

# Automated Discovery of Cross-Plane Event-Based Vulnerabilities in Software-Defined Networking

Benjamin E. Ujcich<sup>1</sup>, Samuel Jero<sup>2</sup>, Richard Skowyra<sup>2</sup>, Steven R. Gomez<sup>2</sup>, Adam Bates<sup>1</sup>, William H. Sanders<sup>1</sup>, and Hamed Okhravi<sup>2</sup>

<sup>1</sup> University of Illinois at Urbana-Champaign, <sup>2</sup> MIT Lincoln Laboratory



2020 Network and Distributed System  
Security Symposium (NDSS)

February 25, 2020  
San Diego, CA, USA



# SDN is Everywhere!



# Network “Appification”

The screenshot shows the HPE SDN App Store interface. The top navigation bar includes 'Hewlett Packard Enterprise', 'Documents', 'Community', 'Contact us', and 'Sign in'. The main navigation bar features 'SDN App Store', 'Categories', 'Dashboard', and 'Develop'. A search icon is visible on the right. The main content area is titled 'Security' and includes a description: 'Apps that extend, or enforce security across the network. If you'd like to talk to us about these solutions, add the products you're interested in to your shopping cart then look for the Contact HP button to submit to a HP sales professional. HP will then reach out to you outside of the store; or put you in contact with an appropriate HP channel partner, or the ISV.' Below the description are filter dropdowns for 'All products', 'All versions', and 'All companies', along with a 'Clear filters' button. The 'Sort by' section is set to 'Newest'. A list of applications is displayed, including 'HPE Network Protector: Free Trial' (Hewlett Packard Enterprise), 'FortiGate Connector for HP VAN SDN Controller' (Fortinet), and 'Active Honeypot' (Guardicore).

**Introducing HPE SDN App Store**  
Select from a range of SDN Applications that allow you to program your network to align with business needs. Deploy directly to the enterprise ready HPE VAN SDN Controller.

### Get Started

- 1 Submit your apps**  
Find the resources you need to build a SDN applications with documentation and discussions from an active developer community.
- 2 Reach your audience**  
After the review process, your SDN application will be available to thousands of users on the industry's first SDN marketplace.



# Network “Appification”

Do apps work well together?

The screenshot shows the HPE SDN App Store interface. At the top, there's a navigation bar with 'SDN App Store', 'Categories', and 'Dashboard'. Below this, a 'Security' category page is displayed. It includes a description: 'Apps that extend, or enforce security across the network. If you'd like to talk to us about these solutions, add the products you're interested in to your shopping cart then look for the Contact HP button to submit to a HP sales professional. HP will then reach out to you outside of the store; or put you in contact with an appropriate HP channel partner, or the ISV.' Below the text are filter options: 'All products', 'All versions', 'All companies', and 'Clear filters'. A 'Sort by' section shows 'Newest' selected, with 'Downloads' and 'A-Z' as other options. A list of apps is shown, including 'HPE Network Protector: Free Trial', 'FortiGate Connector for HP VAN SDN Controller', and 'Active Honeypot'.

**Introducing HPE SDN App Store**  
Select from a range of SDN Applications that allow you to program your network to align with business needs. Deploy directly to the enterprise ready HPE VAN SDN Controller.

**Get Started**

- 1 Submit your apps**  
Find the resources you need to build a SDN applications with documentation and discussions from an active developer community.
- 2 Reach your audience**  
After the review process, your SDN application will be available to thousands of users on the industry's first SDN marketplace.

Sort by:	Newest	Downloads	A-Z
All products	All versions	All companies	Clear filters
HPE COMMUNITY	FREE	COMMUNITY	FREE
COMMUNITY	FREE	COMMUNITY	FREE
HPE Network Protector: Free Trial Hewlett Packard Enterprise		FortiGate Connector for HP VAN SDN Controller Fortinet	Active Honeypot Guardicore



# Network “Appification”

How can they be exploited?

The screenshot displays the HPE SDN App Store interface. At the top, there's a navigation bar with 'SDN App Store', 'Categories', and 'Dashboard'. Below this, a 'Security' category page is shown, featuring a description of security apps and a list of products. The products listed include 'HPE Network Protector: Free Trial', 'FortiGate Connector for HP VAN SDN Controller', and 'Active Honeypot'. A 'Get Started' section is overlaid on the left, providing two steps: '1 Submit your apps' and '2 Reach your audience'.

**Introducing HPE SDN App Store**  
Select from a range of SDN Applications that allow you to program your network to align with business needs. Deploy directly to the enterprise ready HPE VAN SDN Controller.

**Get Started**

- 1 Submit your apps**  
Find the resources you need to build a SDN applications with documentation and discussions from an active developer community.
- 2 Reach your audience**  
After the review process, your SDN application will be available to thousands of users on the industry's first SDN marketplace.

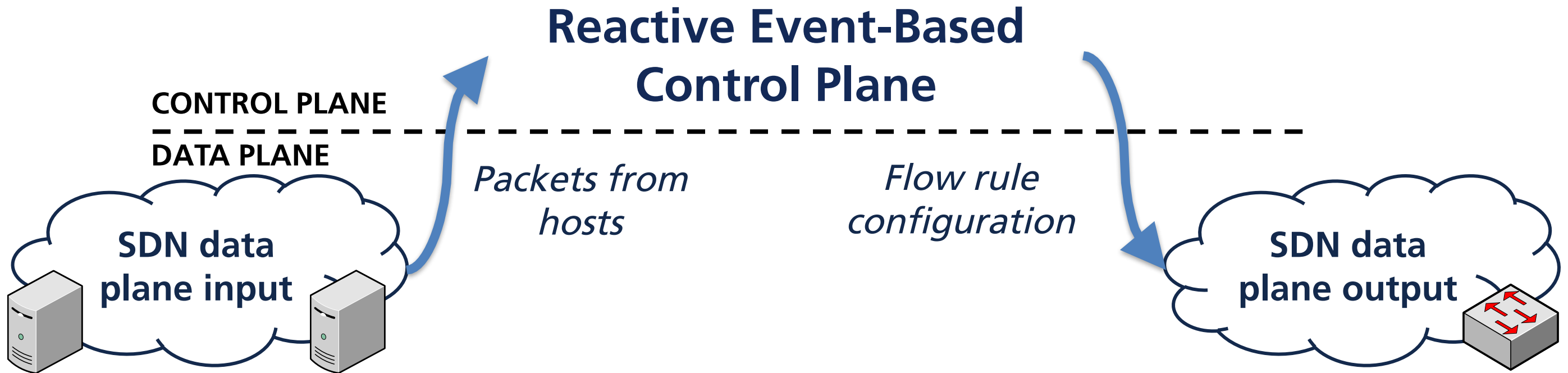
**Security**  
Apps that extend, or enforce security across the network. If you'd like to talk to us about these solutions, add the products you're interested in to your shopping cart then look for the Contact HP button to submit to a HP sales professional. HP will then reach out to you outside of the store; or put you in contact with an appropriate HP channel partner, or the ISV.

All products | All versions | All companies | Clear filters

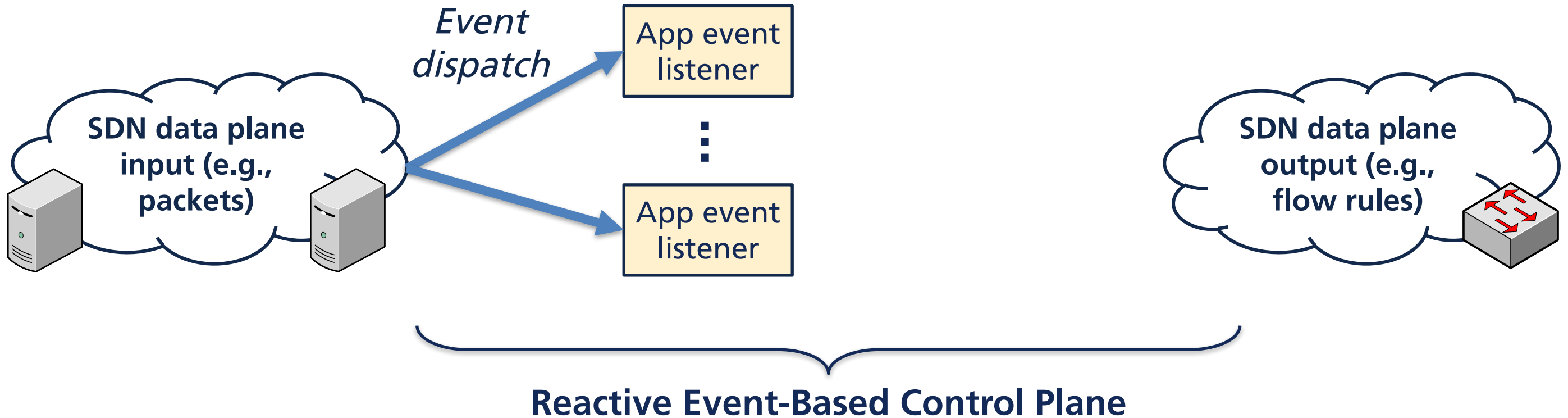
Sort by: **Newest** | Downloads | A-Z

Product Name	Price	Category
HPE Network Protector: Free Trial	FREE	COMMUNITY
FortiGate Connector for HP VAN SDN Controller	FREE	COMMUNITY
Active Honeypot	FREE	COMMUNITY

