HACKING THE BRAIN

Customize Evil Protocol to Pwn an SDN Controller

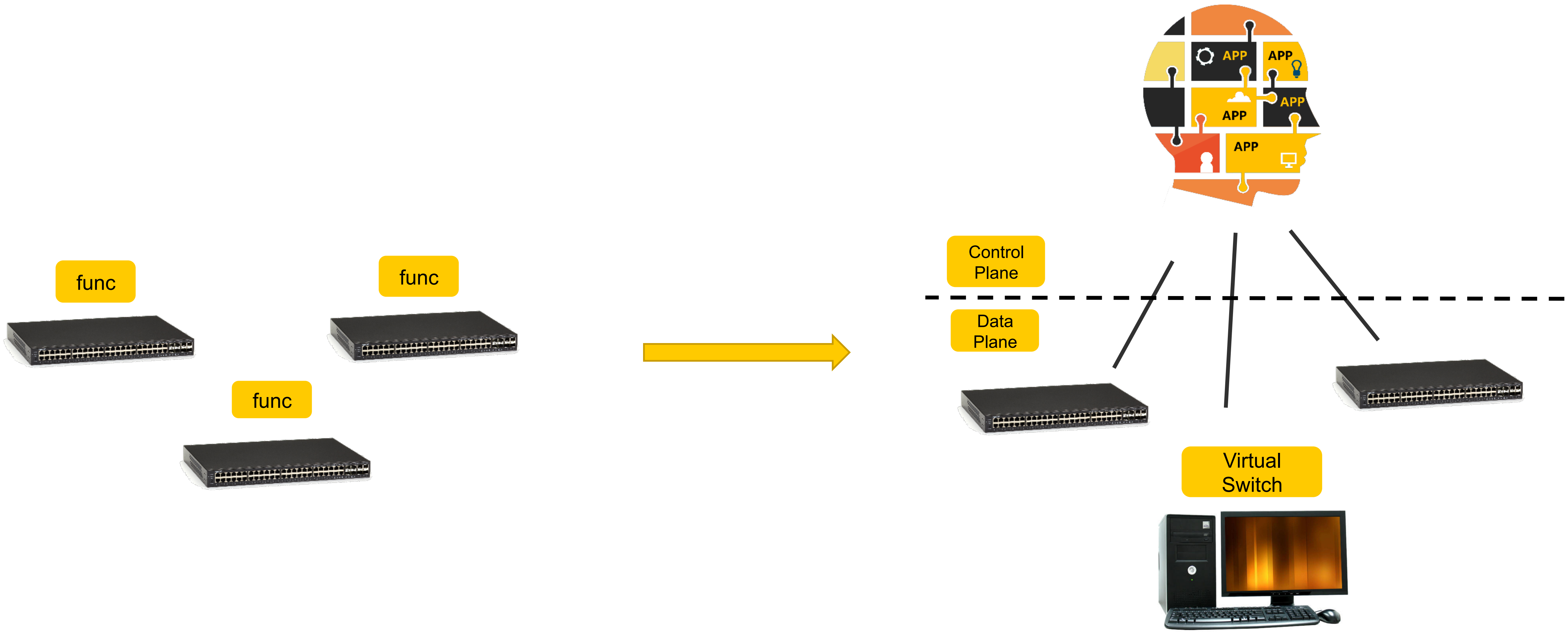
Feng Xiao, Ph.D. student at PennState, Cyber Security Lab

Jianwei Huang, Researcher at Wuhan University

Peng Liu, Professor at PennState, Cyber Security Lab

PENNSTATE

A Brief Introduction to SDN



Software-Defined Networking (SDN) is an emerging architecture that decouples the network control and forwarding functions.

