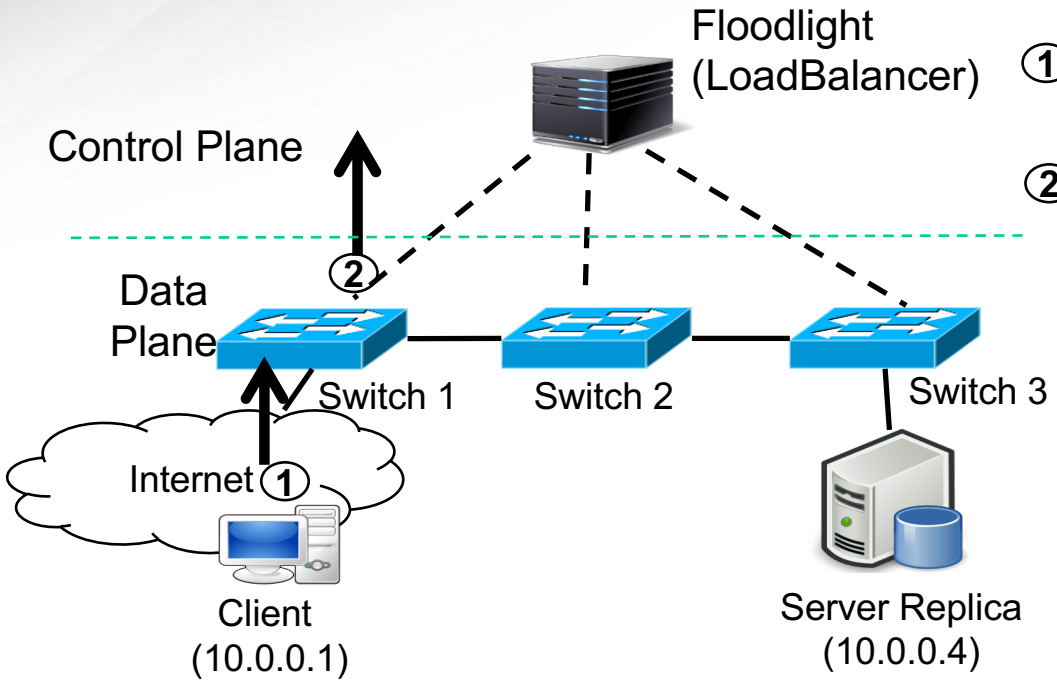
What Can It Cause?

- System Crash
- Connection Disruption
- Service Disruption
- Service Chain Interference
- Privacy Leakage
- …