# Cross-Plane Vulnerabilities

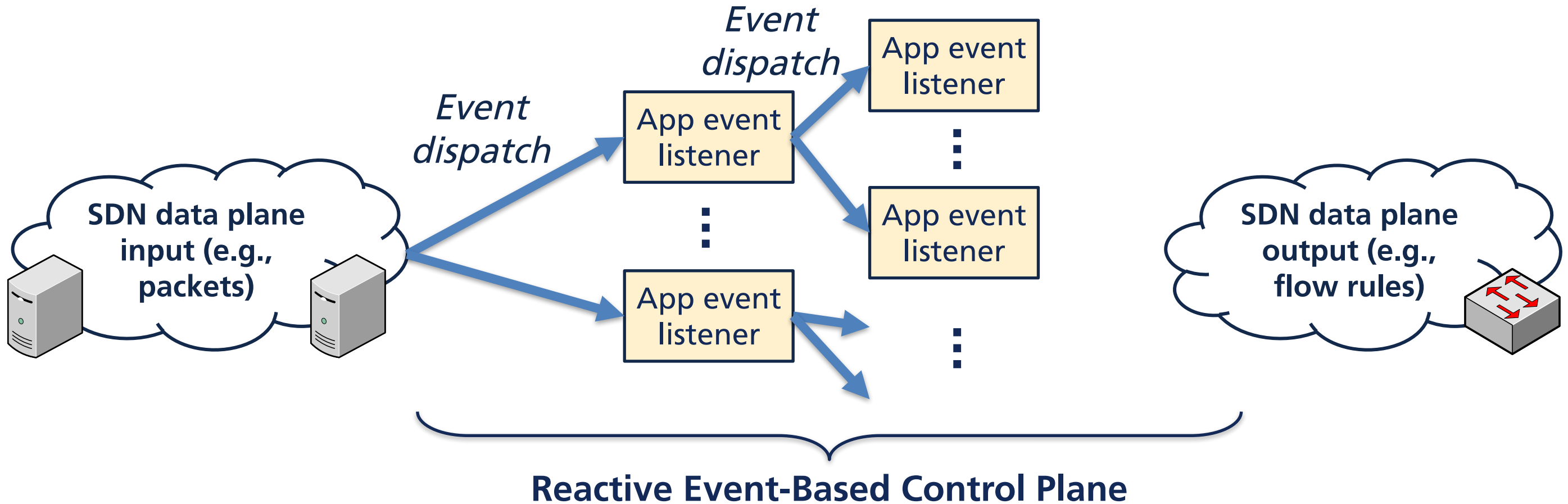


# Cross-Plane Vulnerabilities

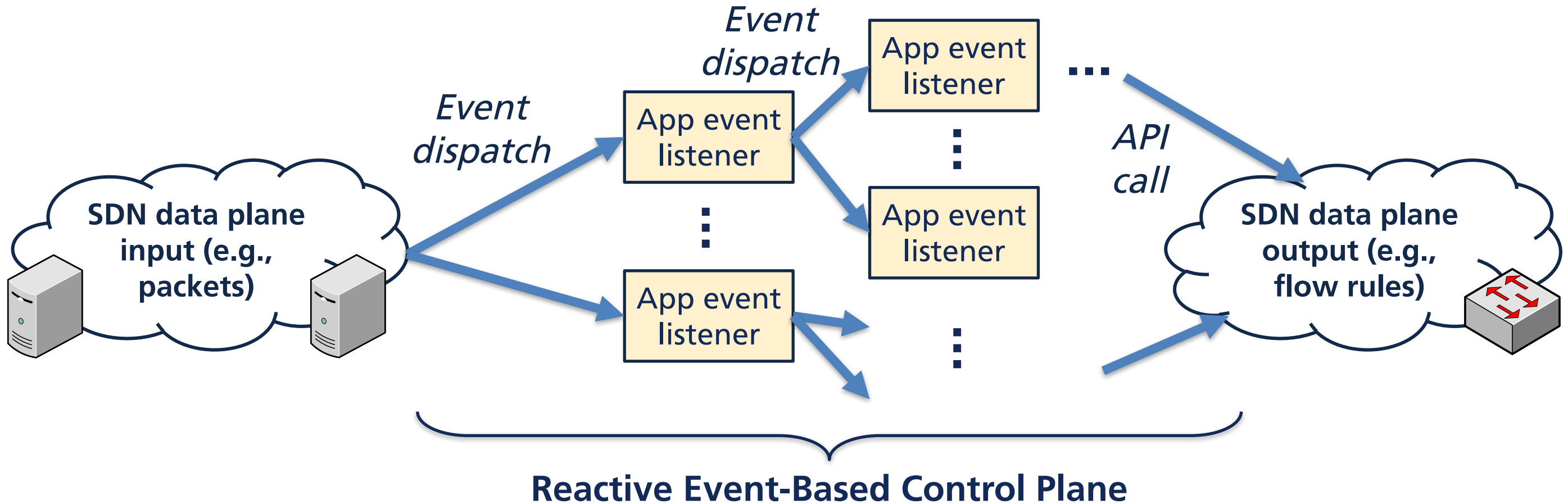




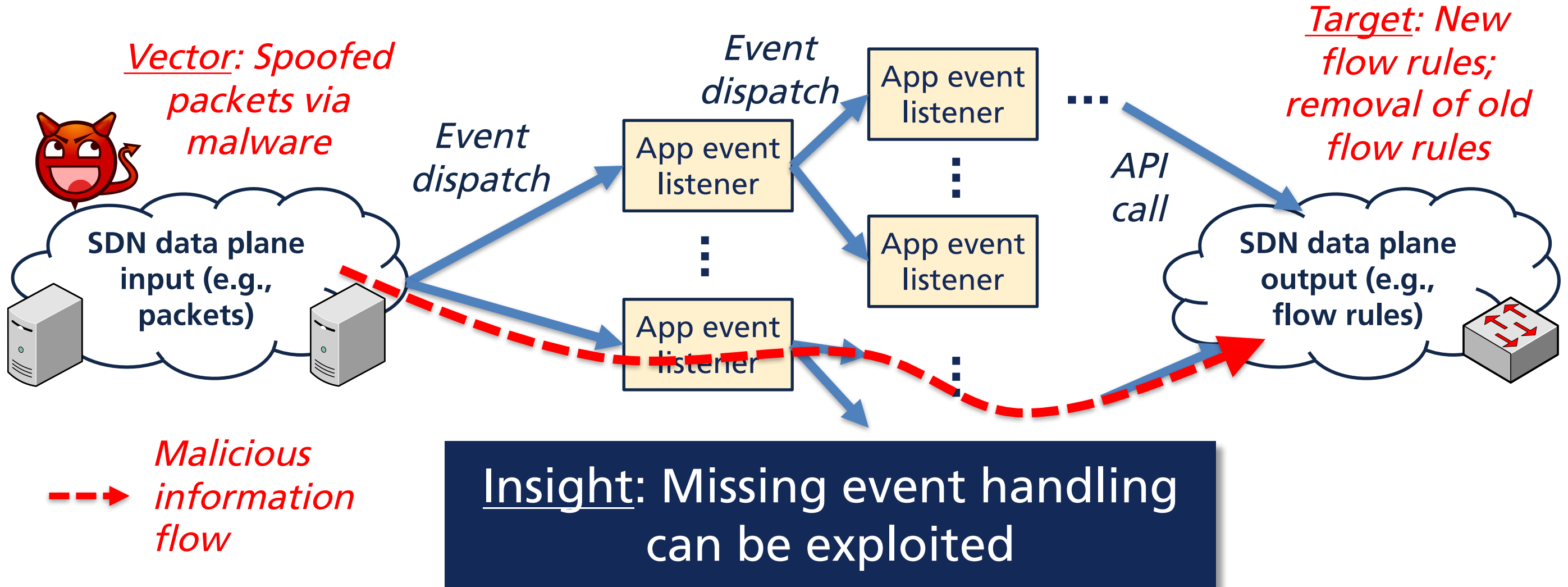
# Cross-Plane Vulnerabilities



# Cross-Plane Vulnerabilities



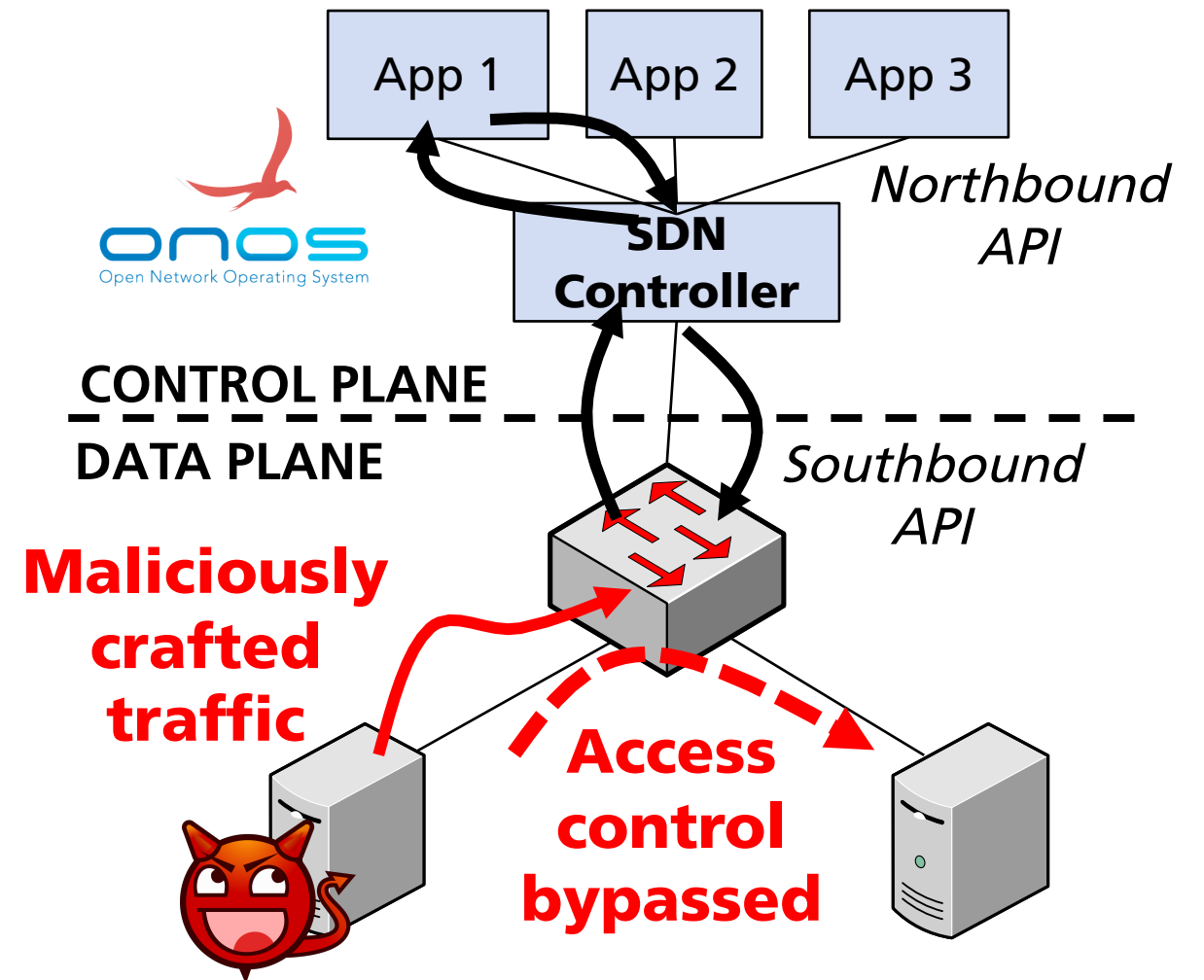
# Cross-Plane Vulnerabilities: **Exploitation**





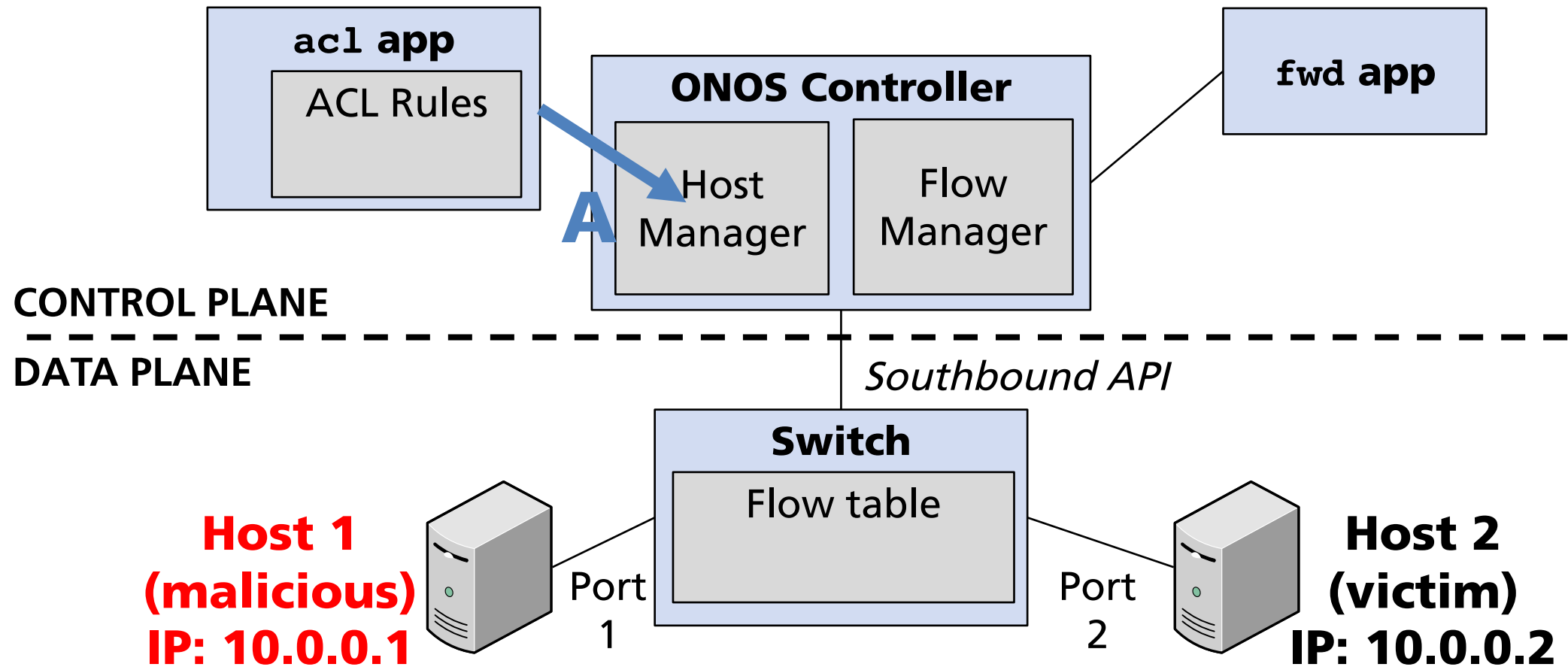
# Data Plane Hosts as Attack Vectors

- Cross-app study led to explore hosts as attackers
- Discovered ONOS data plane firewall vulnerability → **arbitrary lateral movement**
- Reported to ONOS developers (CVE 2018-12691)



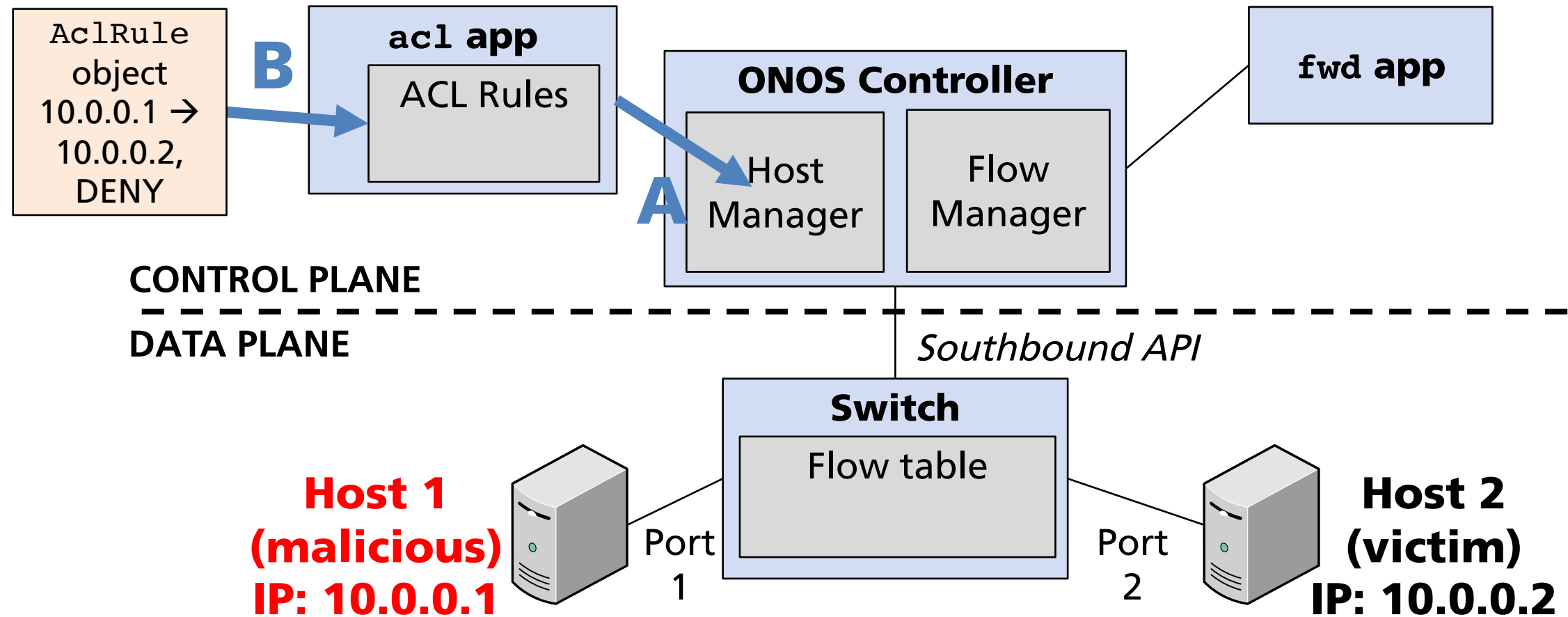
# Anatomy of an Exploit

1. The access control app (ac1) is activated and registers for any host events (A).



# Anatomy of an Exploit

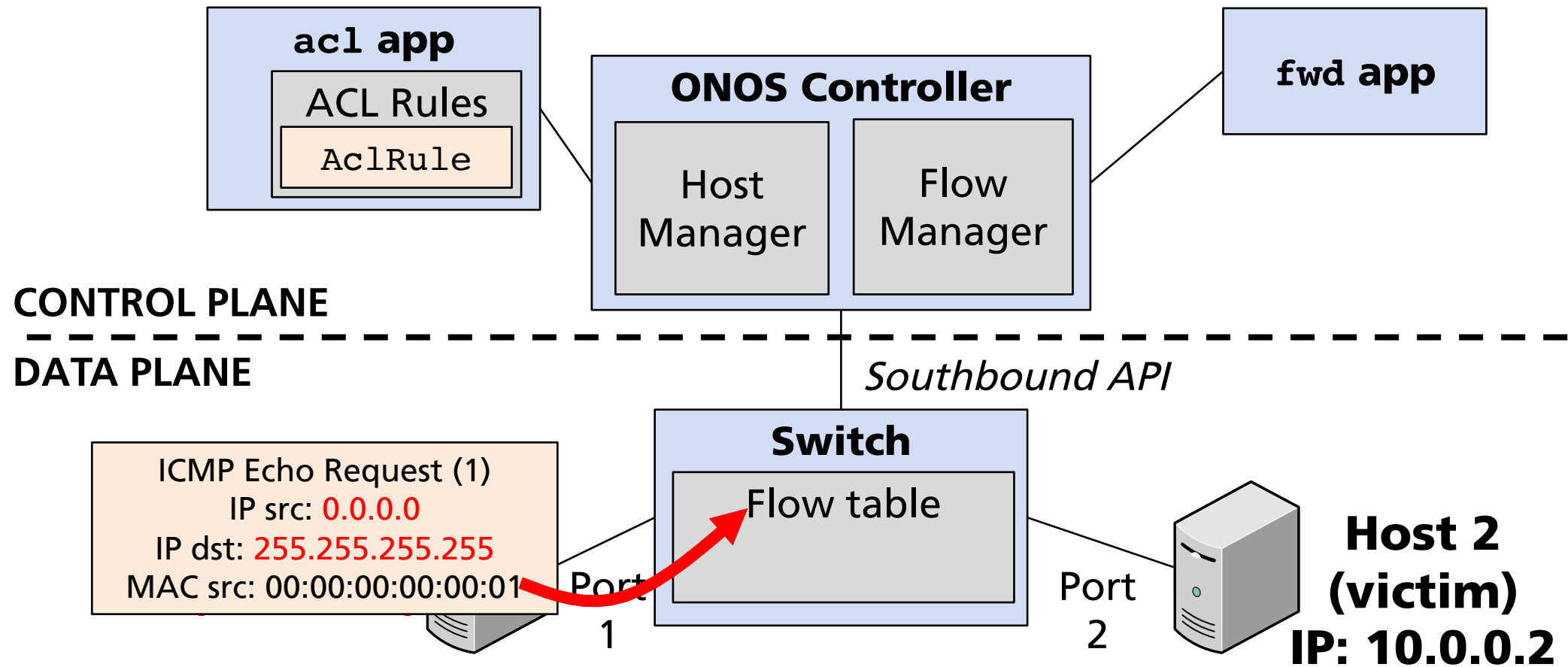
1. The access control app (ac1) is activated and registers for any host events (A). The network operator adds access control policies (B).