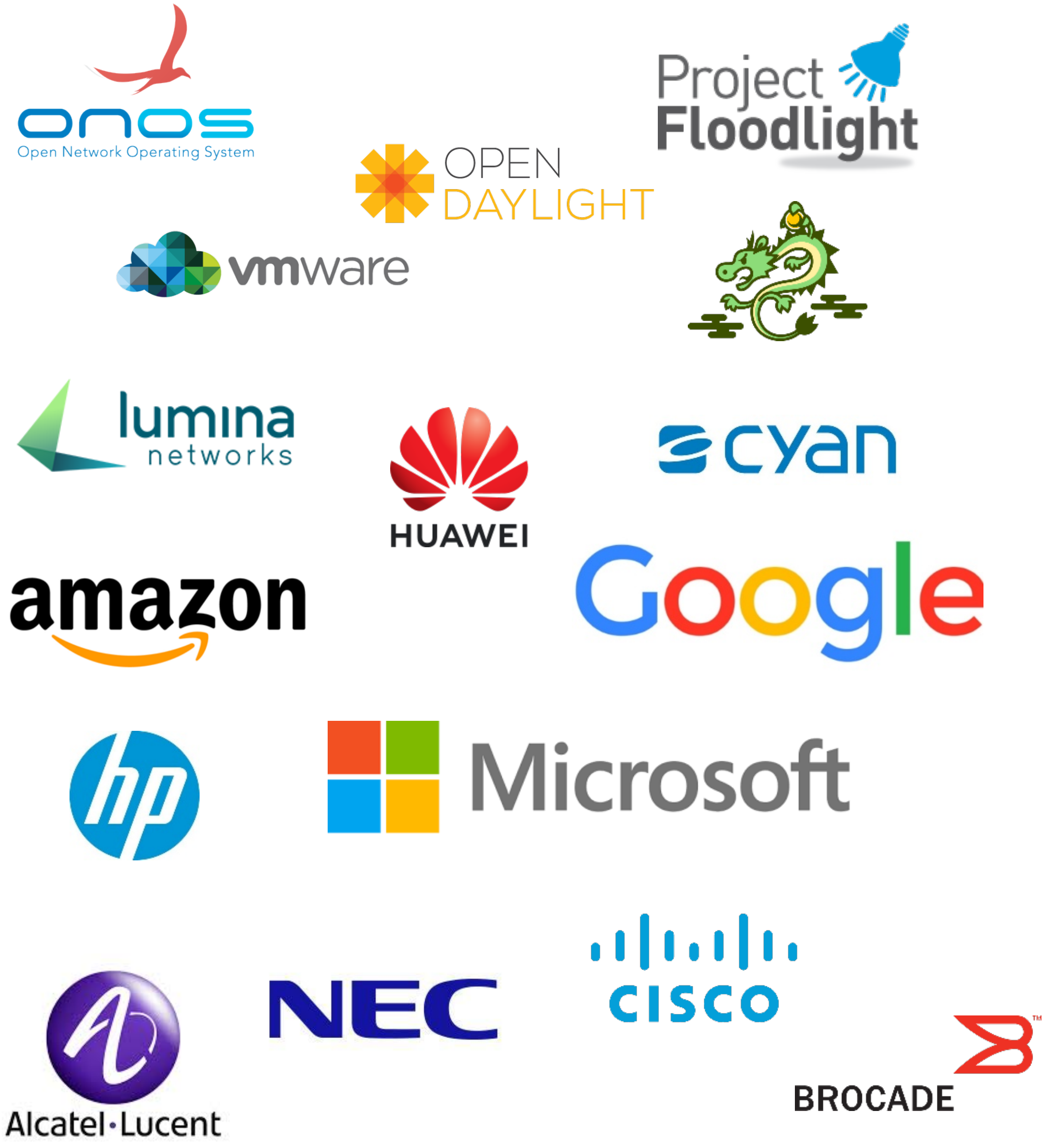
What’s SDN Like Today?

Who are contributing?

- More than 15 popular controllers.
- More than 1700 open source SDN projects.

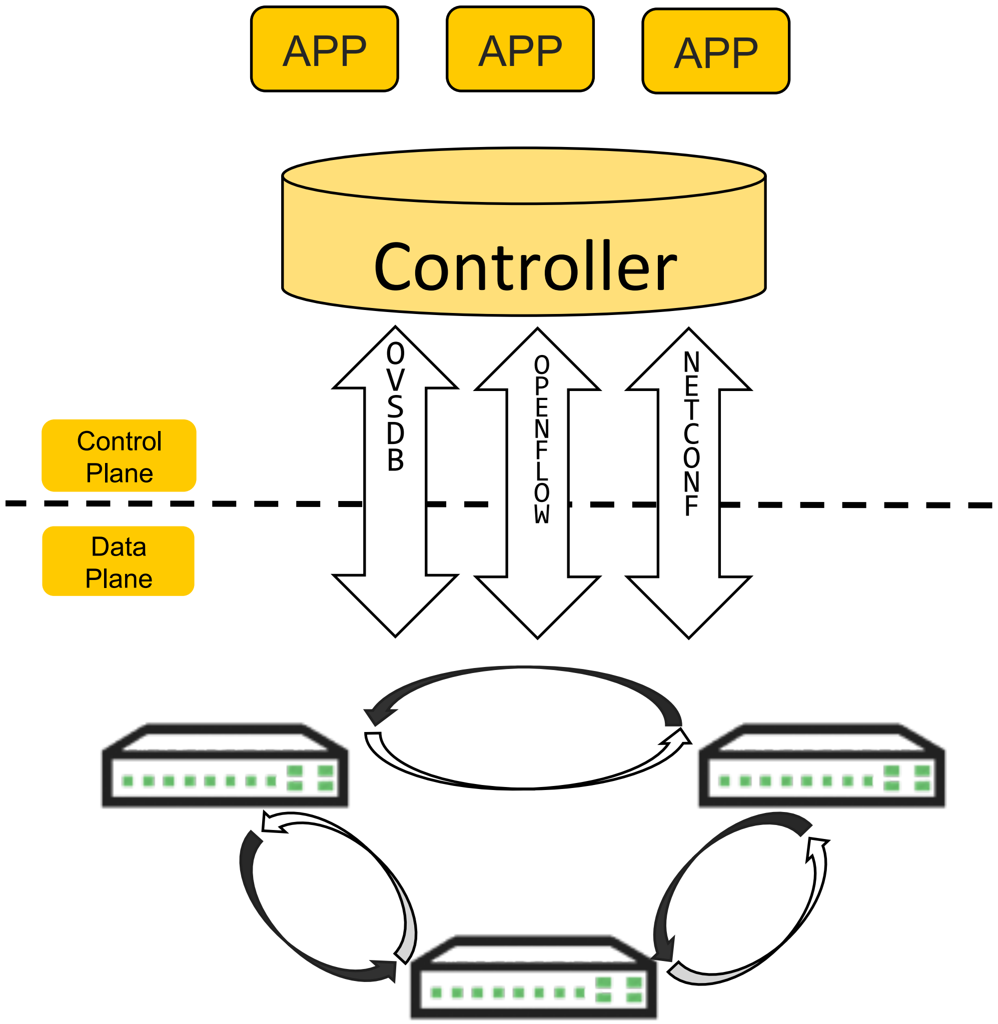
Who are using?