A Real Vulnerability in LoadBalancer



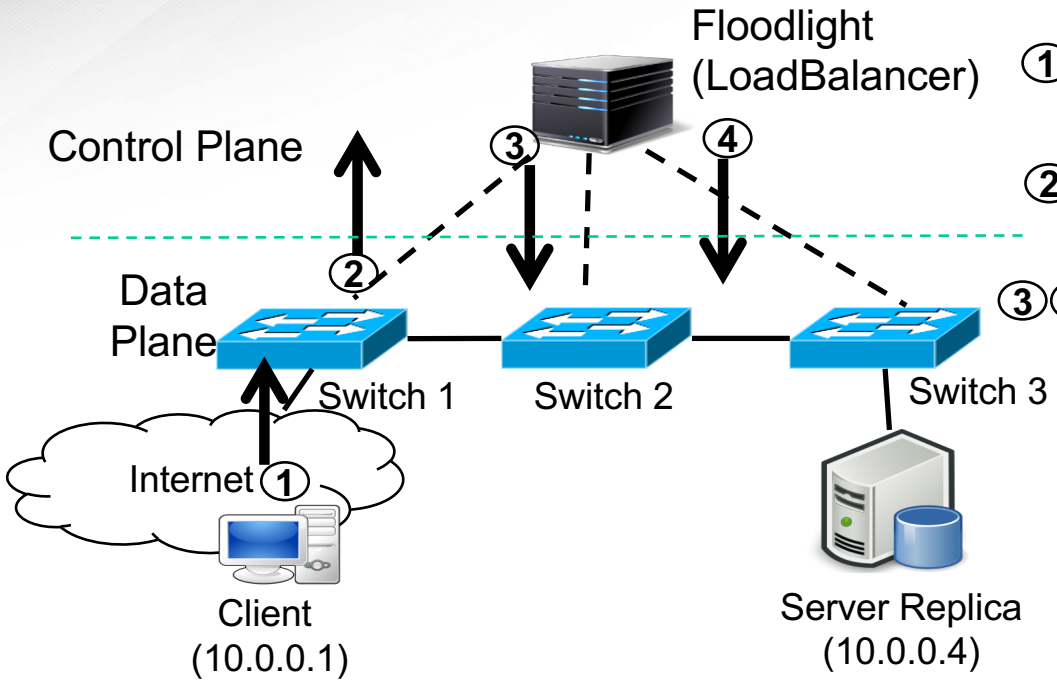
A Real Vulnerability in LoadBalancer



① **Client send requests to public service**

② **Switch issues Packet-In message to SDN controller**

A Real Vulnerability in LoadBalancer

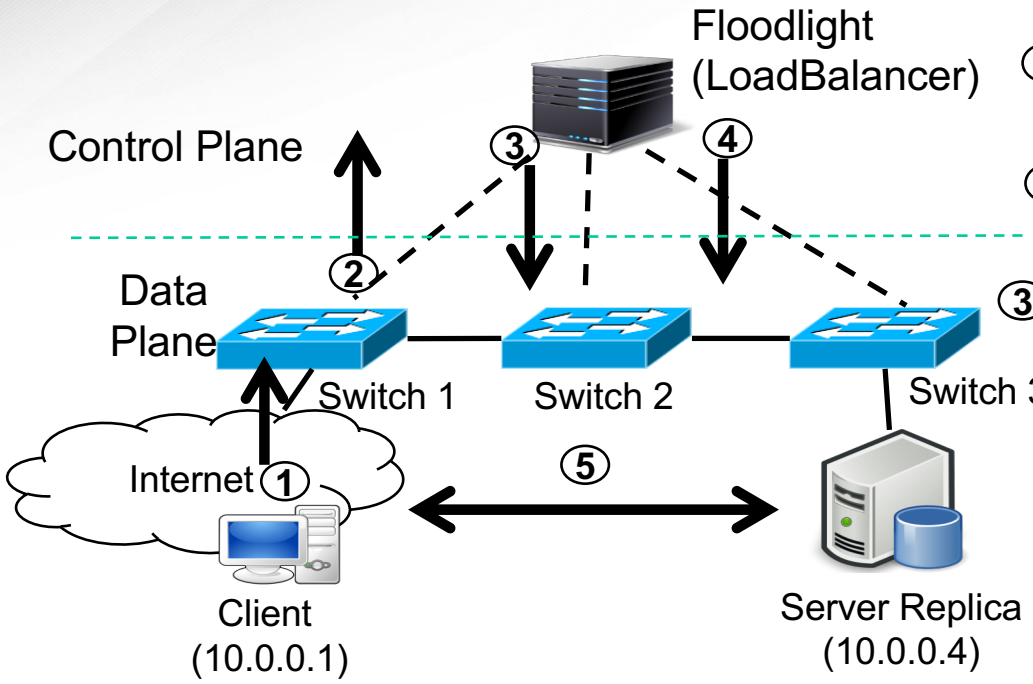


① **Client send requests to public service**

② **Switch issues Packet-In message to SDN controller**

③④ **SDN controller select Server replica and instruct switch to install flow rules to translate traffic between client and server replica**