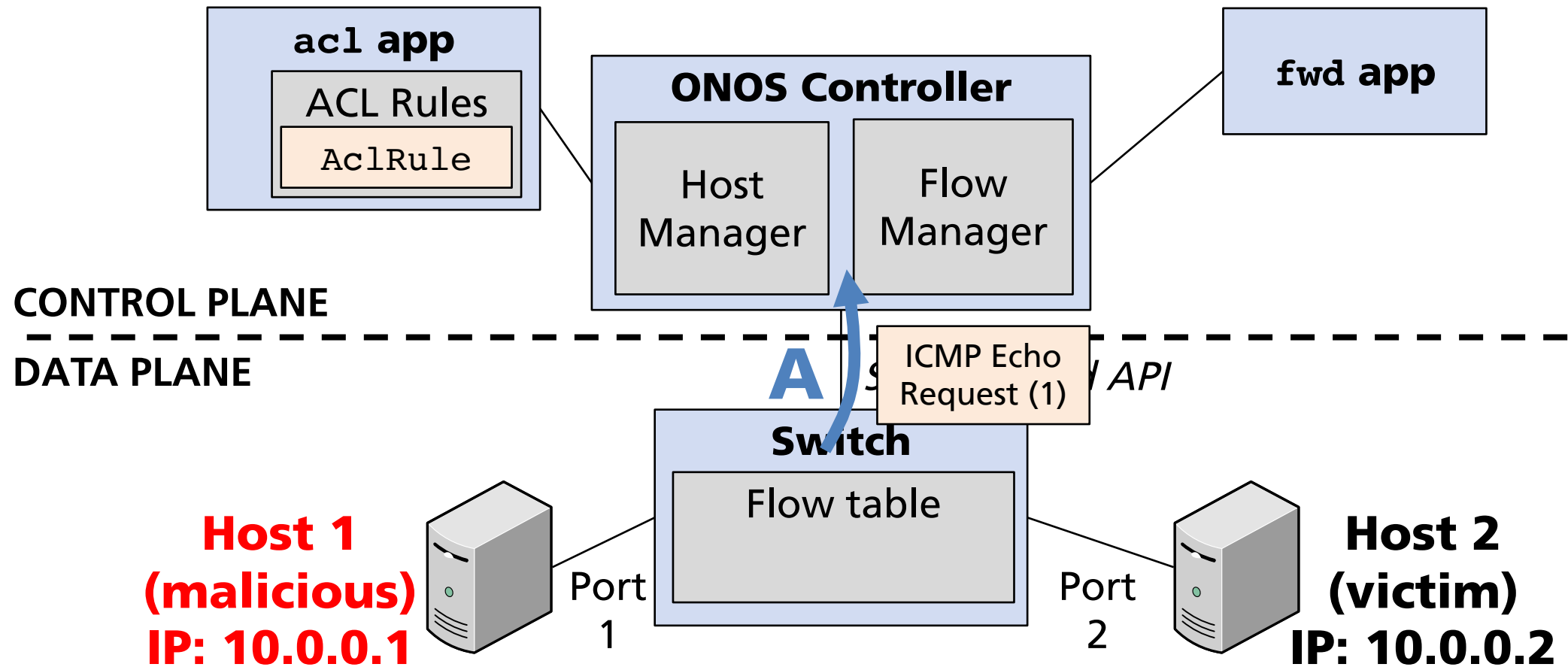
# Anatomy of an Exploit

- Host 1 sends a syntactically correct but semantically invalid ICMP packet with host 1's MAC address into the data plane.



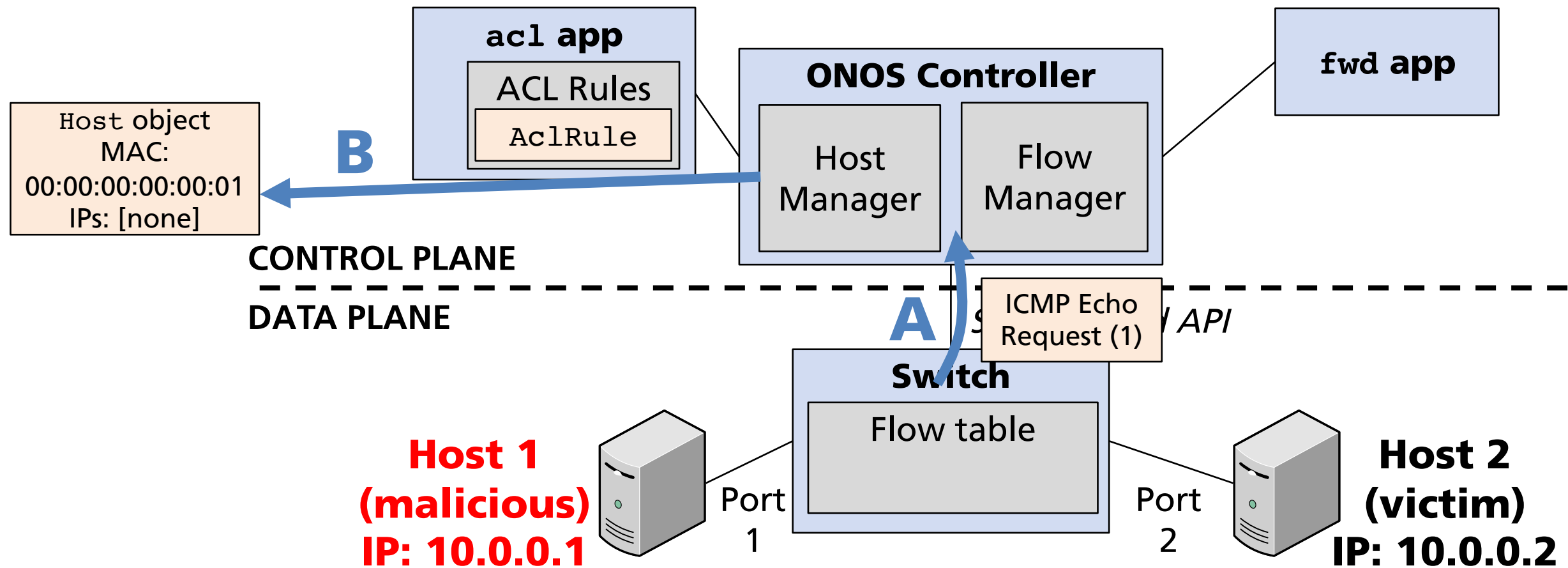
# Anatomy of an Exploit

## 3. ONOS sees the packet (A)



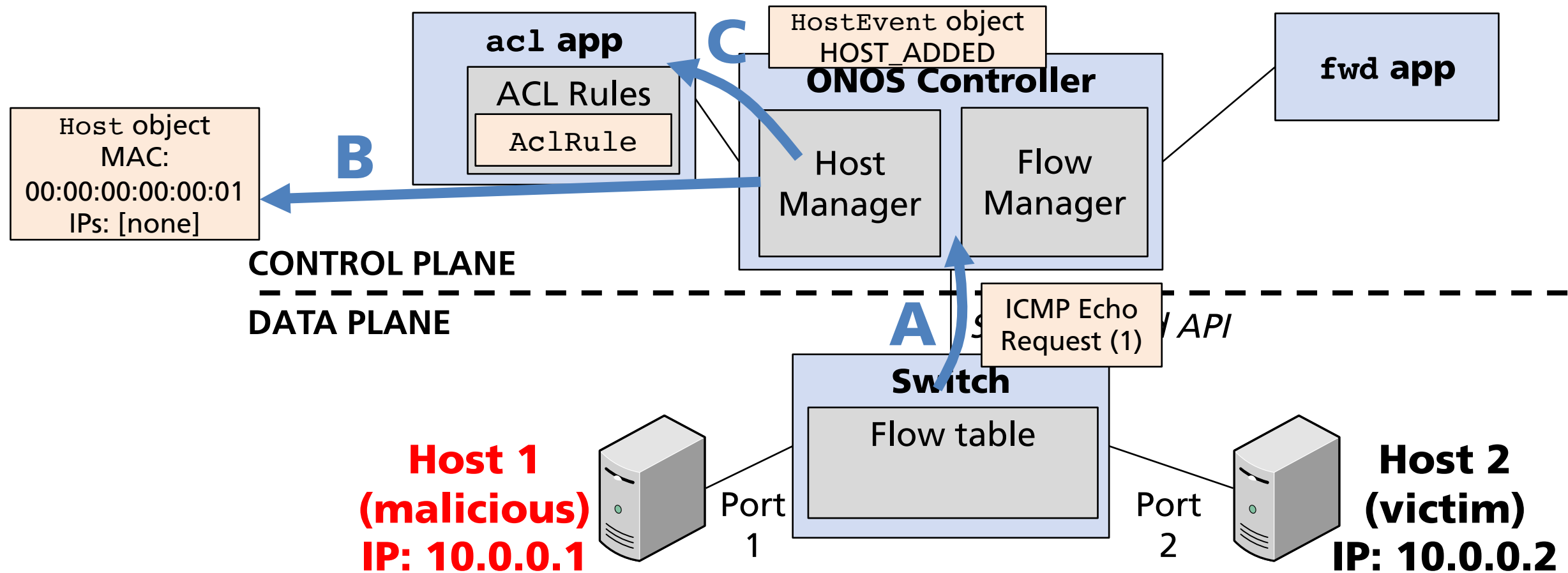
# Anatomy of an Exploit

- ONOS sees the packet (A) and registers a new host with its MAC address but not IP address (B).



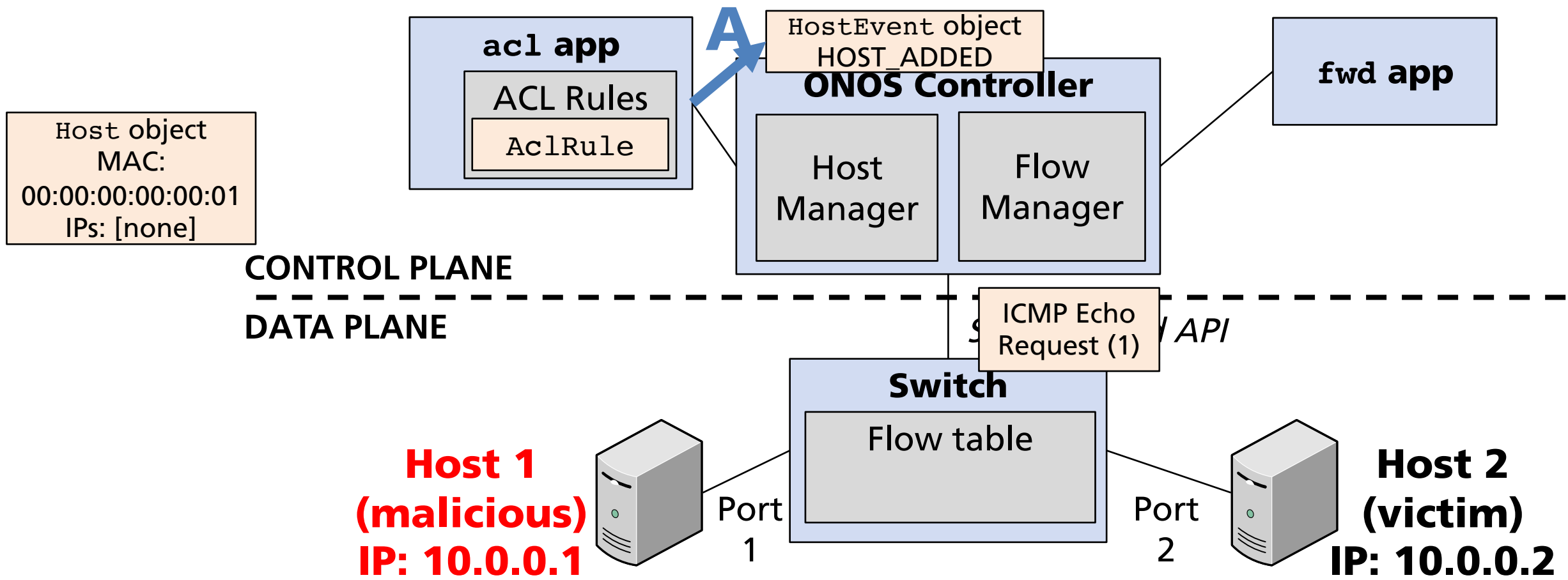
# Anatomy of an Exploit

3. ONOS sees the packet (A) and registers a new host with its MAC address but not IP address (B). It generates a HOST\_ADDED event (C).



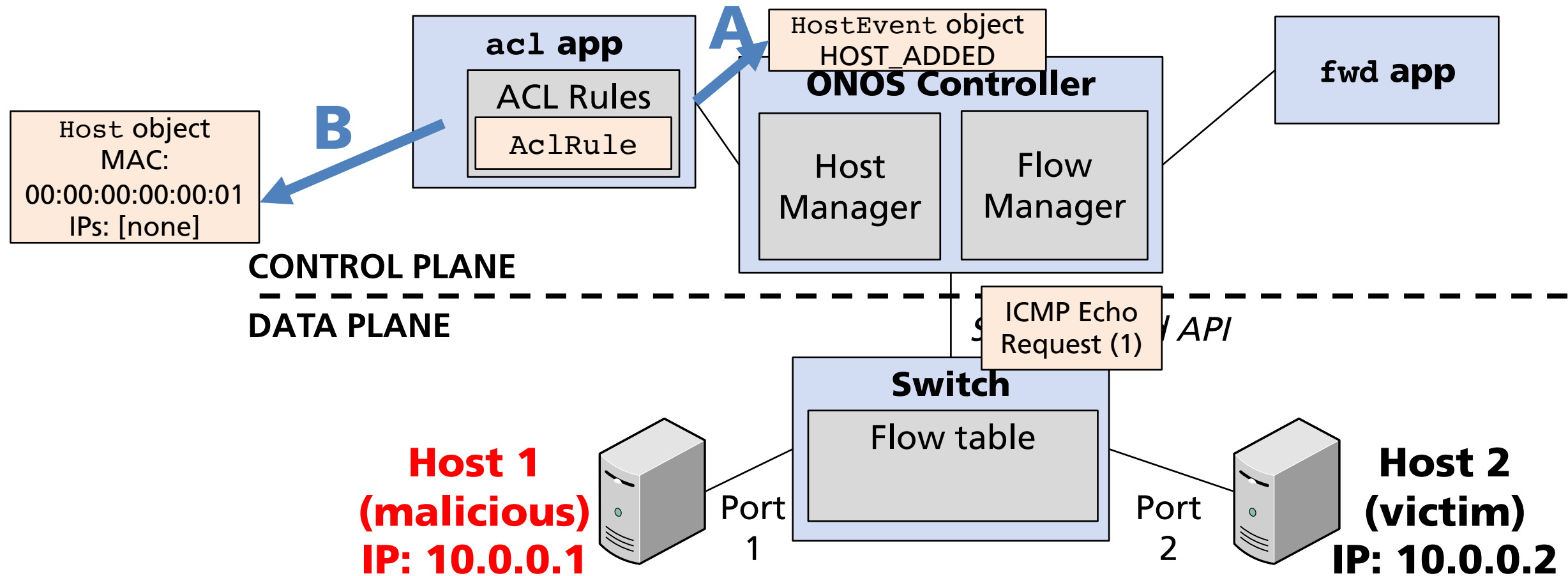
# Anatomy of an Exploit

4. The `acl` app sees the `HOST_ADDED` event (A)



# Anatomy of an Exploit

4. The `acl` app sees the `HOST_ADDED` event (A) and host (B), but since the host doesn't have an IP, the app does not insert flow deny rules.