- Data Center
- Telecom
- Enterprise
- …



Attack Objectives in SDN

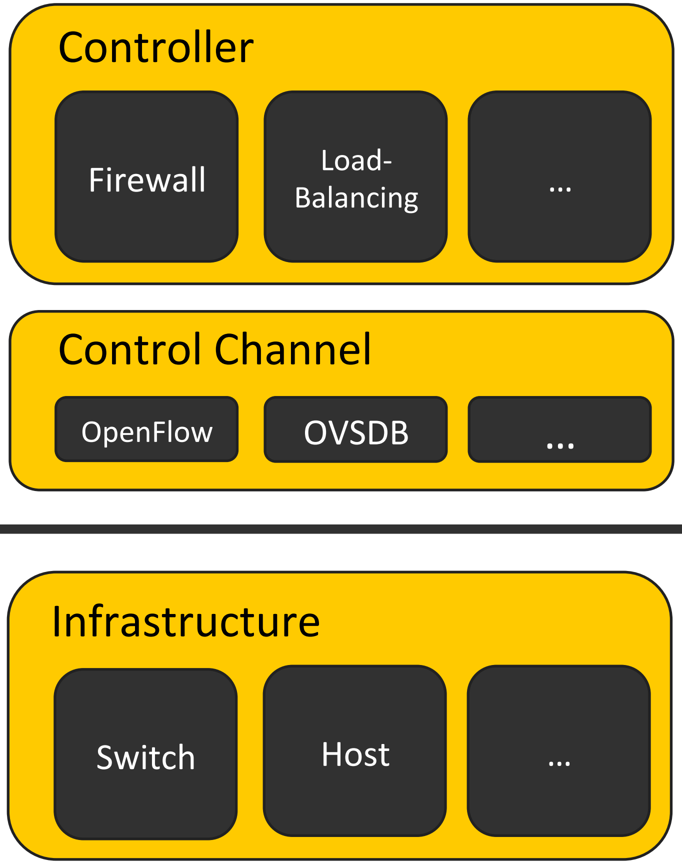
Objective	Reference	Category
Congest control channel	Control plane saturation attack	Denial of Service
Terminate/Disrupt network services	State manipulation attack	Denial of Service
Steal confidential configuration	New	Data leakage
Probe network information	New	Data leakage
Install flow rules	New	Network manipulation
Fabricate links or hosts	Topology poisoning attack	Network manipulation
Distort network service results	New	Network manipulation
Disconnect network elements	New	Network manipulation
Install malicious SDN applications	New	Network manipulation



Pwn It Like A Hacker

Software-Defined Networks

Decoupled Control Plane and Data Plane



Controller

Firewall

Load-Balancing

...

Control Channel

OpenFlow

OVSDB

...