A Real Vulnerability in LoadBalancer



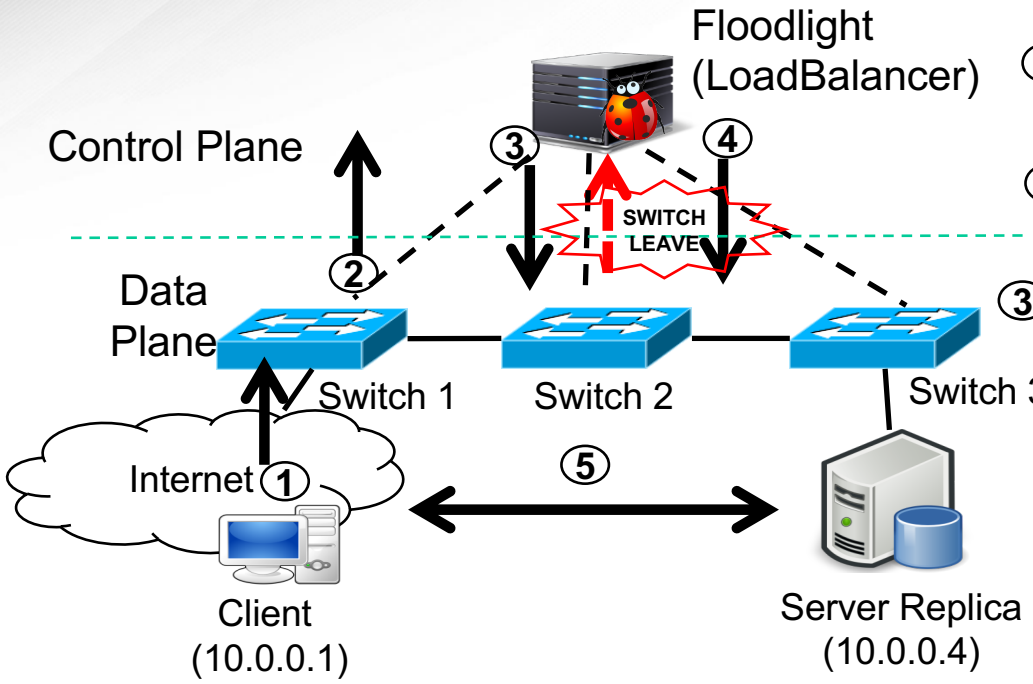
① **Client send requests to public service**

② **Switch issues Packet-In message to SDN controller**

③④ **SDN controller select Server replica and instruct switch to install flow rules to translate traffic between client and server replica**

⑤ **The communication between Client and public service is successfully set up**

A Real Vulnerability in LoadBalancer



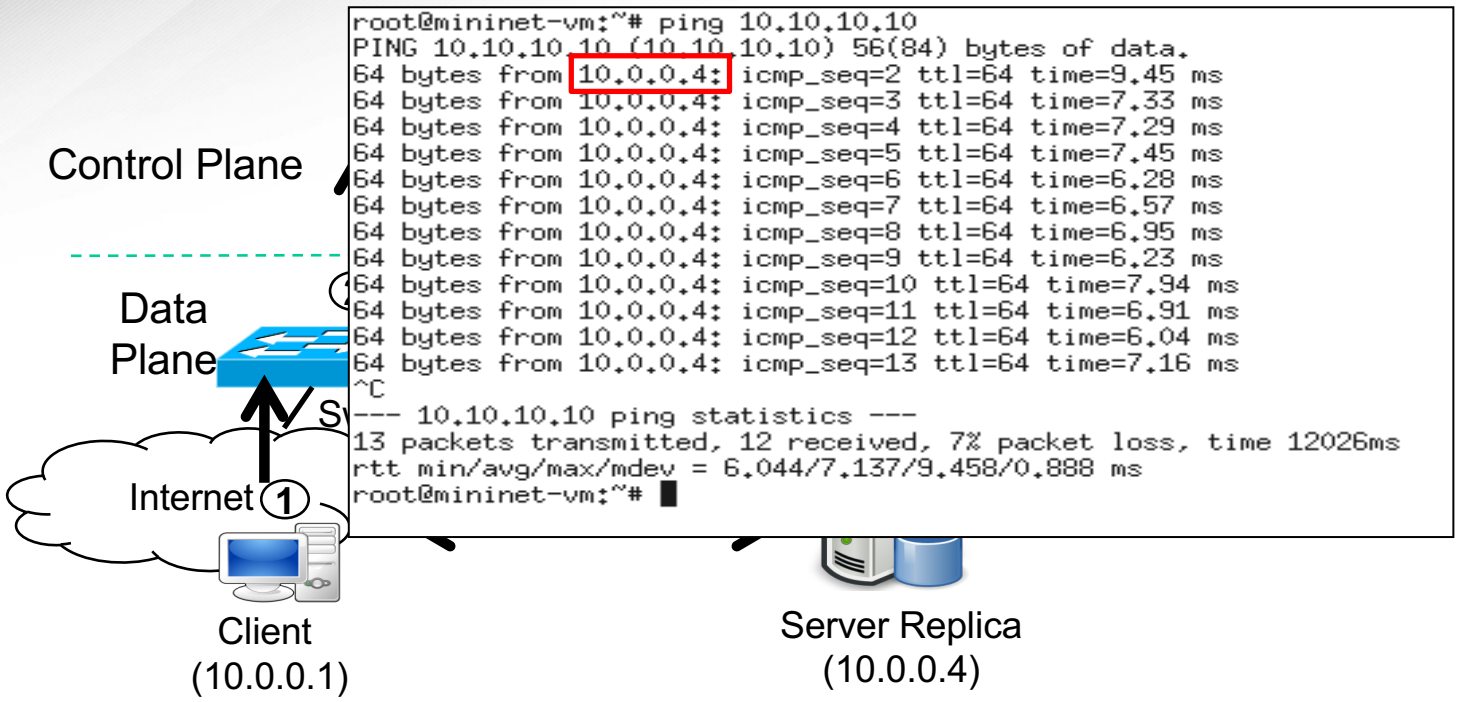
① **Client send requests to public service (10.10.10.10)**