# Anatomy of an Exploit

4. The `acl` app sees the `HOST_ADDED` event (A) and host (B), but since the host doesn't have an IP, the app does not insert flow deny rules.

```
AclManager.java  
handling new host events  
  
@Override  
public void event(HostEvent event) {  
    // if a new host appears and an existing rule denies  
    // its traffic, a new ACL flow rule is generated.  
    if (event.type() == HostEvent.Type.HOST_ADDED) {  
        DeviceId deviceId = event.subject().location().deviceId();  
        if (mastershipService.getLocalRole(deviceId) == MasterRole.MASTER) {  
            for (AclRule rule : aclStore.getAclRules()) {  
                if (rule.action() != AclRule.Action.ALLOW) {  
                    processHostAddedEvent(event, rule);  
                }  
            }  
        }  
    }  
}
```

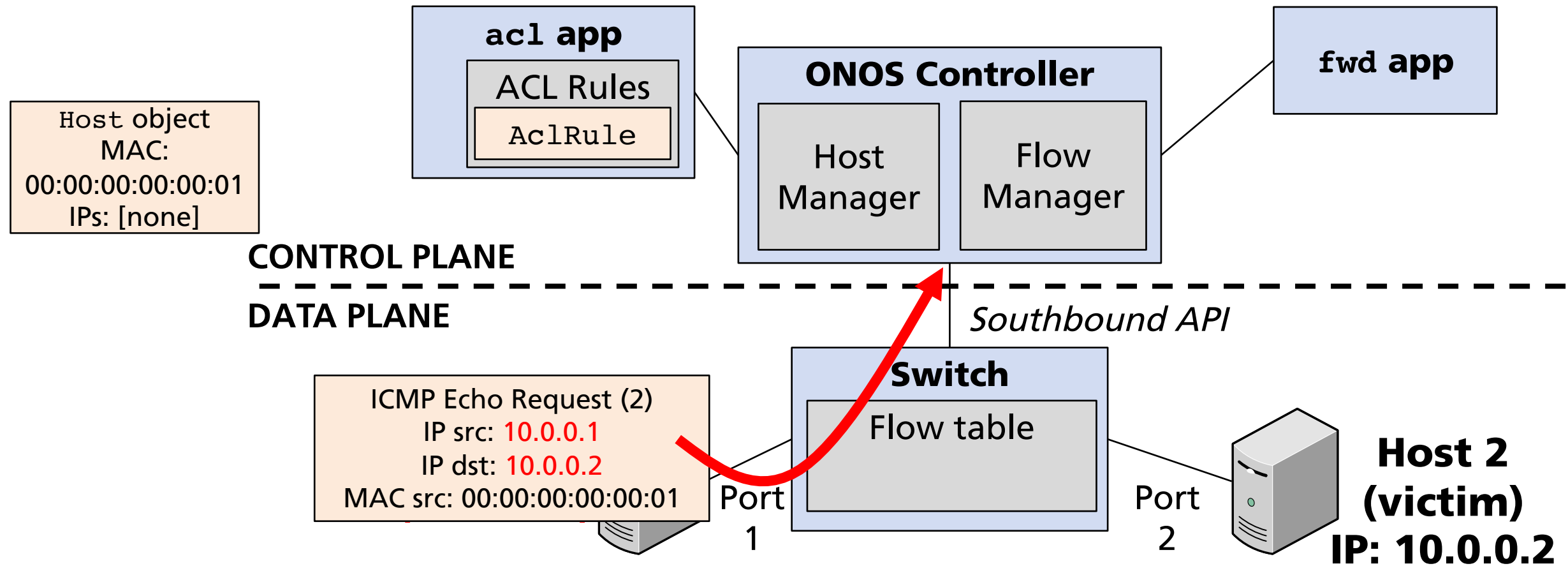
```
AclManager.java processing host events  
private void processHostAddedEvent(HostEvent event, AclRule rule) {  
    DeviceId deviceId = event.subject().location().deviceId();  
    for (IpAddress address : event.subject().ipAddresses()) {  
        if ((rule.srcIp() != null) ?  
            (checkIpInCidr(address.getIp4Address(), rule.srcIp())  
             && checkIpInCidr(address.getIp4Address(), rule.dstIp()))  
            : !aclStore.checkIfRuleWorksInDevice(rule.id(), deviceId)) {  
                List<RuleId> allowingRuleList = aclStore  
                    .getAllowingRuleByDenyingRule(rule.id());  
                if (allowingRuleList != null) {  
                    for (RuleId allowingRuleId : allowingRuleList) {  
                        generateAclFlow(aclStore.getAclRule(allowingRuleId),  
                                       deviceId);  
                    }  
                }  
            }  
        generateAclFlow(rule, deviceId);  
    }  
}
```

**✗ No IP addresses**

**✗ Never gets called**

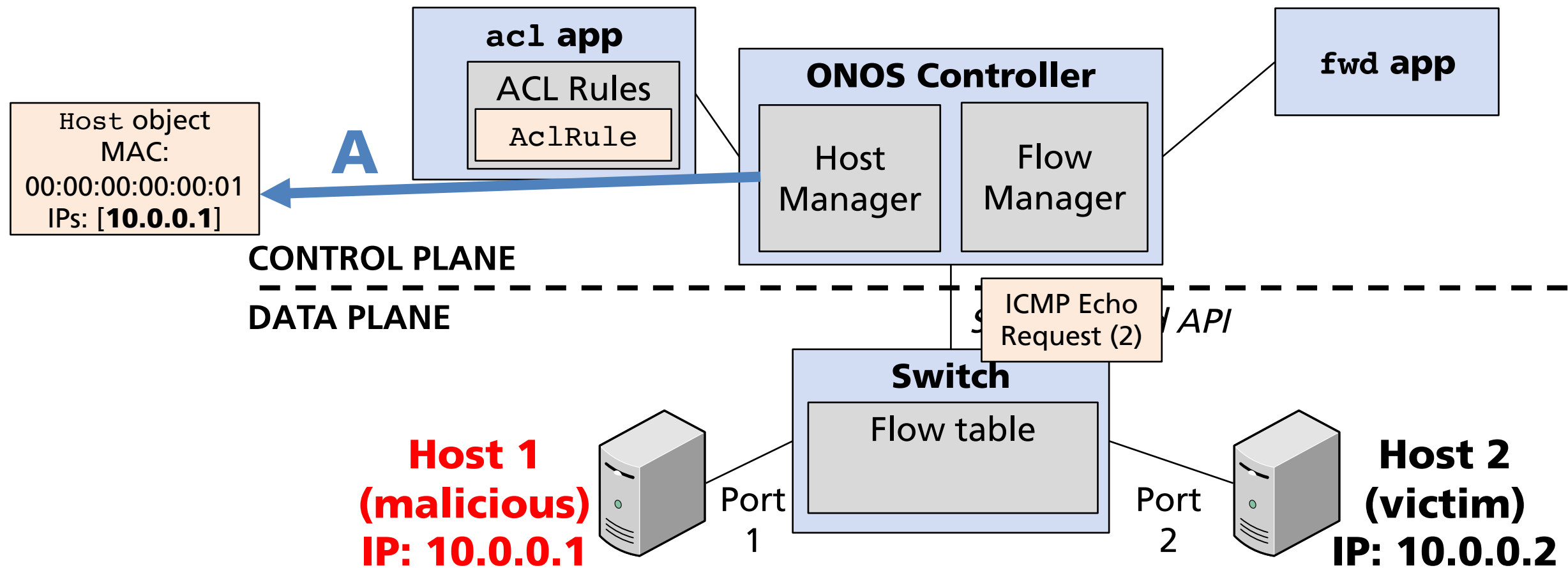
# Anatomy of an Exploit

5. Host 1 attempts to send regular traffic to its desired victim destination (host 2). Since no matching flows exist, ONOS handles the packet.



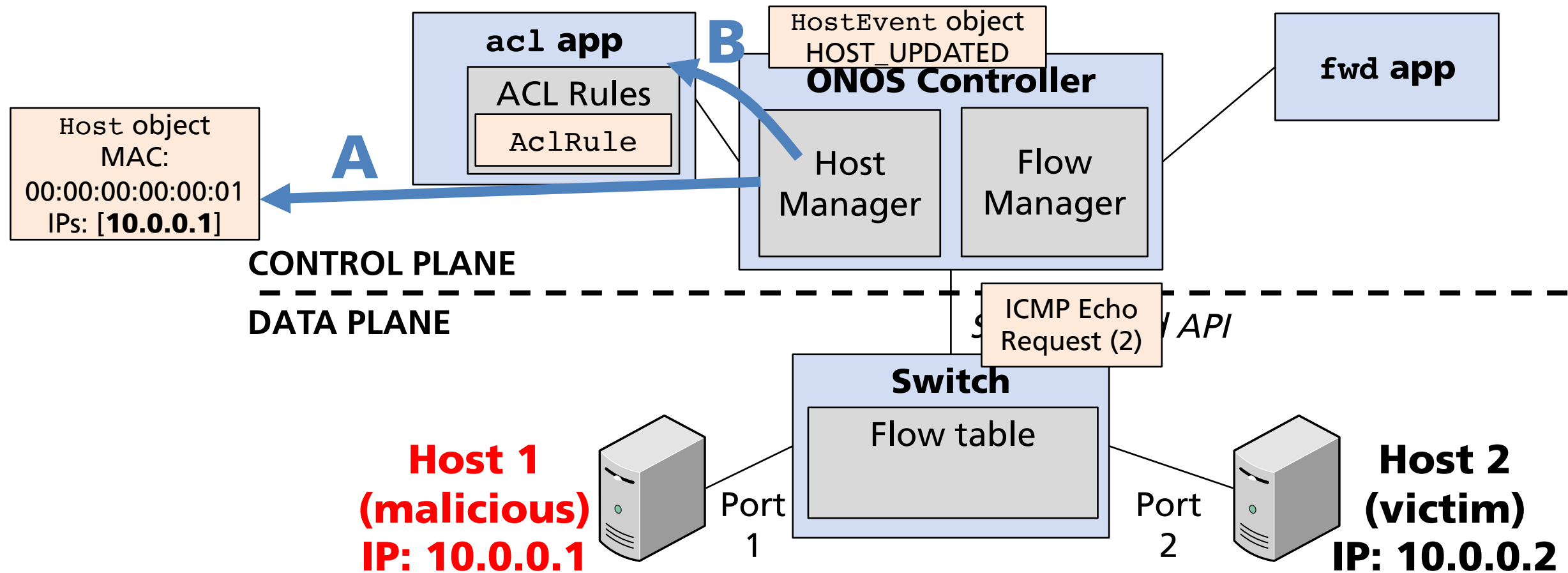
# Anatomy of an Exploit

## 6. ONOS registers host 1's new IP address (A)



# Anatomy of an Exploit

6. ONOS registers host 1's new IP address (A) as a HOST\_UPDATED event (B). ac1 does not handle HOST\_UPDATED events, so it does nothing.



# Anatomy of an Exploit

6. ONOS registers host 1's new IP address (A) as a `HOST_UPDATED` event (B). `ac1` does not handle `HOST_UPDATED` events, so it does nothing.

## `Ac1Manager.java` handling new host events

```
@Override
public void onEvent(HostEvent event) {
    // If a host is added and an existing rule denies
    // its traffic, a new ACL flow rule is generated.
    if (event.type() == HostEvent.Type.HOST_ADDED) {
        DeviceId deviceId = event.subject().location().deviceId();
        if (mastershipService.getLocalRole(deviceId) == MasterRole) {
            for (Ac1Rule rule : ac1Store.getAc1Rules()) {
                if (rule.action() != Ac1Rule.Action.ALLOW) {
                    processHostAddedEvent(event, rule);
                }
            }
        }
    }
}
```

**HOST\_UPDATED events not handled** ✗

**✗ Never gets called**

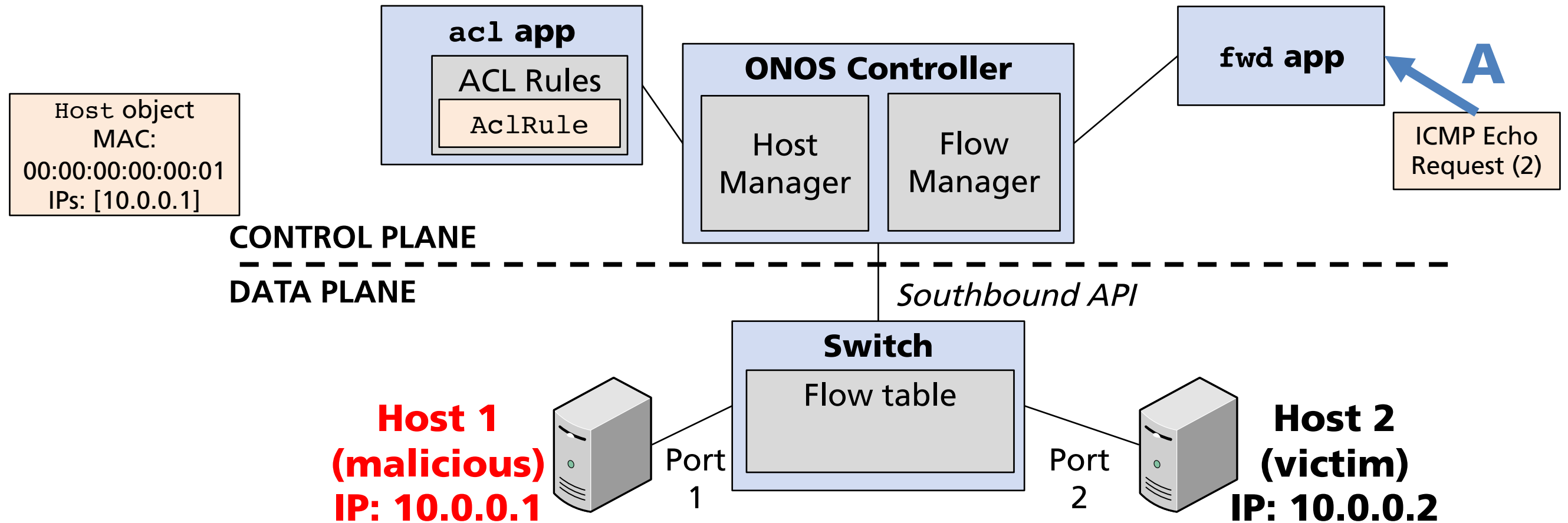
## `Ac1Manager.java` processing host events

```
private void processHostAddedEvent(HostEvent event, Ac1Rule rule) {
    DeviceId deviceId = event.subject().location().deviceId();
    for (IpAddress address : event.subject().ipAddresses()) {
        if ((rule.srcIp() != null) ?
            (checkIpInCidr(address.getIp4Address(), rule.srcIp())
            (checkIpInCidr(address.getIp4Address(), rule.dstIp())
            if (!ac1Store.checkIfRuleWorksInDevice(rule.id(), deviceId)
                List<RuleId> allowingRuleList = ac1Store
                    .getAllowingRuleByDenyingRule(rule.id());
                if (allowingRuleList != null) {
                    for (RuleId allowingRuleId : allowingRuleList) {
                        generateAc1Flow(ac1Store.getAc1Rule(allowingRuleId), deviceId);
                    }
                }
            }
        }
    }
}
```

**✗ Never gets called**

# Anatomy of an Exploit

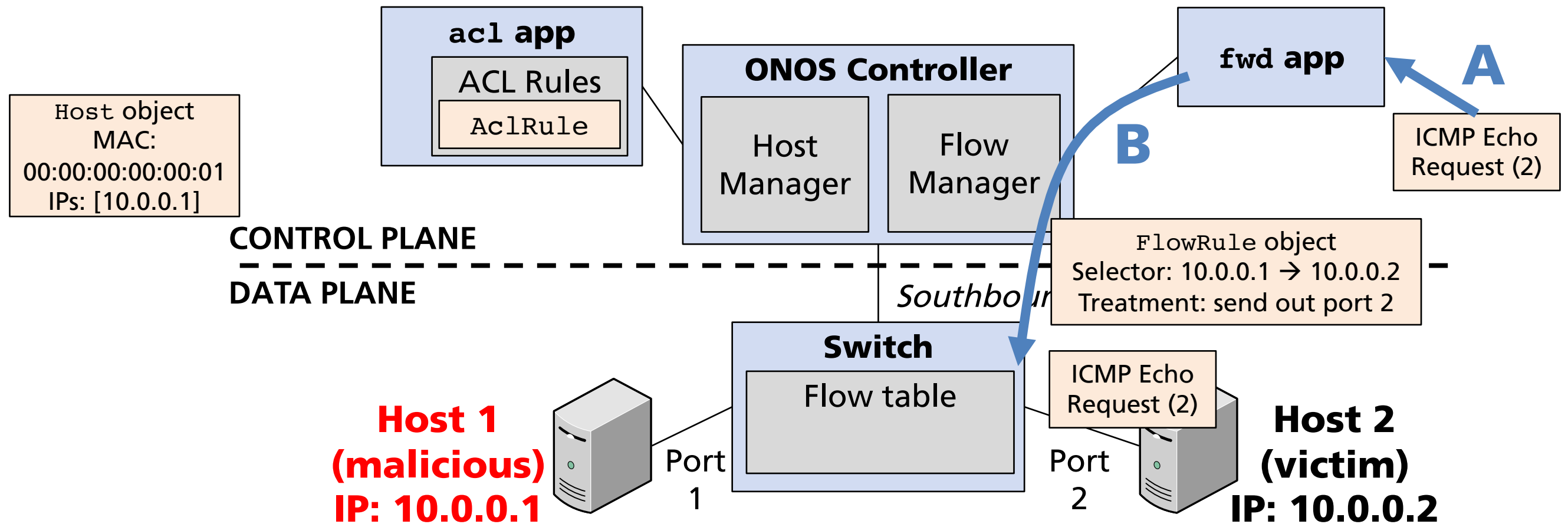
7. The packet gets sent to a second app (A)





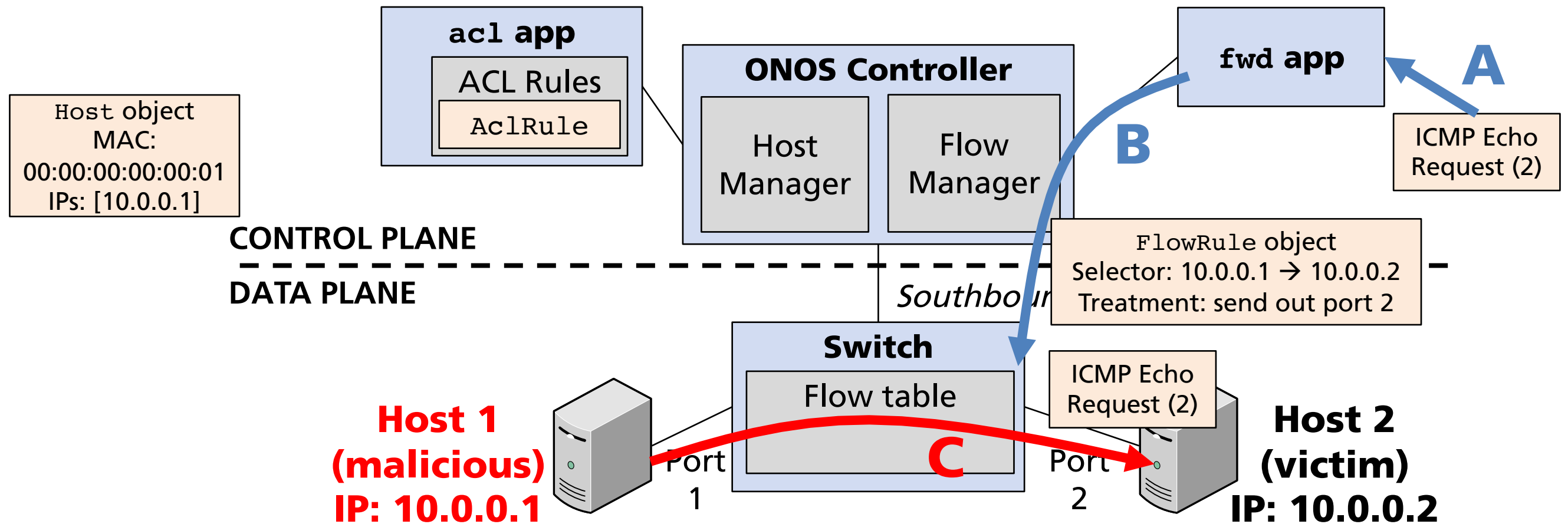
# Anatomy of an Exploit

7. The packet gets sent to a second app (A), which instantiates the flow (allow) rule (B)



# Anatomy of an Exploit

7. The packet gets sent to a second app (A), which instantiates the flow (allow) rule (B) and allows host 1 to communicate with host 2 (C).