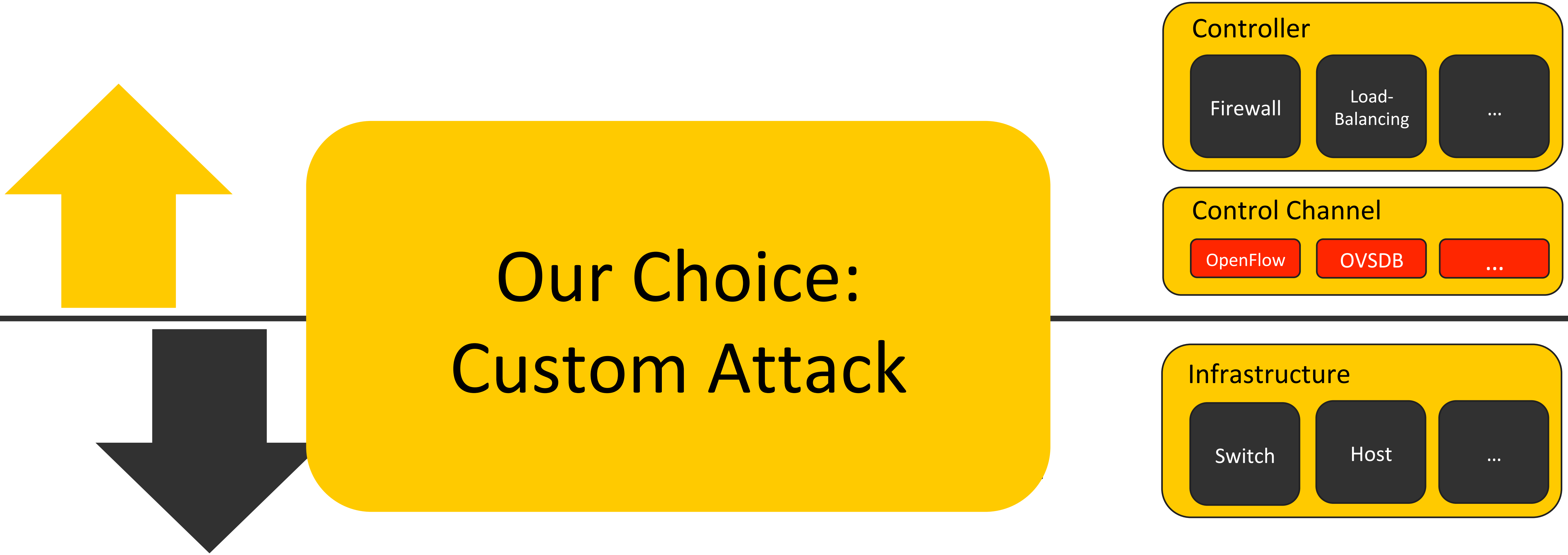
Infrastructure

Switch

Host

...