② **Switch issues Packet-In message to SDN controller**

③④ **SDN controller select Server replica and instruct switch to install flow rules to translate traffic between client and server replica**

⑤ **The communication between Client and public service is successfully set up**

A Real Vulnerability in LoadBalancer



Research Questions

- How to detect harmful race conditions in the SDN control plane?
- How to exploit harmful race conditions by an external attacker?

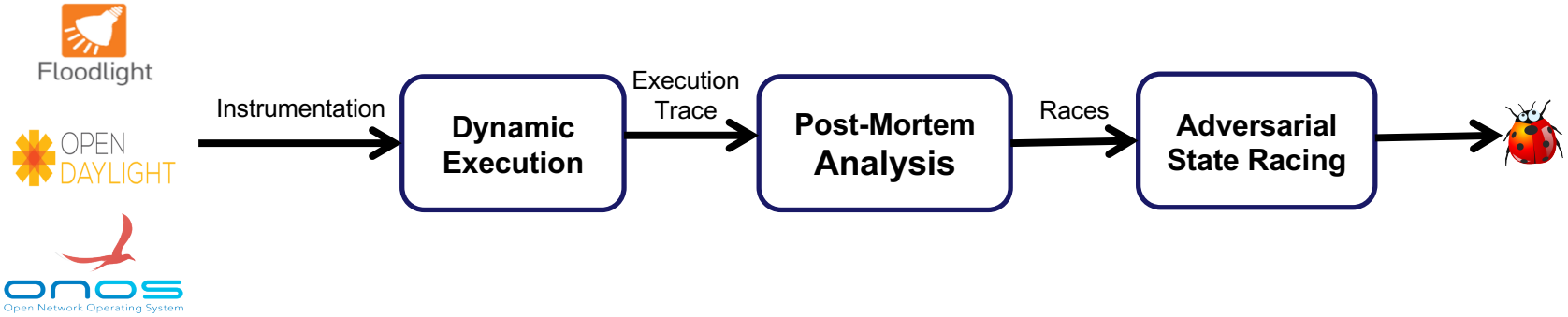
Research Questions

- How f conditions in the S
- How conditions by an e

Our Solution: ConGuard

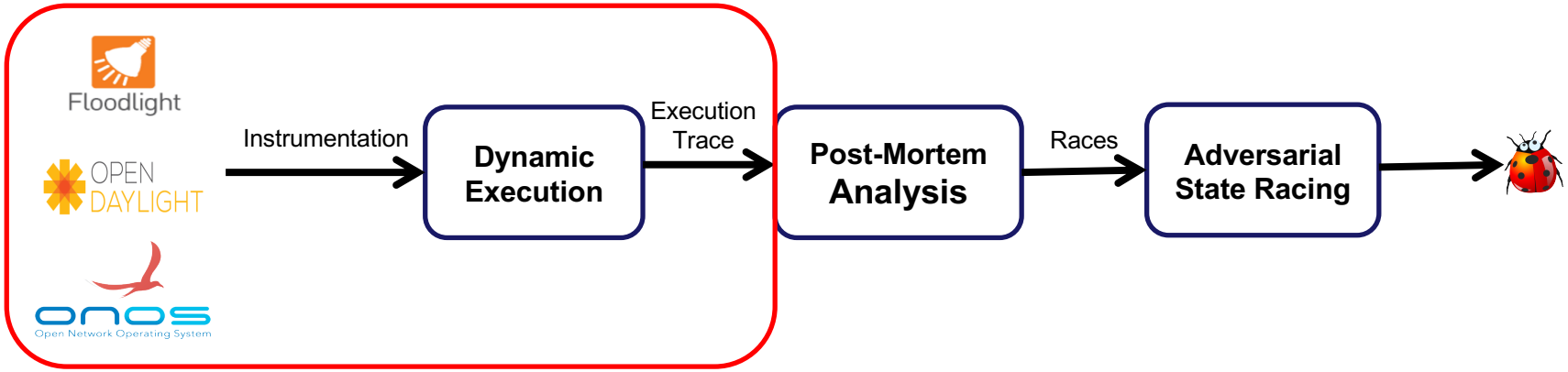
Detection of Harmful Race Conditions

- Key Insight: Harmful race conditions are rooted by two race operations upon shared network states that are not commutative, i.e., mutating the scheduling order of them leads to a different state though the two operations can be well-synchronized (e.g., by using locks).