# What Makes This Challenging?

No ground truth about what events ought to be handled

Multiple entry points for code analysis

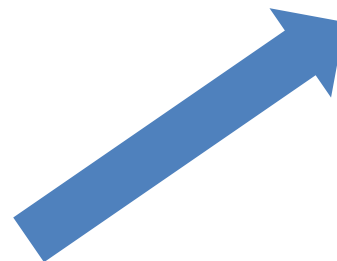
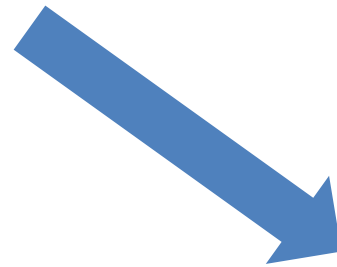
Not all event handling can affect the data plane

# What Makes This Challenging?

No ground truth about what events ought to be handled

Multiple entry points for code analysis

Not all event handling can affect the data plane



## **EVENTSCOPE**

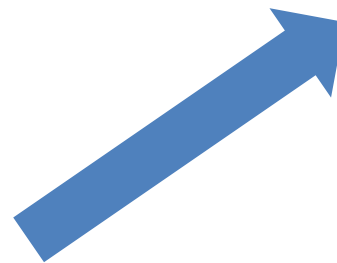
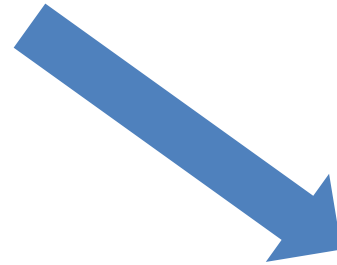
Automated Discovery of Cross-Plane Event-Based Vulnerabilities in SDN

# What Makes This Challenging?

No ground truth about what events ought to be handled

Multiple entry points for code analysis

Not all event handling can affect the data plane



## **EVENTSCOPE**

Automated Discovery of Cross-Plane Event-Based Vulnerabilities in SDN

**Found 14 new vulnerabilities**



# EVENTSCOPE Solution

No ground truth about what events ought to be handled



Cluster apps according to similar functionality

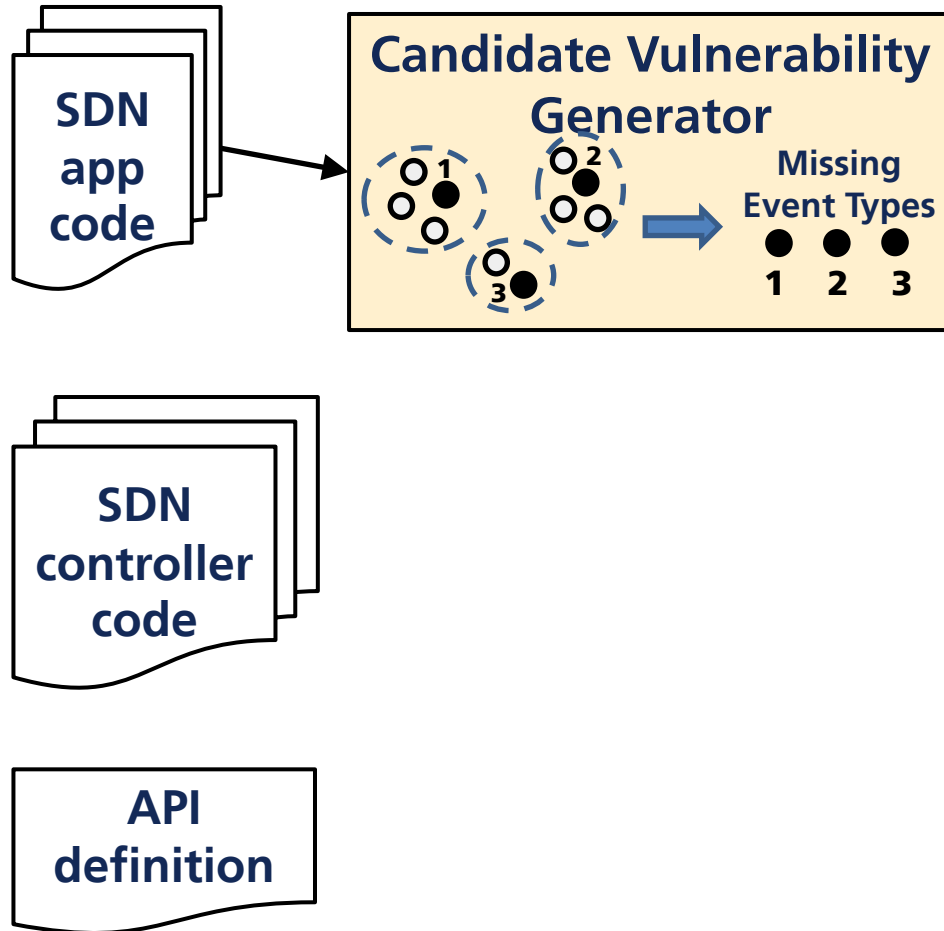
Multiple entry points for code analysis

Not all event handling can affect the data plane



# EVENTSCOPE

## App Event Use



### Host event kind

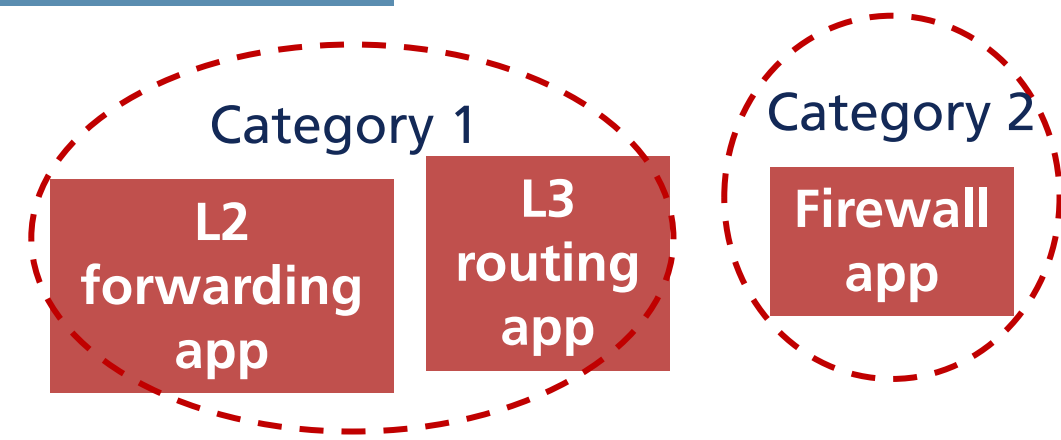
- HOST\_ADDED
- HOST\_REMOVED
- HOST\_UPDATED
- HOST\_MOVED

Event types of **Host** event kind

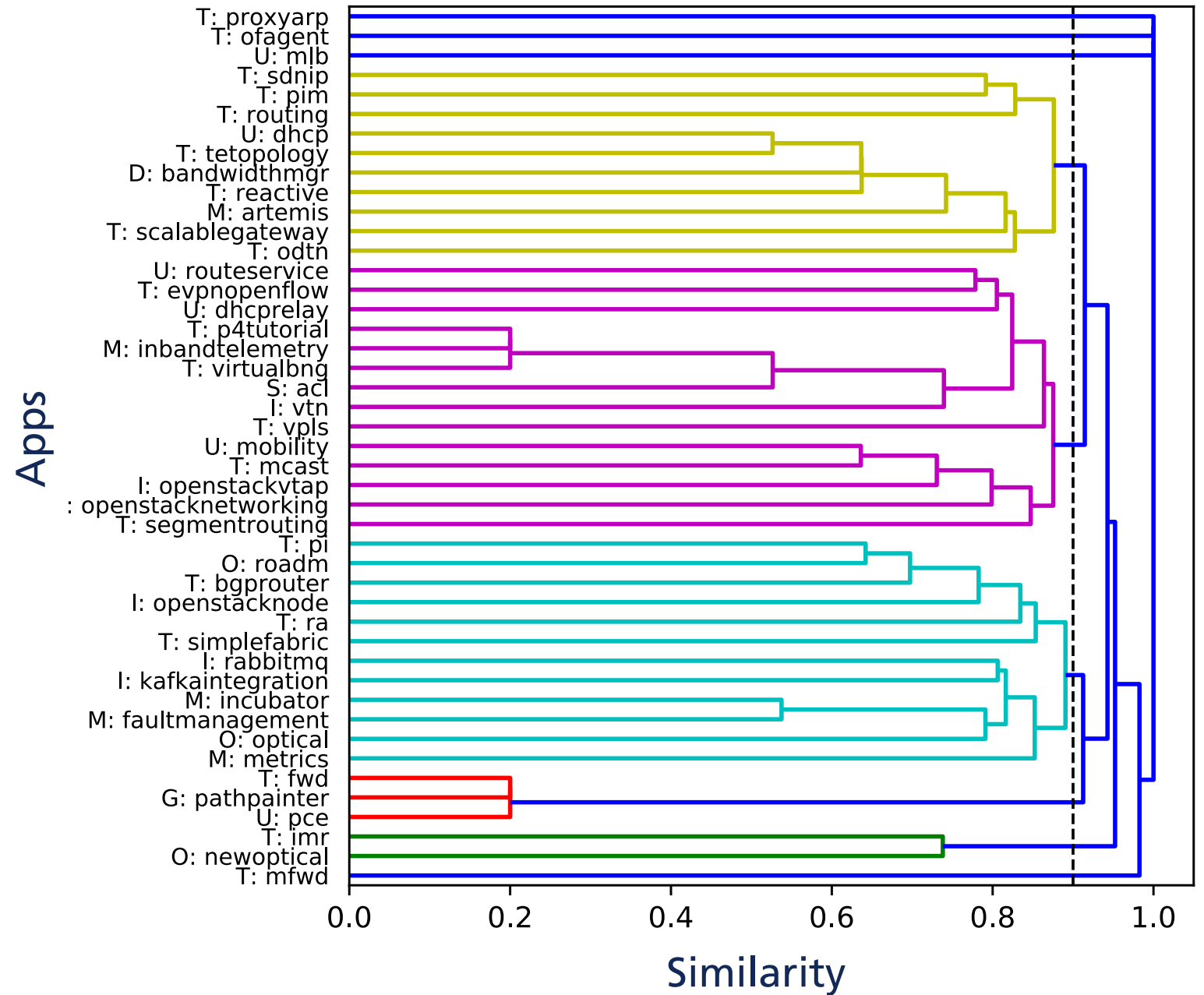
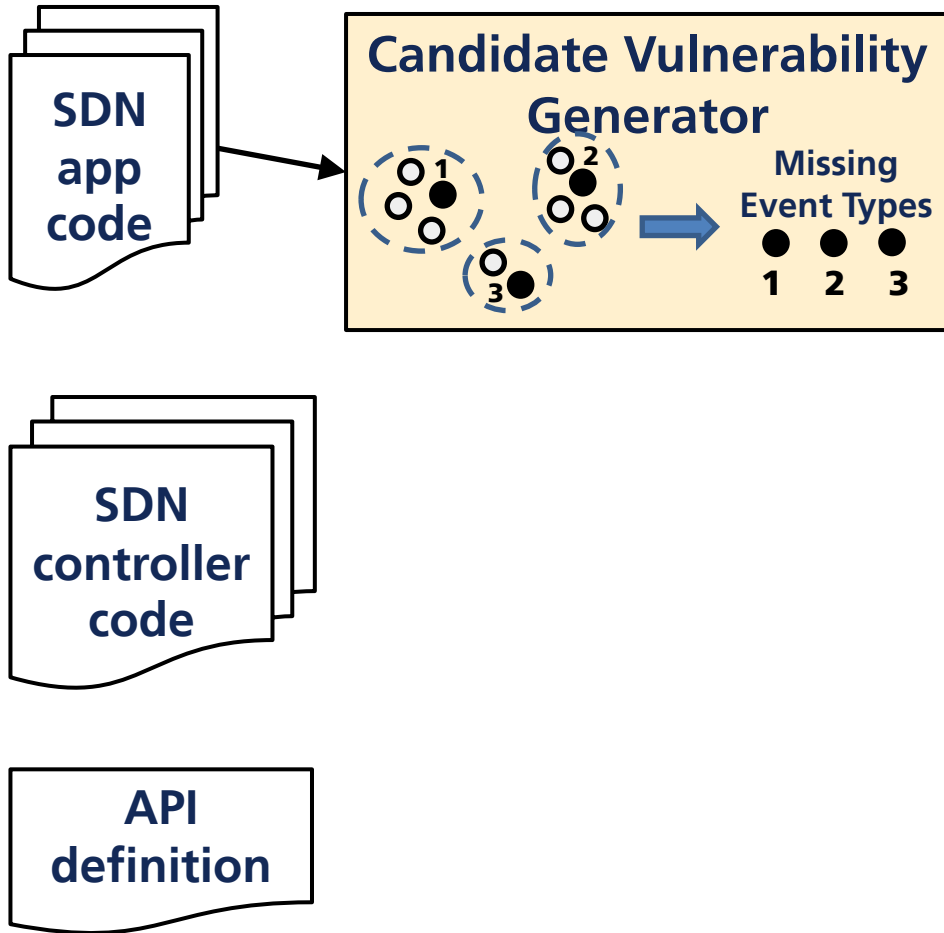
Event types of **Link** event kind