Pwn It Like A Hacker

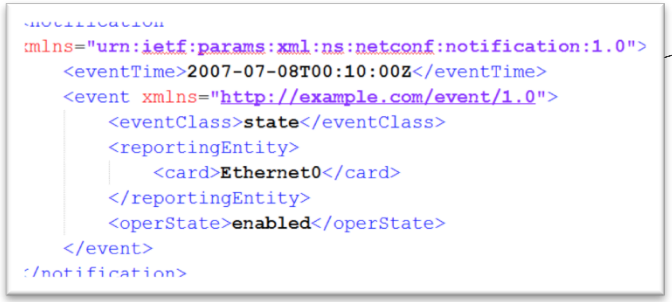


Our Choice:
Custom Attack

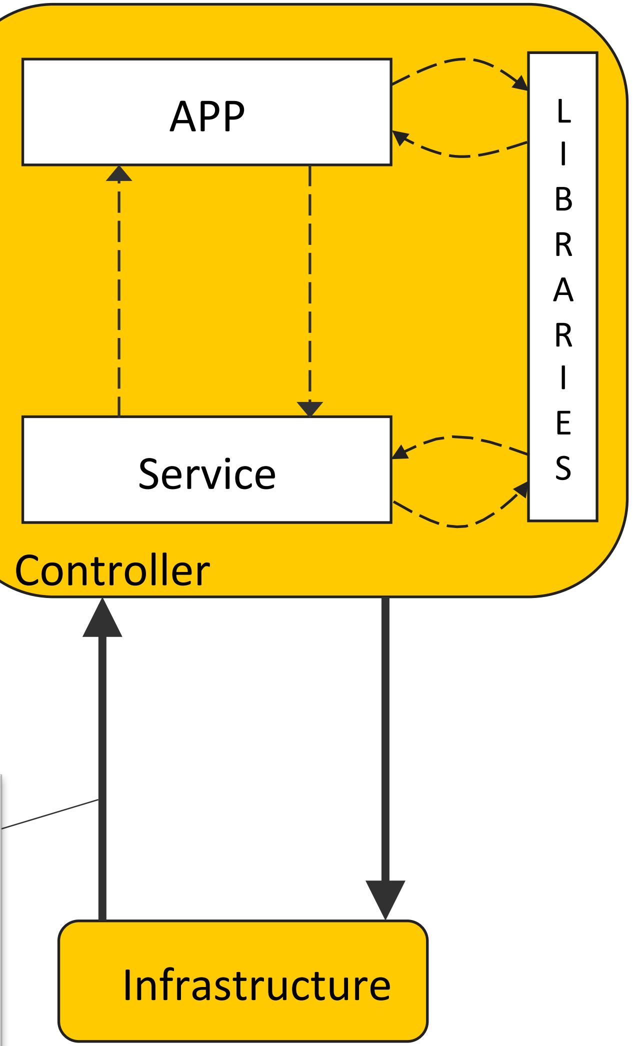
Custom Attack

Custom Field (CF) in legitimate protocol interactions

- CF is controlled by data plane (hacker)
- CF will be processed by components in the controller



```
<notification
xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2007-07-08T00:10:00Z</eventTime>
    <event xmlns="http://example.com/event/1.0">
        <eventClass>state</eventClass>
        <reportingEntity>
            <card>Ethernet0</card>
        </reportingEntity>
        <operState>enabled</operState>
    </event>
</notification>
```

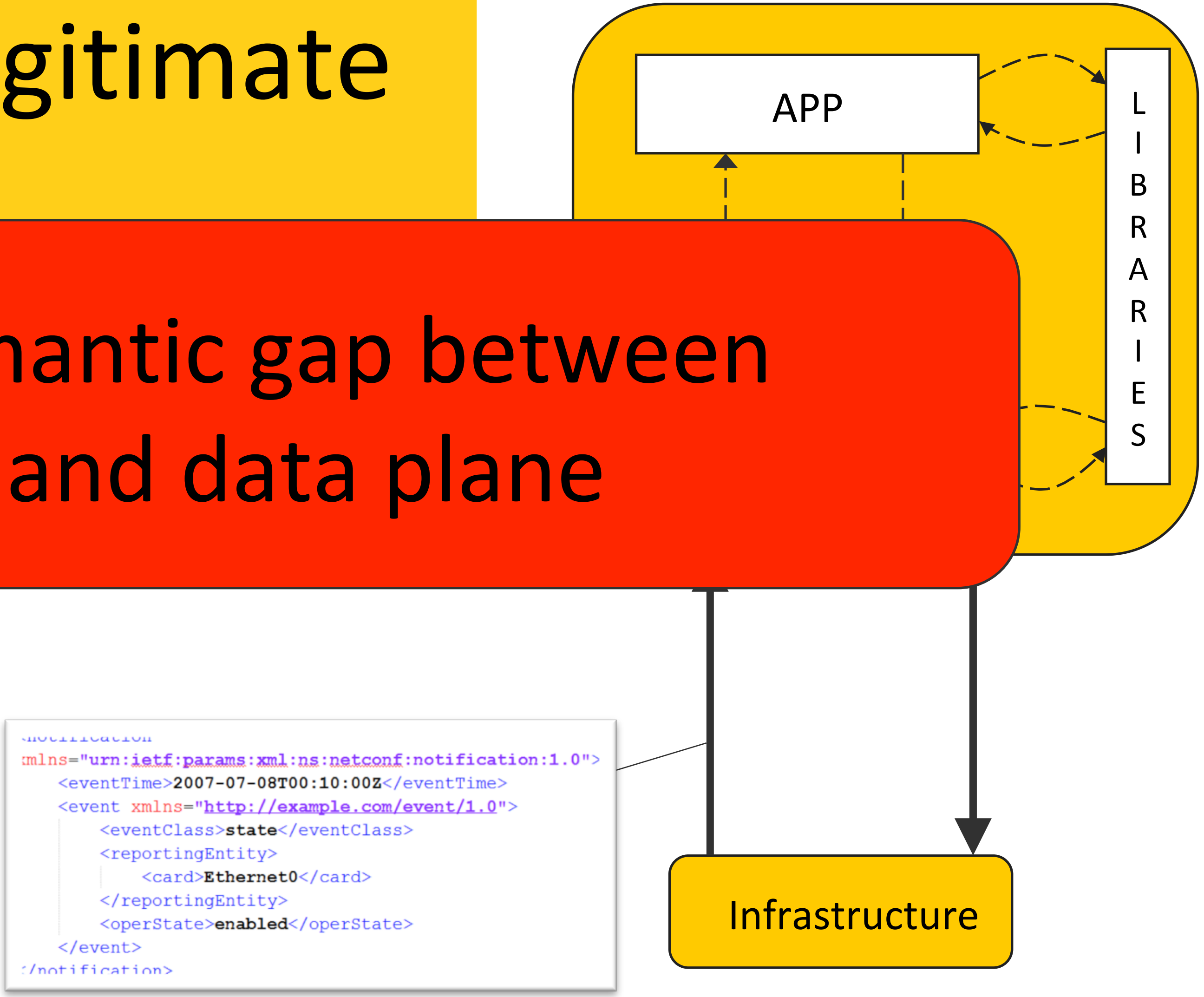


Custom Attack

Custom Field (CF) in legitimate protocol interactions

CF results in a semantic gap between control plane and data plane

- C
- C
  in the controller



```
<notification
xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2007-07-08T00:10:00Z</eventTime>
    <event xmlns="http://example.com/event/1.0">
        <eventClass>state</eventClass>
        <reportingEntity>
            <card>Ethernet0</card>
        </reportingEntity>
        <operState>enabled</operState>
    </event>
</notification>
```

What Can It Cause?