Execution Trace

- We dynamically log a sequence of critical operations to model the operations of SDN control plane from instrumented SDN control plane.

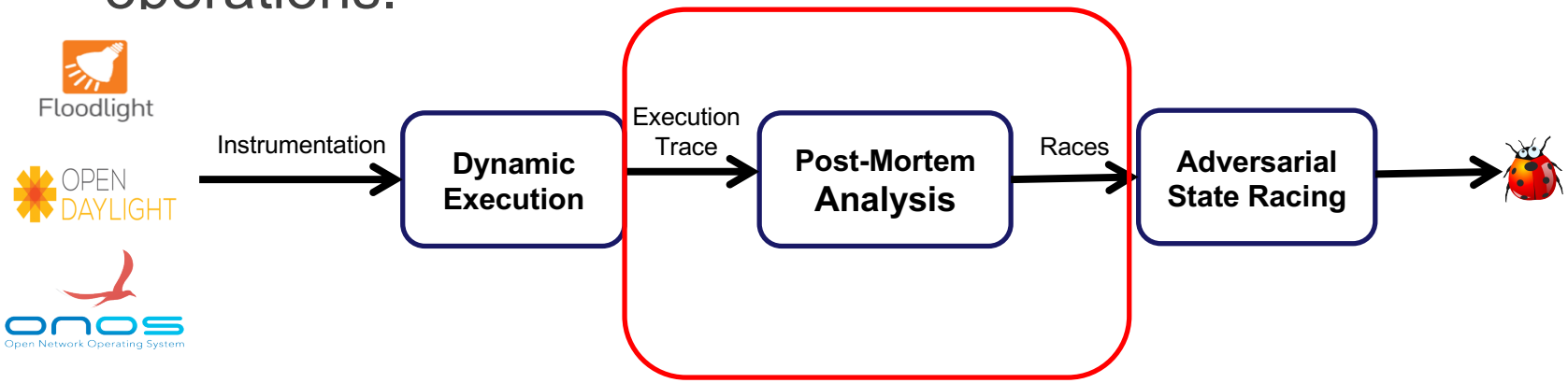


Critical Operations in Execution Trace

- **read(T,V)** : reads variable V in thread T
- **write(T,V)** : writes variable V in thread T
- **init(A)** : initializes application A
- **terminate(A)** : terminates application A
- **dispatch(E)** : dispatches event E
- **receive(H,E)** : receives event E by event handler H
- **schedule(TA)** : instantiates a singleton task TA
- **end(TA)** : terminates a singleton task TA

Post-Mortem Analysis

- We develop happens-before relations to model concurrency semantics of the SDN control plane.
- We utilize graph-based approach to locate race operations.