### Link event kind

- LINK\_ADDED
- LINK\_REMOVED
- LINK\_UPDATED



# EVENTSCOPE App Event Use



# EVENTSCOPE Solution

No ground truth about what events ought to be handled

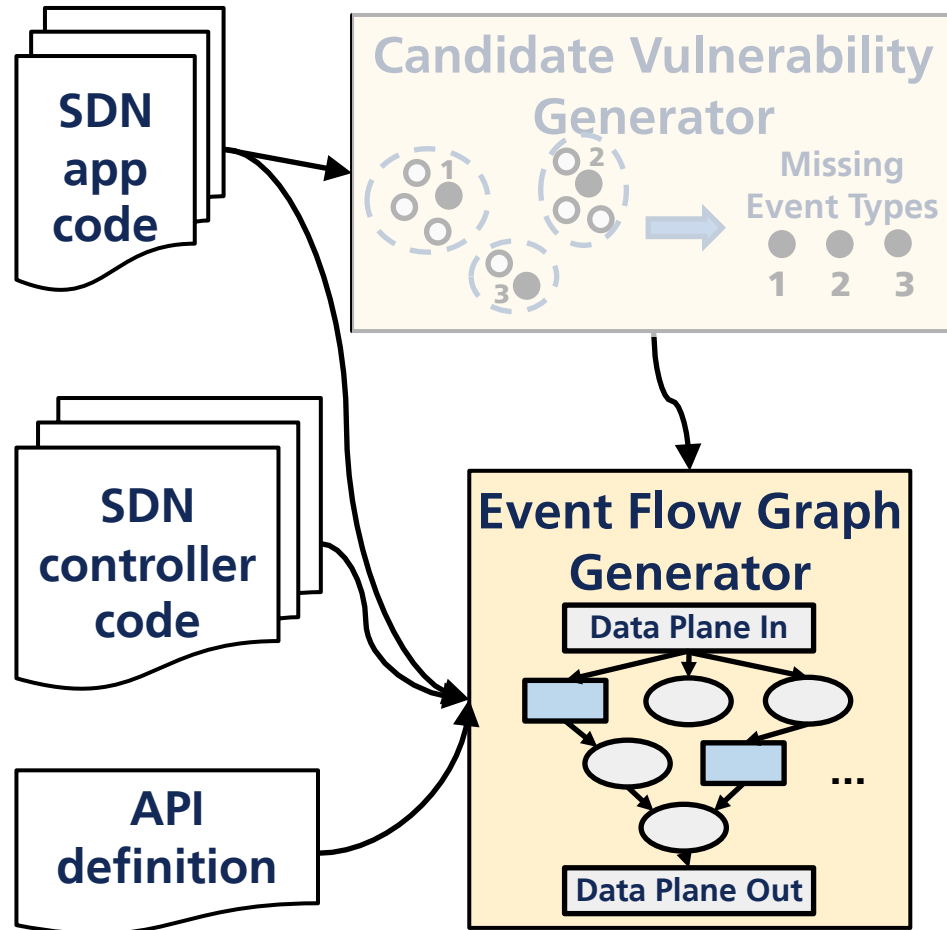
Multiple entry points for code analysis

Not all event handling can affect the data plane



Abstract event flow with graphical model

# EVENTSCOPE Event Flow Graph



Component 1 **Packet**  
event listener

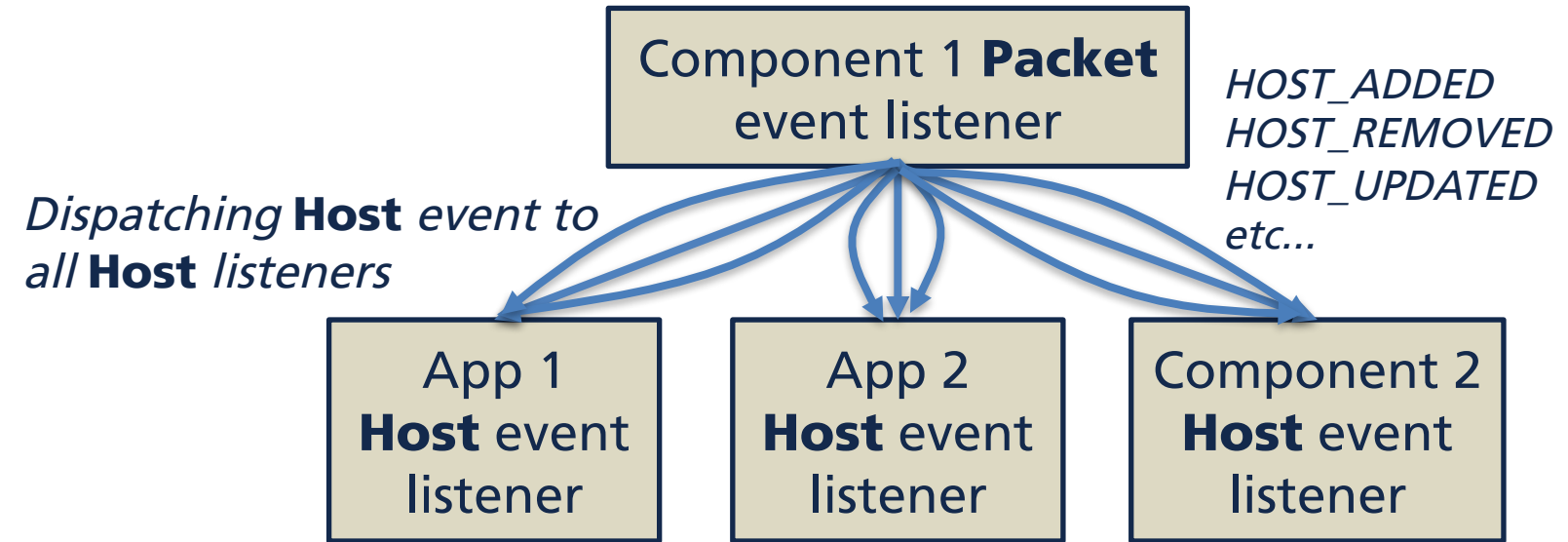
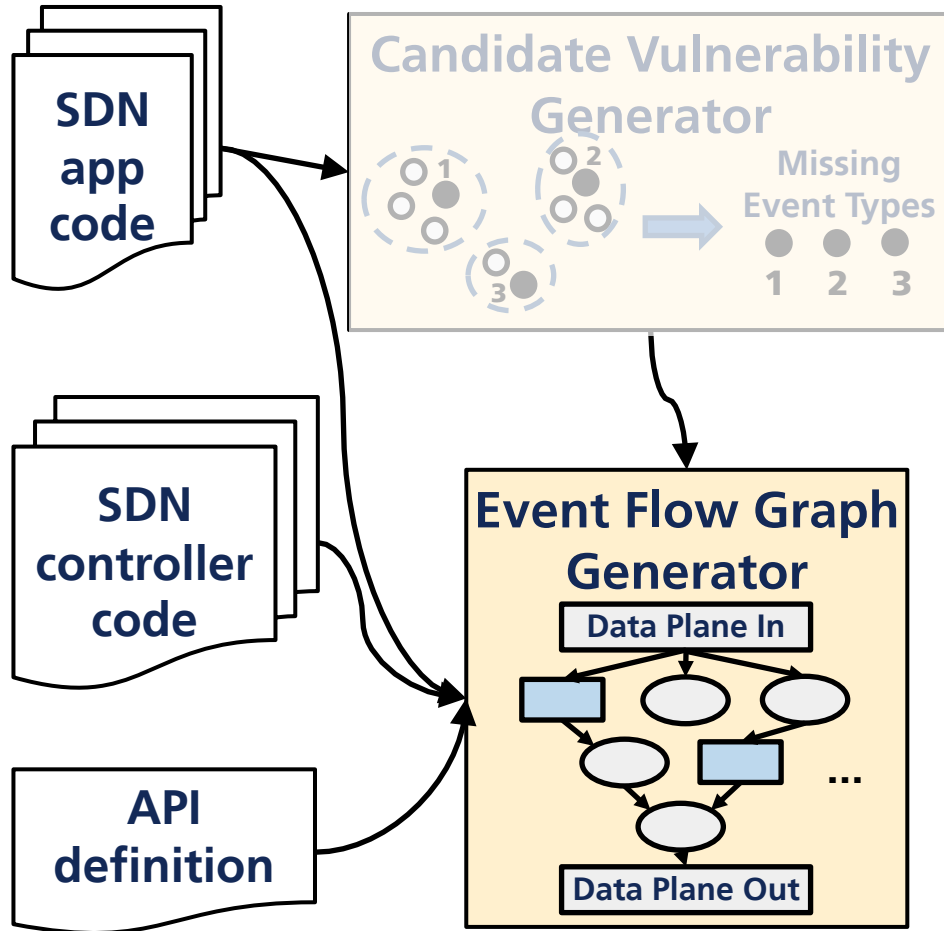
App 1  
**Host** event  
listener

App 2  
**Host** event  
listener

Component 2  
**Host** event  
listener

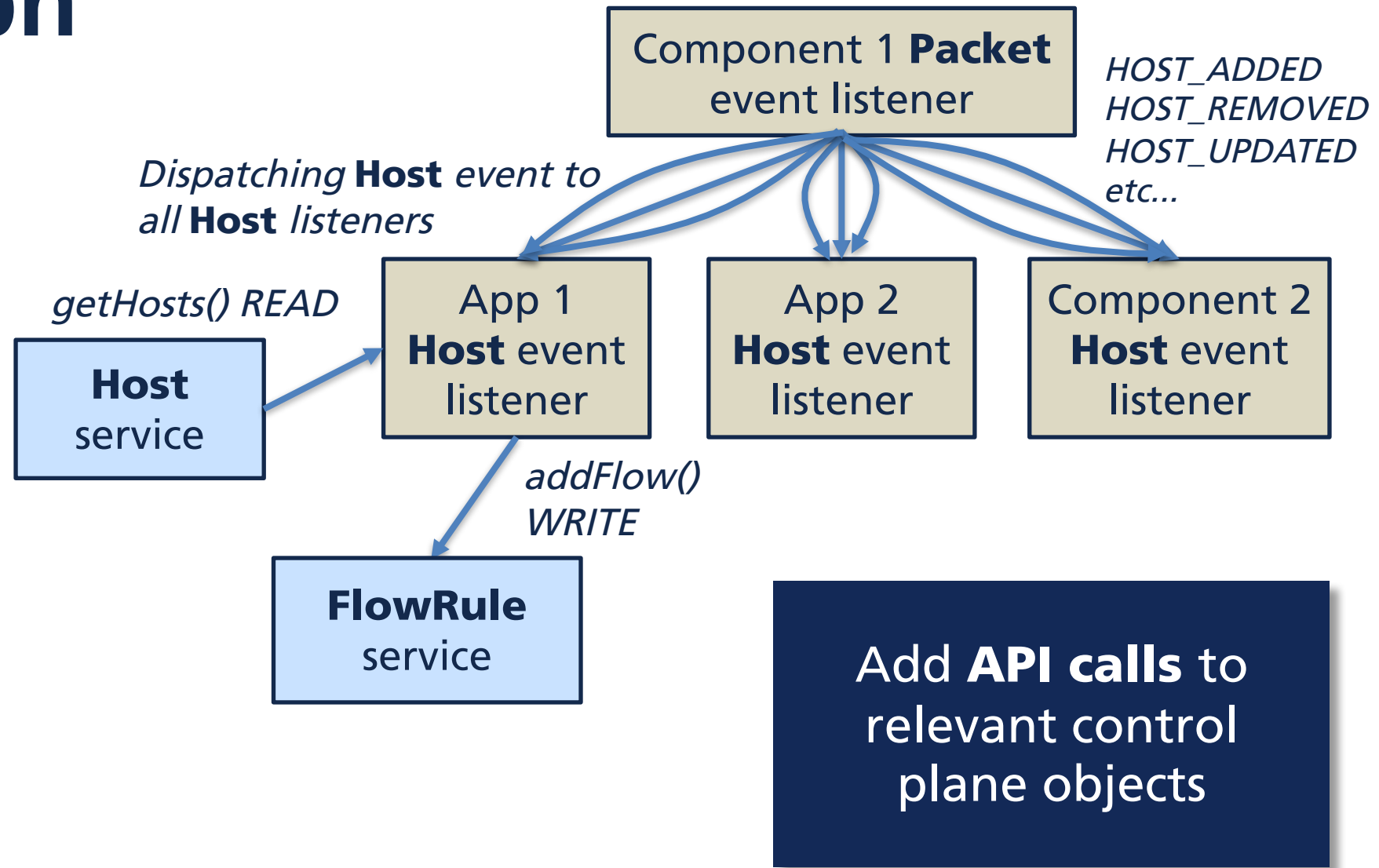
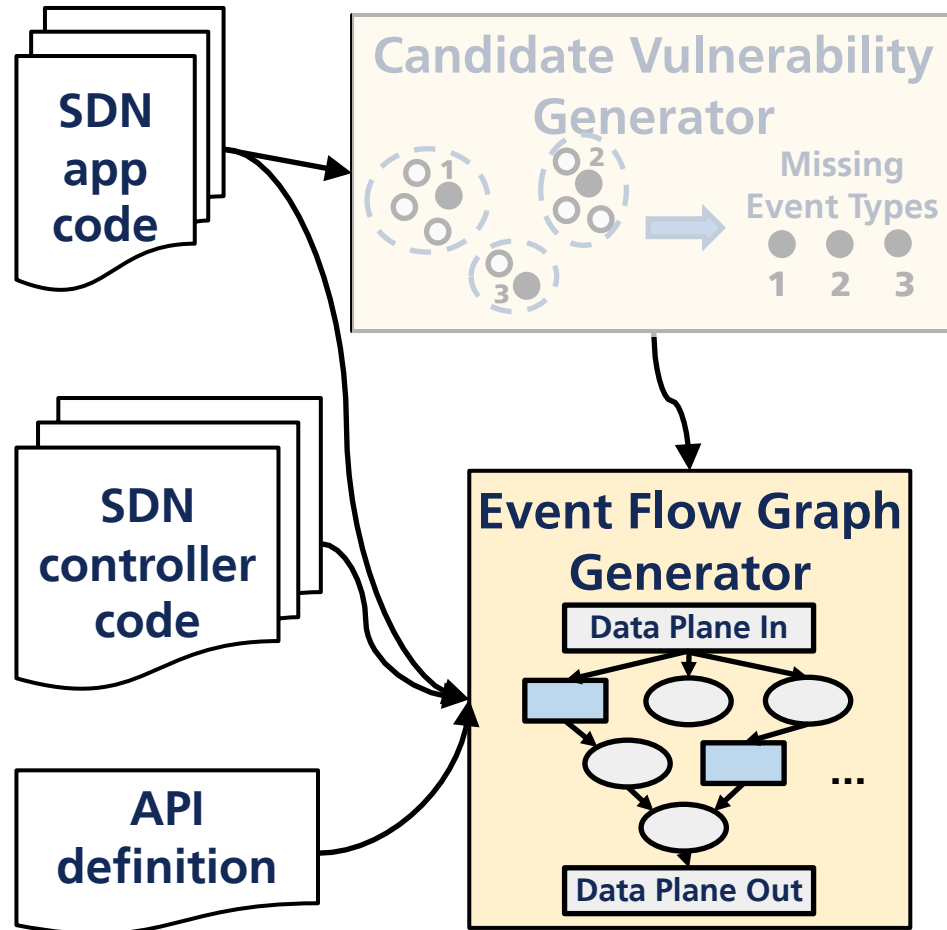
Use event listeners of  
components and apps  
as **entry points**