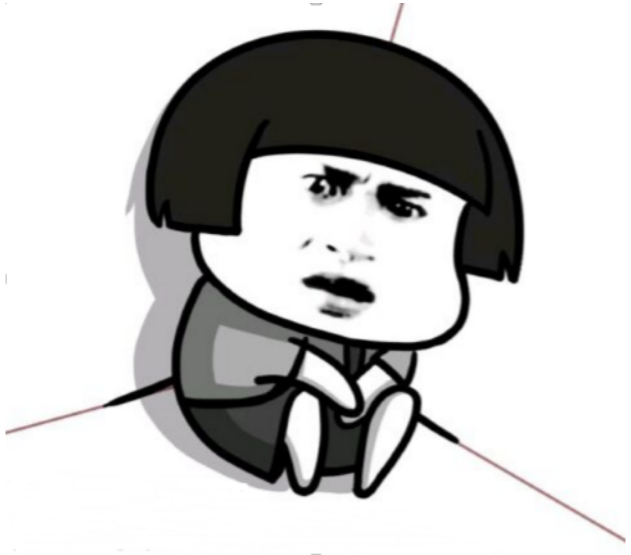
Execute Arbitray SDN Commands

Steal Confidential Data

Crash/Disrupt Service

Disable Network Function

...



Threat Model

We do NOT assume that hackers can have network access to SDN controllers or SDN applications

Control channel is well protected by SSL/TLS

Threat Model

We do NOT assume that hackers can have network access to SDN controllers or SDN applications

Control channel is well protected by SSL/TLS

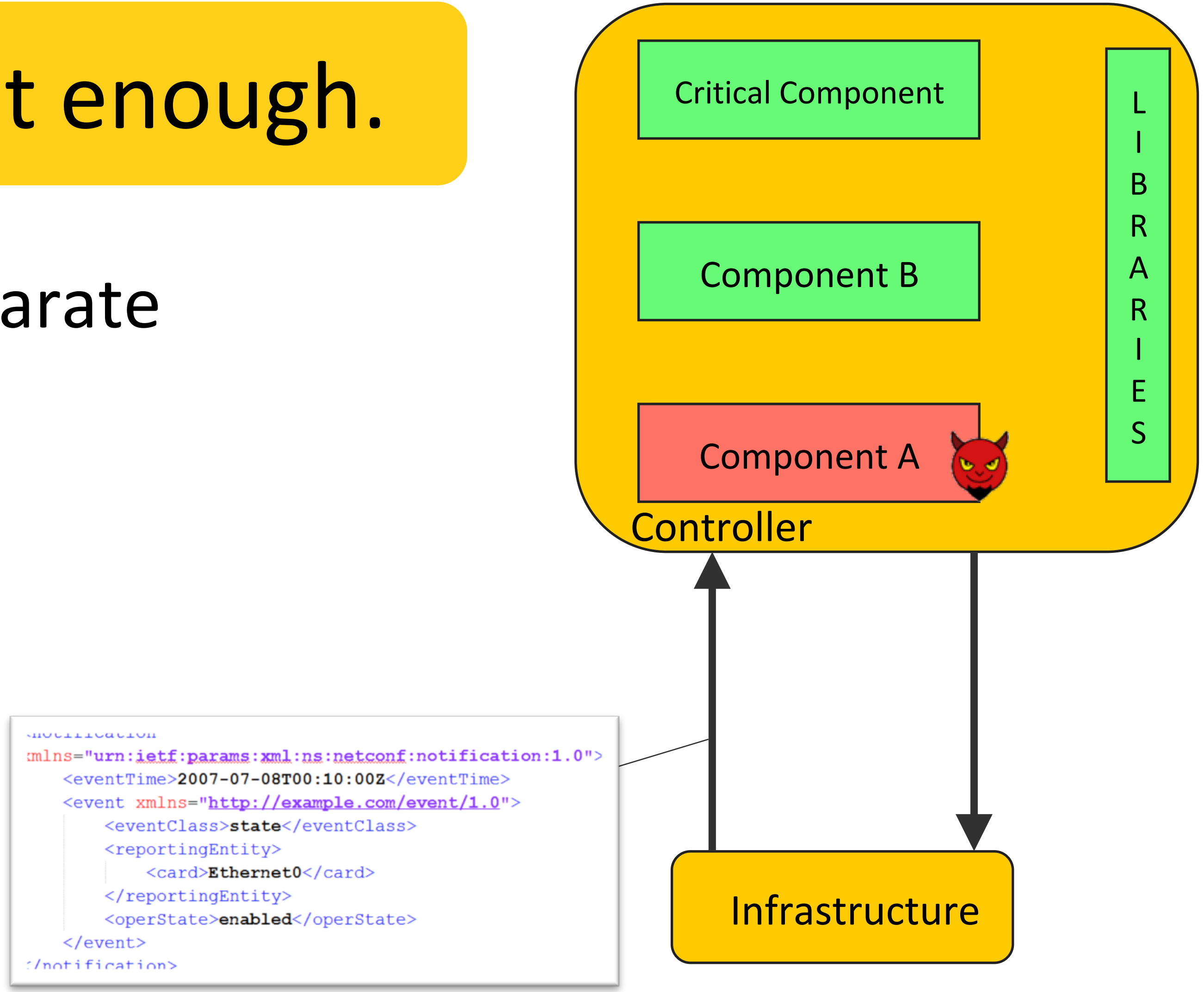
A compromised host[1] or switch[2]