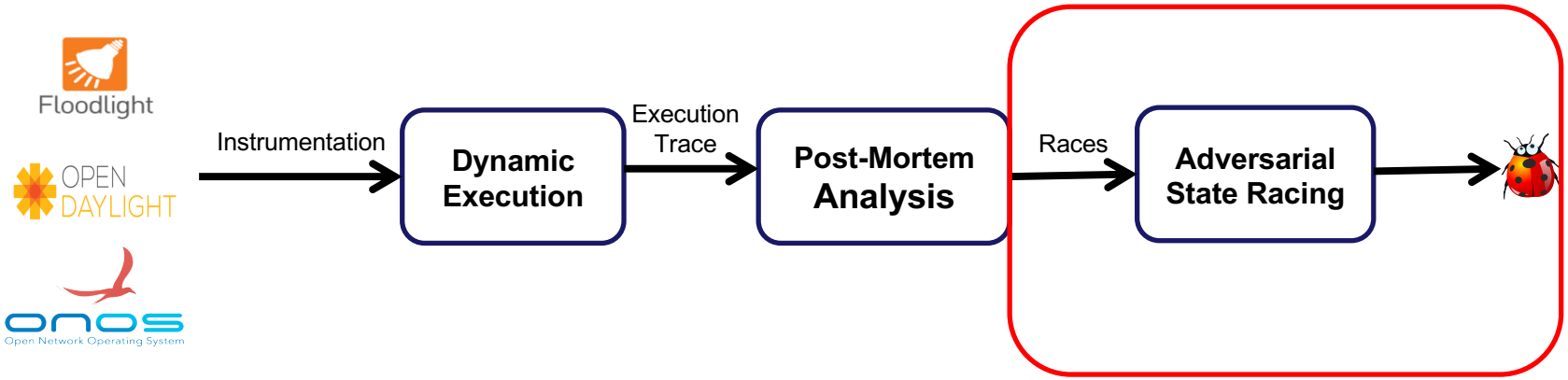
Race Detection

- Pre-processing
  - Prune operations on thread-local or immutable variable
  - Duplicated operations removal
- Graph-based Race Detection Algorithm
  - Use DAG to model operations

    operations ➡ nodes happens-before ➡ edges
  - Reachability Check in the graph ➡ Race Operations

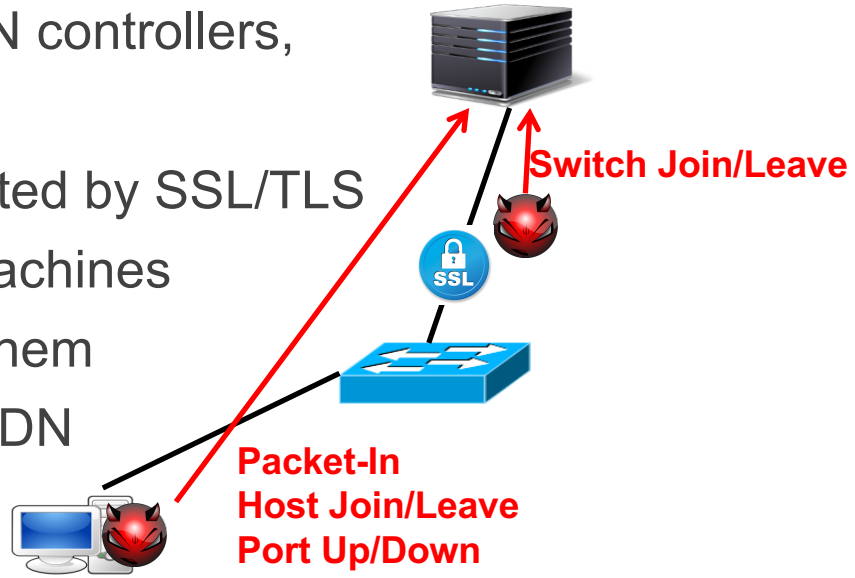
Adversarial State Racing

- We instrument control logic to force an erroneous execution order, e.g., the state update executes before the state references.