# EVENTSCOPE Event Flow Graph



**Link event  
dispatchers and  
event listeners**

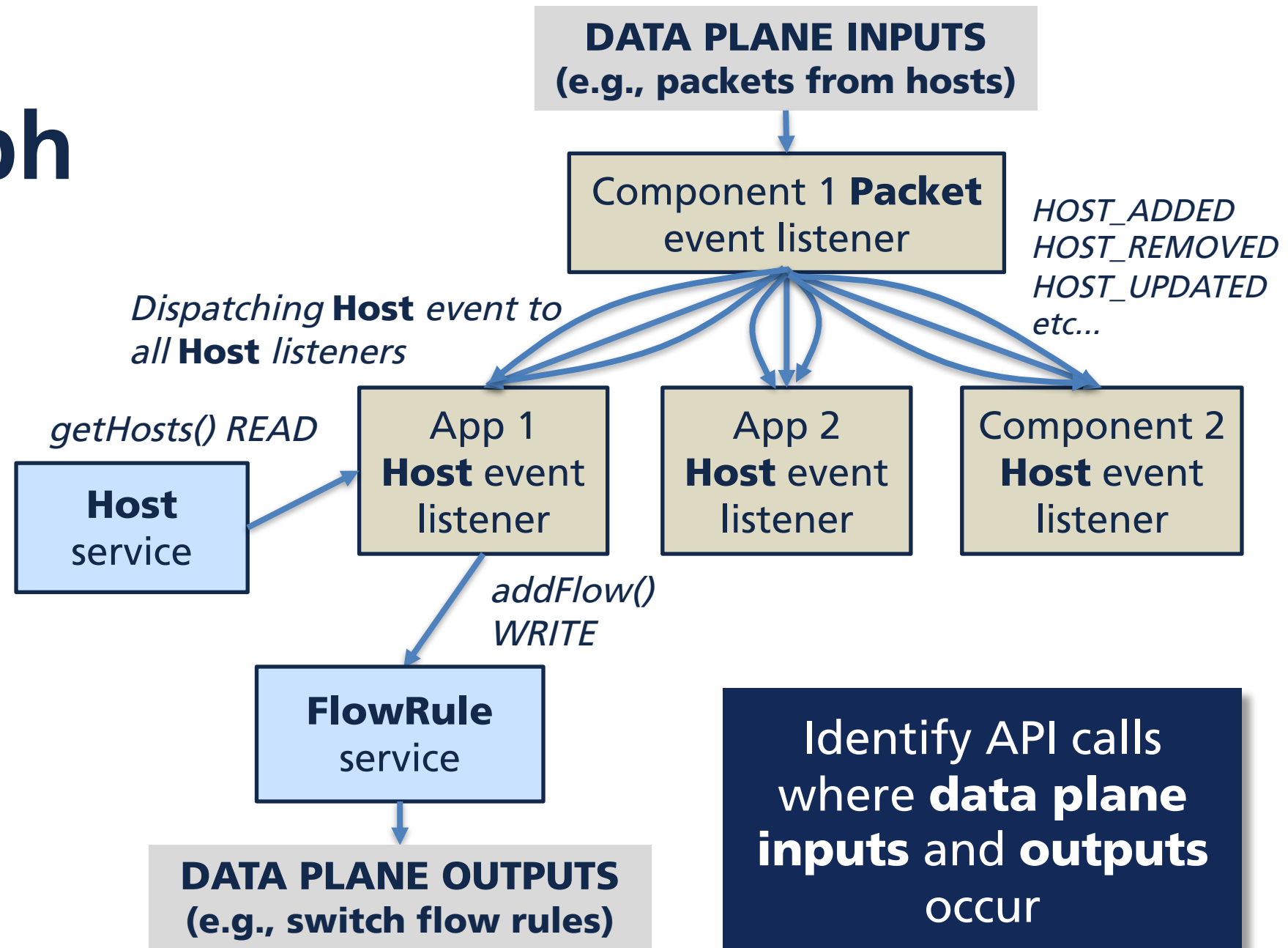
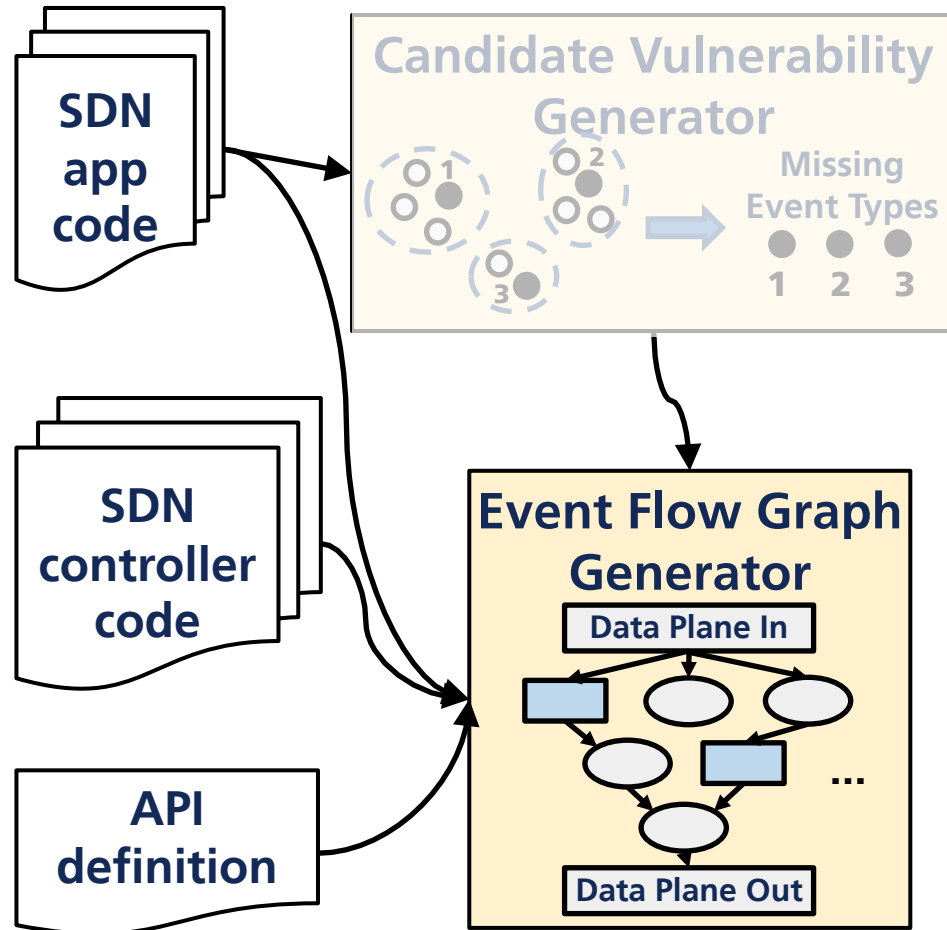
# EVENTSCOPE Event Flow Graph





# EVENTSCOPE

## Event Flow Graph



# EVENTSCOPE Solution

No ground truth about what events ought to be handled

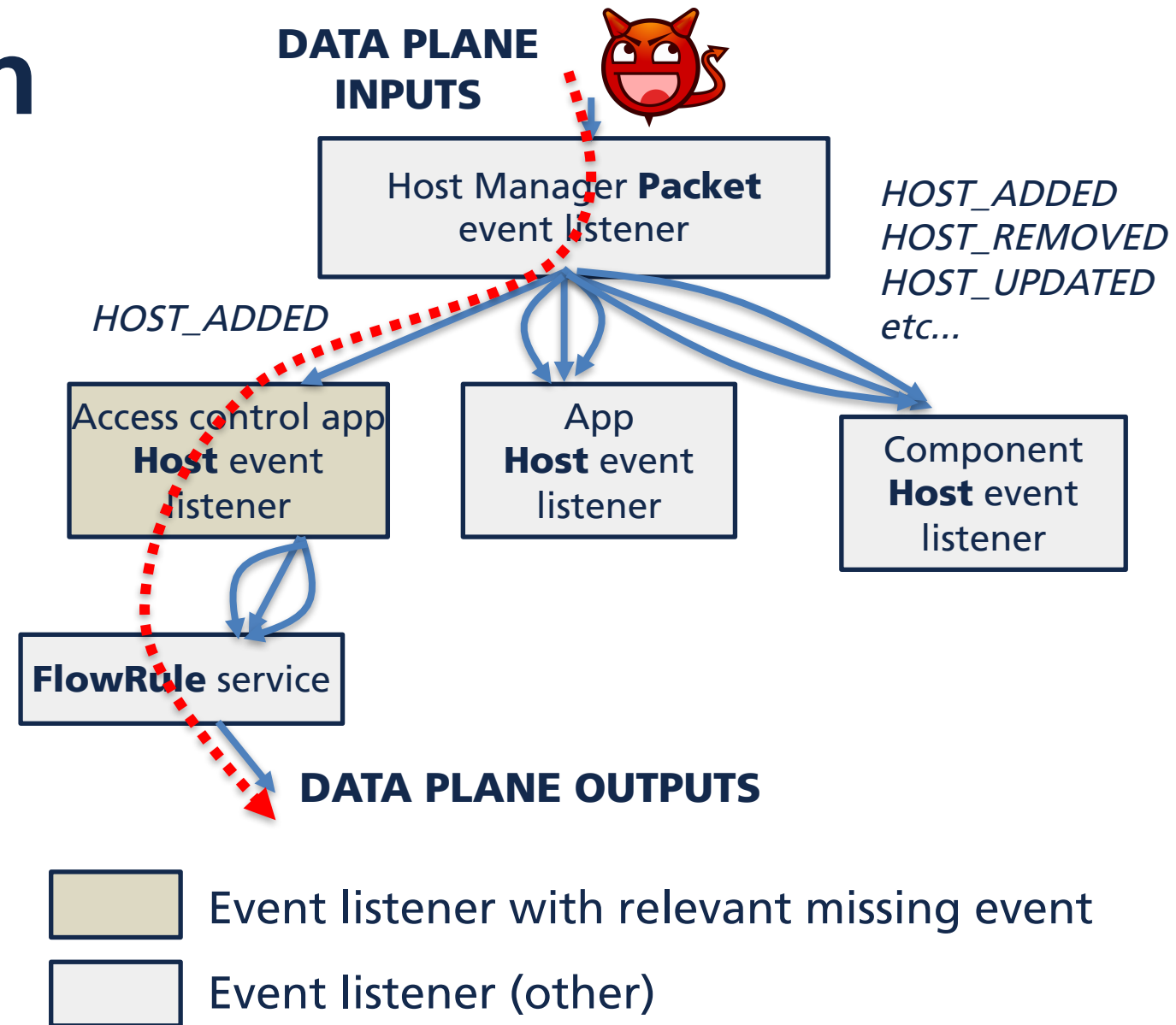
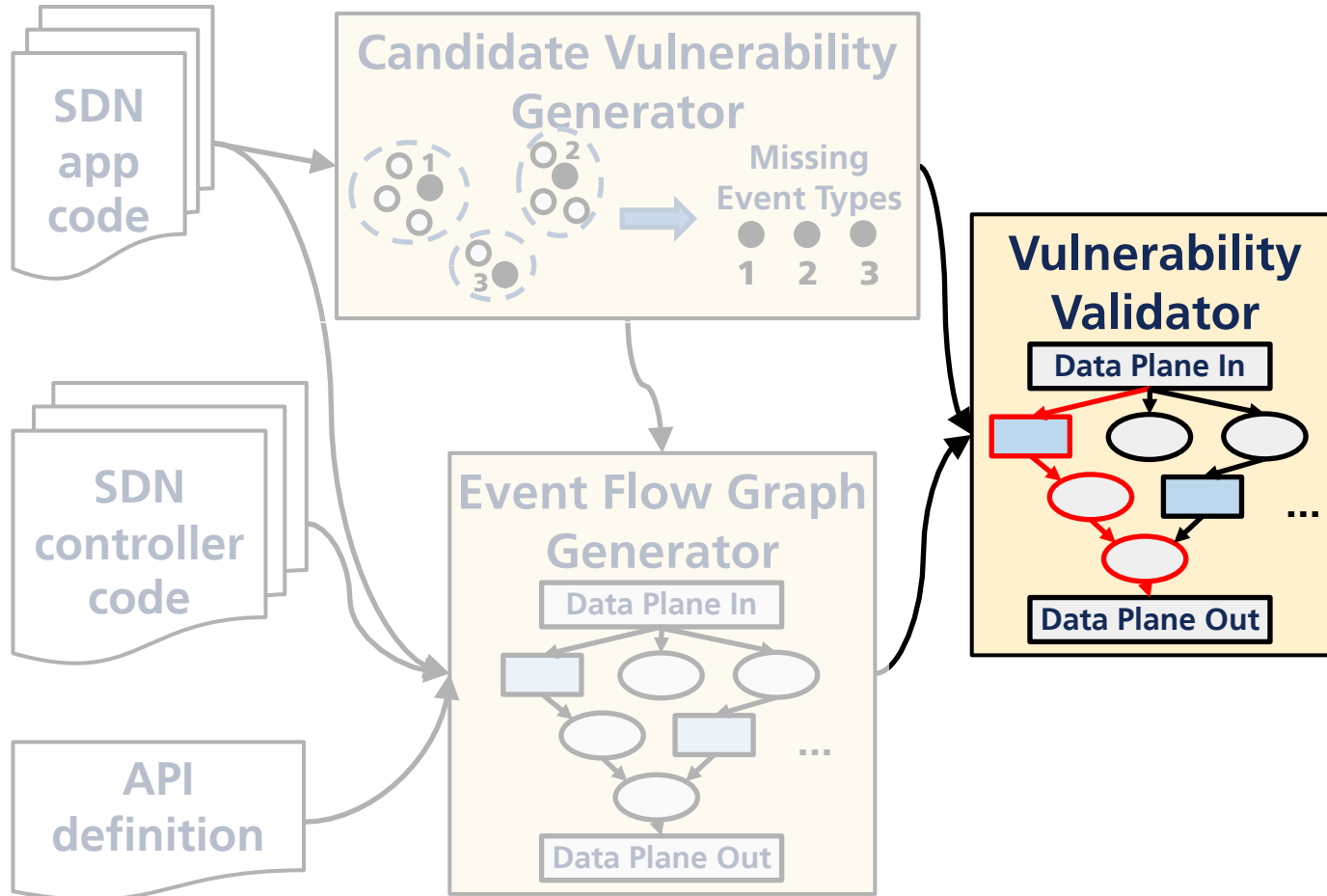
Multiple entry points for code analysis

Not all event handling can affect the data plane

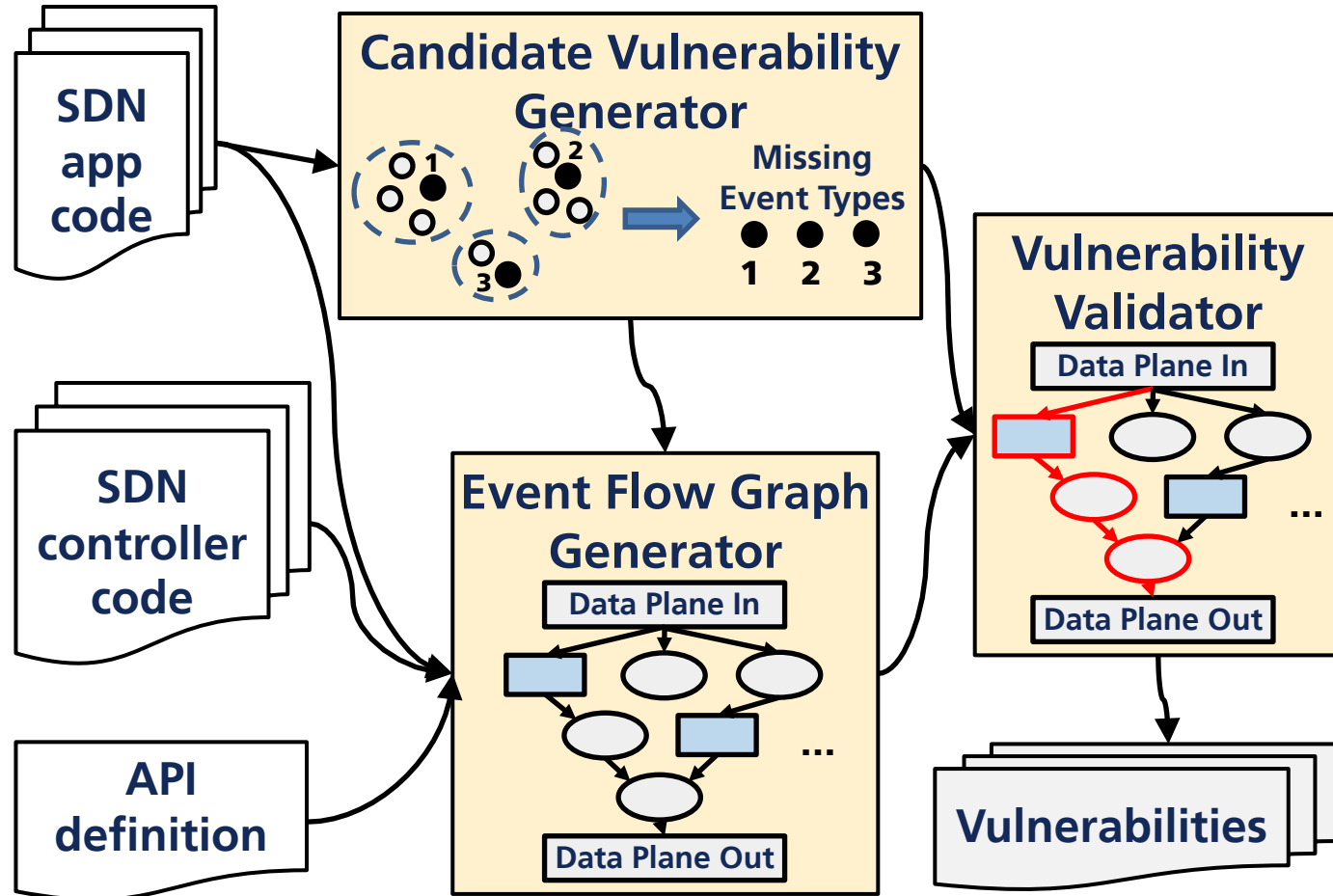


Trace viable control paths in event flow graph

# EVENTSCOPE Vulnerability Validation



# EVENTSCOPE Evaluation



Reported **14** vulnerabilities to ONOS Security Team and requested CVE identifiers