[1] exploitable if the target network is configured with in-band control.
[2] Switches are vulnerable to multiple remote attacks (e.g., Buffer Overflow[CVE-2016-2074]).

Challenges

Abusing Custom Field is not enough.

- Every Component runs in its separate context.

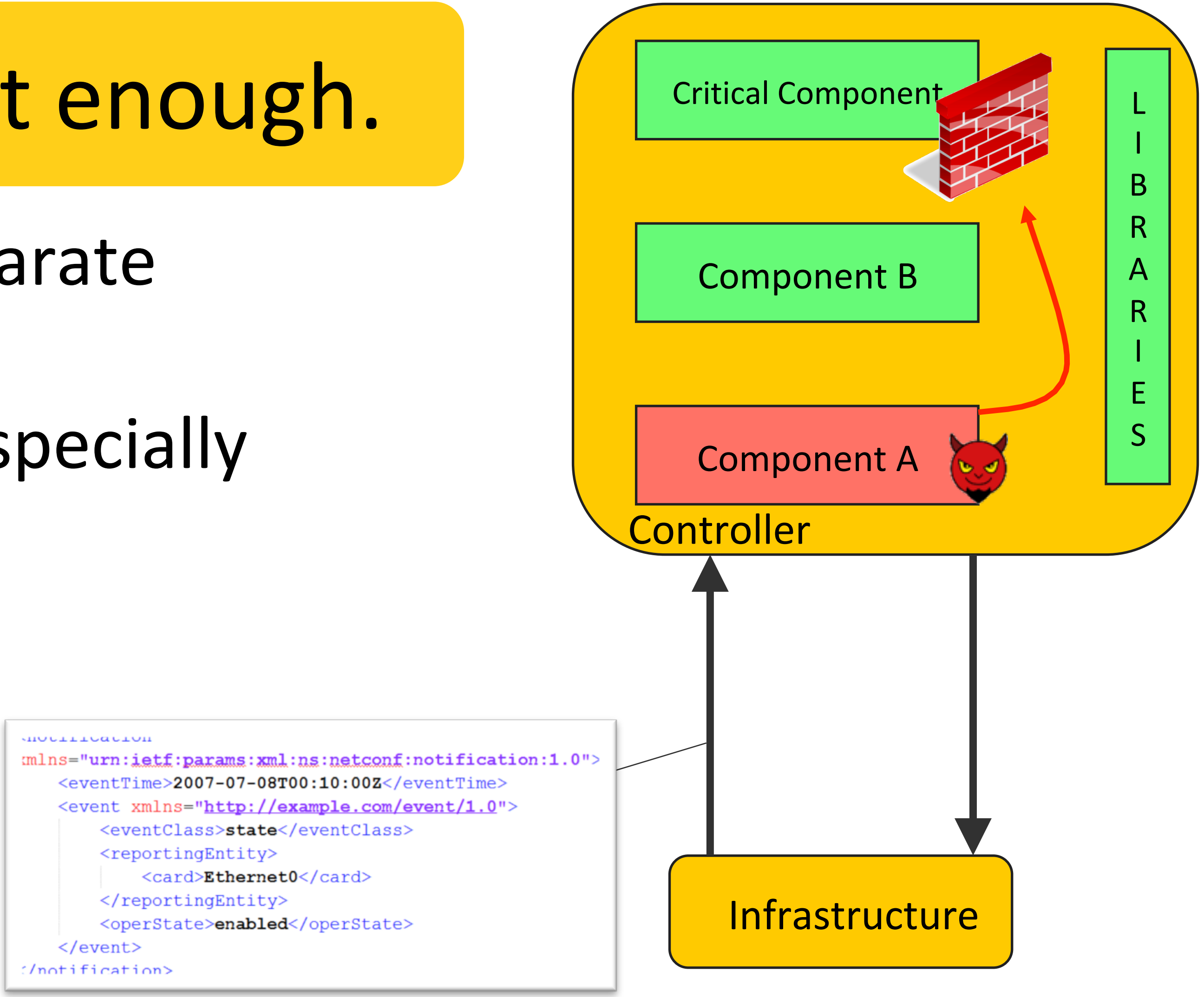


```
<notification
xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2007-07-08T00:10:00Z</eventTime>
    <event xmlns="http://example.com/event/1.0">
        <eventClass>state</eventClass>
        <reportingEntity>
            <card>Ethernet0</card>
        </reportingEntity>
        <operState>enabled</operState>
    </event>
</notification>
```