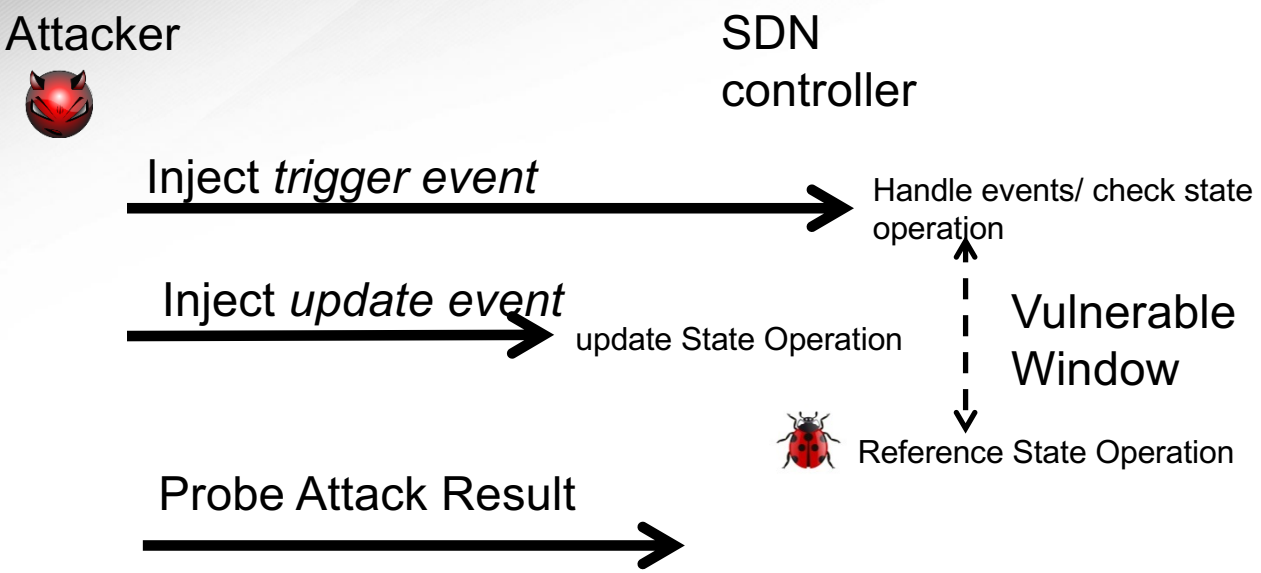


Exploitation of Harmful Race Conditions

- Thread Model
  - No need of compromised SDN controllers, apps, switches and protocol
  - Control channel is well protected by SSL/TLS
  - Compromised hosts/virtual machines
  - Inject 7 network events, 2 of them need in-band deployment of SDN



Exploitation of Harmful Race Conditions



- Attack Strategies
  - Repeat ordered event sequences ***<trigger event, update event>***
  - Feedback Probing
  - Exploit larger vulnerable windows

Evaluation

- Implementation of ConGuard
  - On Floodlight, ONOS and OpenDaylight controllers with 34 apps.
  - Input Generator: Mininet & Rest API scripts
  - Instrumentation & Static Analysis: ASM framework