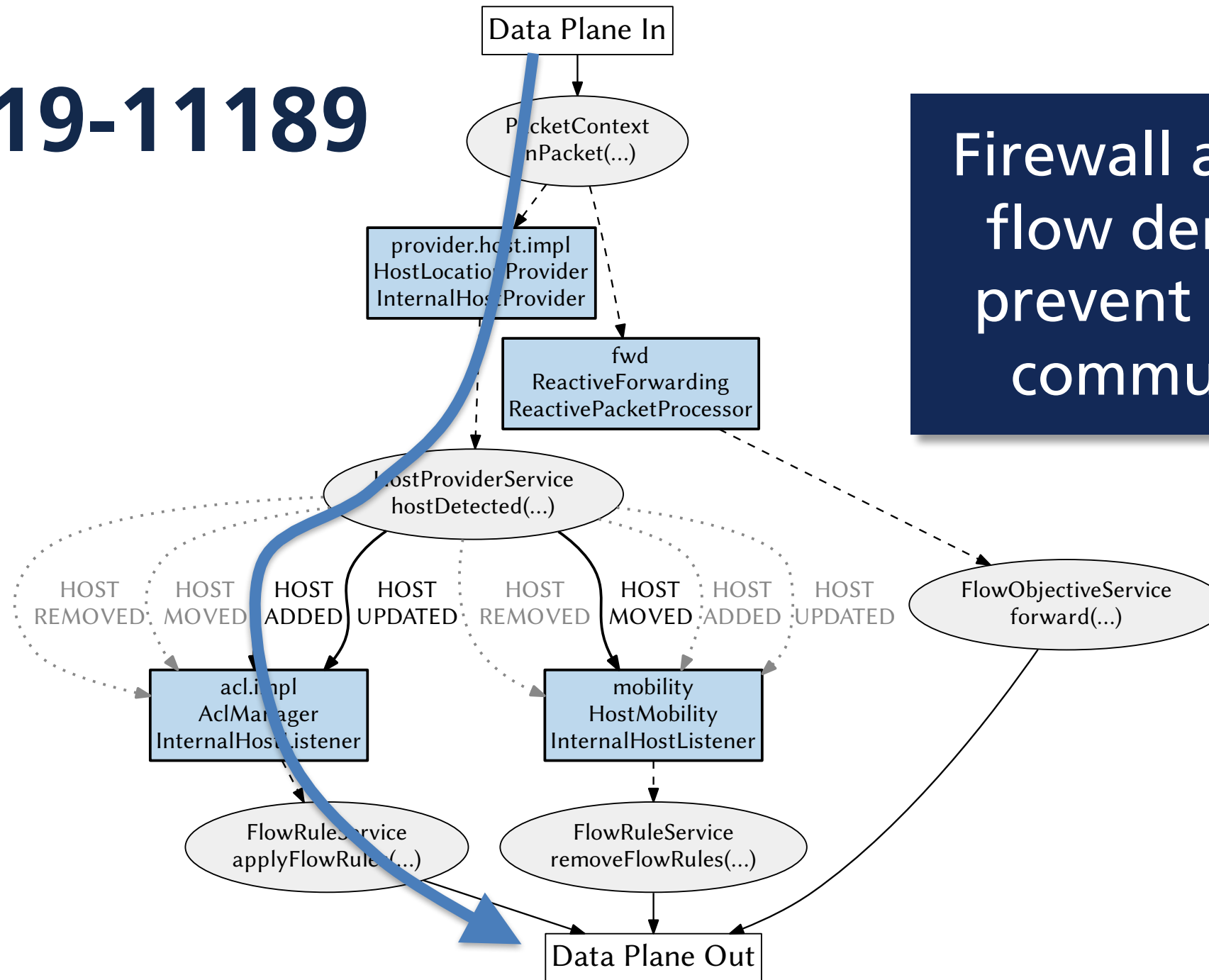


TABLE 7. Event List

#	ID	Component	Unhandled type	Example event flow graph path showing potential data plane input to data plane effect
1	CVE-2019-11189	ac	HOST_UPDATED, HOST_MOVED	See Figures 8 and 9 for event flow graph examples.
2	CVE-2019-16300	acl	HOST_REMOVED	
3	CVE-2019-16298	virtualbng	HOST_MOVED	DPIn $\xrightarrow{DP\_IN}$ inPacket() $\xrightarrow{API\_READ}$ provider.host.InternalHostProvider $\xrightarrow{API\_WRITE}$ hostDetected() $\xrightarrow{HOST\_ADDED}$ virtualbng.InternalHostListener $\xrightarrow{API\_WRITE}$ startMonitoringIp() $\xrightarrow{DP\_OUT}$ DPOut
4	CVE-2019-16298	virtualbng	HOST_REMOVED	
5	CVE-2019-16298	virtualbng	HOST_UPDATED	
6	CVE-2019-16299	mobility	HOST_ADDED	DPIn $\xrightarrow{DP\_IN}$ inPacket() $\xrightarrow{API\_READ}$ provider.host.InternalHostProvider $\xrightarrow{API\_WRITE}$ hostDetected() $\xrightarrow{HOST\_MOVED}$ mobility.InternalHostListener $\xrightarrow{API\_WRITE}$ removeFlowRules() $\xrightarrow{DP\_OUT}$ DPOut
7	CVE-2019-16299	mobility	HOST_REMOVED	
8	CVE-2019-16299	mobility	HOST_UPDATED	
9	CVE-2019-16301	vtn	HOST_MOVED	DPIn $\xrightarrow{DP\_IN}$ inPacket() $\xrightarrow{API\_READ}$ provider.host.InternalHostProvider $\xrightarrow{API\_WRITE}$ hostDetected() $\xrightarrow{HOST\_ADDED, HOST\_UPDATED, \text{ or } HOST\_REMOVED}$ vtn.InternalHostListener $\xrightarrow{API\_WRITE}$ forward() $\xrightarrow{DP\_OUT}$ DPOut
10	CVE-2019-16302	evpnopenflow	HOST_MOVED	DPIn $\xrightarrow{DP\_IN}$ inPacket() $\xrightarrow{API\_READ}$ provider.host.InternalHostProvider $\xrightarrow{API\_WRITE}$ hostDetected() $\xrightarrow{HOST\_ADDED \text{ or } HOST\_REMOVED}$ evpnopenflow.InternalHostListener $\xrightarrow{API\_WRITE}$ forward() $\xrightarrow{DP\_OUT}$ DPOut
11	CVE-2019-16302	evpnopenflow	HOST_UPDATED	
12	CVE-2019-16297	p4tutorial	HOST_MOVED	DPIn $\xrightarrow{DP\_IN}$ inPacket() $\xrightarrow{API\_READ}$ provider.host.InternalHostProvider $\xrightarrow{API\_WRITE}$ hostDetected() $\xrightarrow{HOST\_ADDED}$ p4tutorial.InternalHostListener $\xrightarrow{API\_WRITE}$ applyFlowRules() $\xrightarrow{DP\_OUT}$ DPOut
13	CVE-2019-16297	p4tutorial	HOST_REMOVED	
14	CVE-2019-16297	p4tutorial	HOST_UPDATED	

# CVE-2019-11189

Firewall app installs flow deny rule to prevent host from communicating

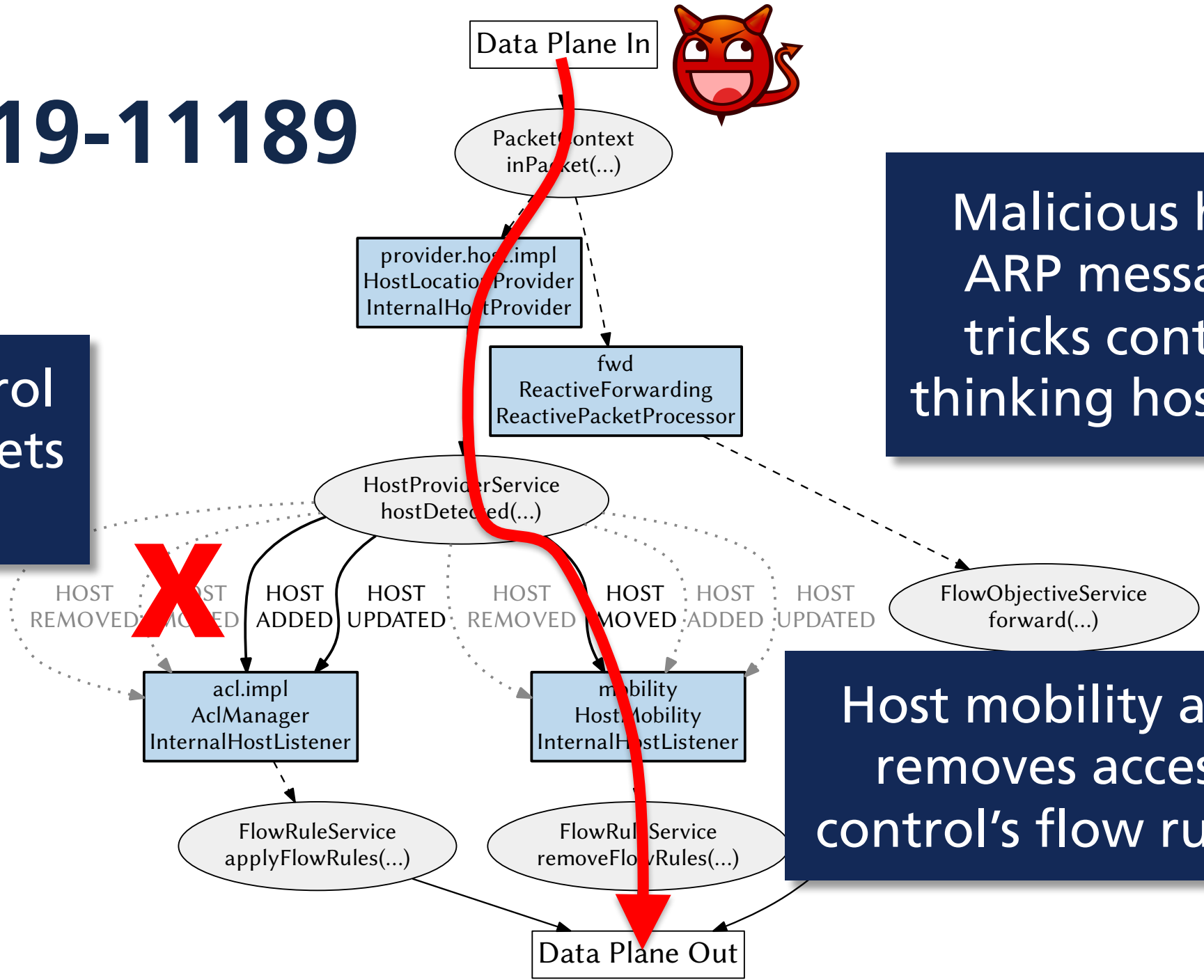


# CVE-2019-11189



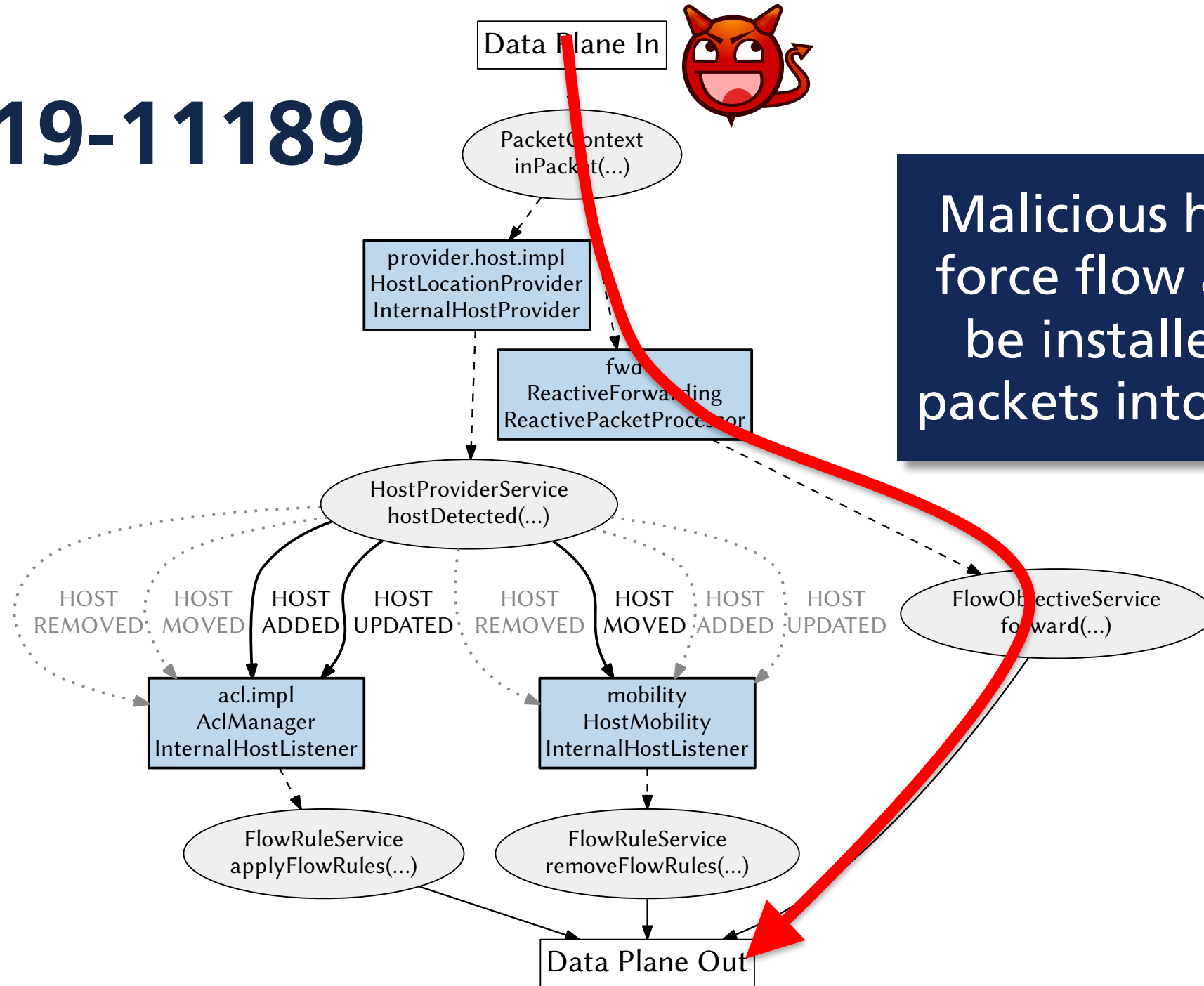
Access control app never gets called!

Malicious host spoofs ARP message, which tricks controller into thinking host has moved



Host mobility app removes access control's flow rules!

# CVE-2019-11189



Malicious host can now force flow allow rule to be installed and send packets into the network

# Conclusions

- Considered the **cross-plane event-based vulnerability** problem in SDN
- Design takeaways
  - **Hosts** have **outsized effect** on SDN operation
  - Security analysis must consider **all apps working together**
  - Developers must **design defensively**
- Discovered and validated **14 new vulnerabilities** in ONOS SDN controller



# Questions?

Thank you for your time!

**Benjamin E. Ujcich**

E-mail: [ujcich2@illinois.edu](mailto:ujcich2@illinois.edu)

Web: <http://ujcich2.web.engr.illinois.edu>



*This work was supported in part by NSF Grant Nos. CNS-1657534 and CNS-1750024. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.*