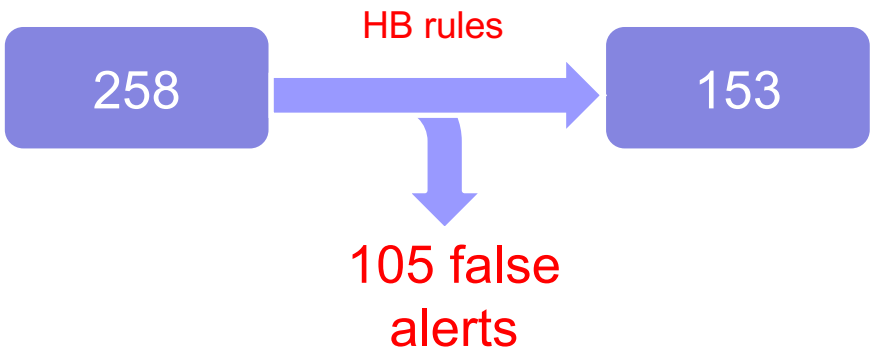
- Totally pinpoint 15 unknown harmful race conditions

Race Detection Result

Controller	Version	Preprocessing Reduction	Race Operations
Floodlight	1.1	96.6%	153
Floodlight	1.2	87.2%	184
OpenDaylight	0.1.7	92.1%	221
ONOS	1.2	98.1%	13

Effectiveness of HB rules

- In Floodlight Controller



Impact Analysis

- All located 15 harmful race conditions with 4 harmful impacts:
  - System Crash (4 of them)
  - Connection Disruption (7 of them)
  - Service Disruption (13 of them)
  - Service Chain Interference (7 of them)

Correlation of External Events

Controller	Application	Bug#	Correlated Attack Event Pairs <trigger event, update event>
Flood-light	Link Discovery Manager	1*	<SWITCH_JOIN, SWITCH_LEAVE>, <PORT_UP, SWITCH_LEAVE>
		2*	<SWITCH_JOIN, SWITCH_LEAVE>, <PORT_UP, SWITCH_LEAVE>
		3*	<SWITCH_JOIN, SWITCH_LEAVE>, <PORT_UP, SWITCH_LEAVE>
	DHCPServer	4*	<SWITCH_JOIN, SWITCH_LEAVE>, <PORT_UP, SWITCH_LEAVE>
	Load Balancer	5*	<OFP_PACKET_IN, SWITCH_LEAVE>
		6*	<OFP_PACKET_IN, SWITCH_LEAVE>
		7†	<OFP_PACKET_IN, REST_REQUEST>
		8†	<OFP_PACKET_IN, REST_REQUEST>
		9†	<OFP_PACKET_IN, REST_REQUEST>
	Statistics	10†	<REST_REQUEST, SWITCH_LEAVE>
ONOS	SegmentRouting	11	<OFP_PACKET_IN, HOST_LEAVE>
	DHCPRelay	12	<OFP_PACKET_IN, HOST_LEAVE>
OpenDay-light	Host Tracker	13†	<REST_REQUEST, HOST_LEAVE>
		14	<HOST_JOIN, HOST_LEAVE>
	Web UI	15†*	<REST_REQUEST, SWITCH_LEAVE>

* in-band

† REST API

Remote Exploitations

Average trials to get a successful exploitation

Bug #	Attack Case	Trials (average)
1	(SWITCH_JOIN,SWITCH_LEAVE)	10.6
2	(SWITCH_JOIN,SWITCH_LEAVE)	78.4
3	(SWITCH_JOIN,SWITCH_LEAVE)	120
4	(SWITCH_JOIN,SWITCH_LEAVE)	10
5	(OFP_PACKET_IN,SWITCH_LEAVE)	67.6
6	(OFP_PACKET_IN,SWITCH_LEAVE)	106.8
11	(OFP_PACKET_IN,HOST_LEAVE)	-
12	(OPP_PACKET_IN,HOST_LEAVE)	1
14	(HOST_LEAVE,HOST_JOIN)	-

Potential Defense Schemes

- Safety Check
  - Ensure consistent state at the reference location
- Deterministic Execution Runtime
  - Guarantee the deterministic execution of state operations
- Sanitizing External Network Events
  - Anomaly detection system to sanitize suspicious state update events

Conclusion

- We report State Manipulation Attacks that target the SDN control plane.
- We design ConGuard framework to pinpoint and exploit harmful race conditions in the SDN control plane.
- We present an extensive evaluation of ConGuard that uncovered 15 unknown vulnerabilities (we have helped developers patch most of them already)

Thanks for attention!
Q&A