Challenges

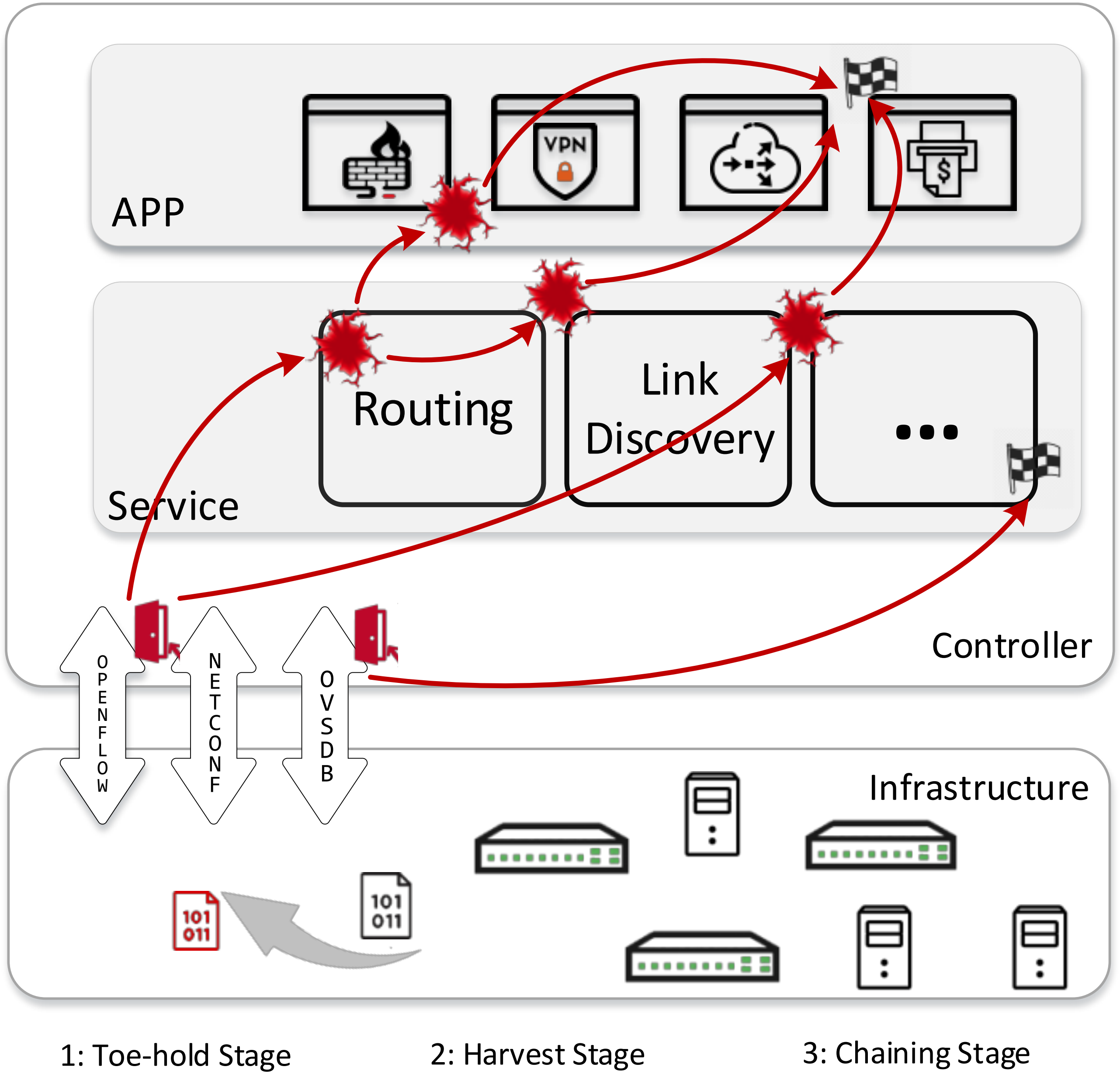
Abusing Custom Field is not enough.

- Every Component runs in its separate context.
- Critical components are usually specially protected.

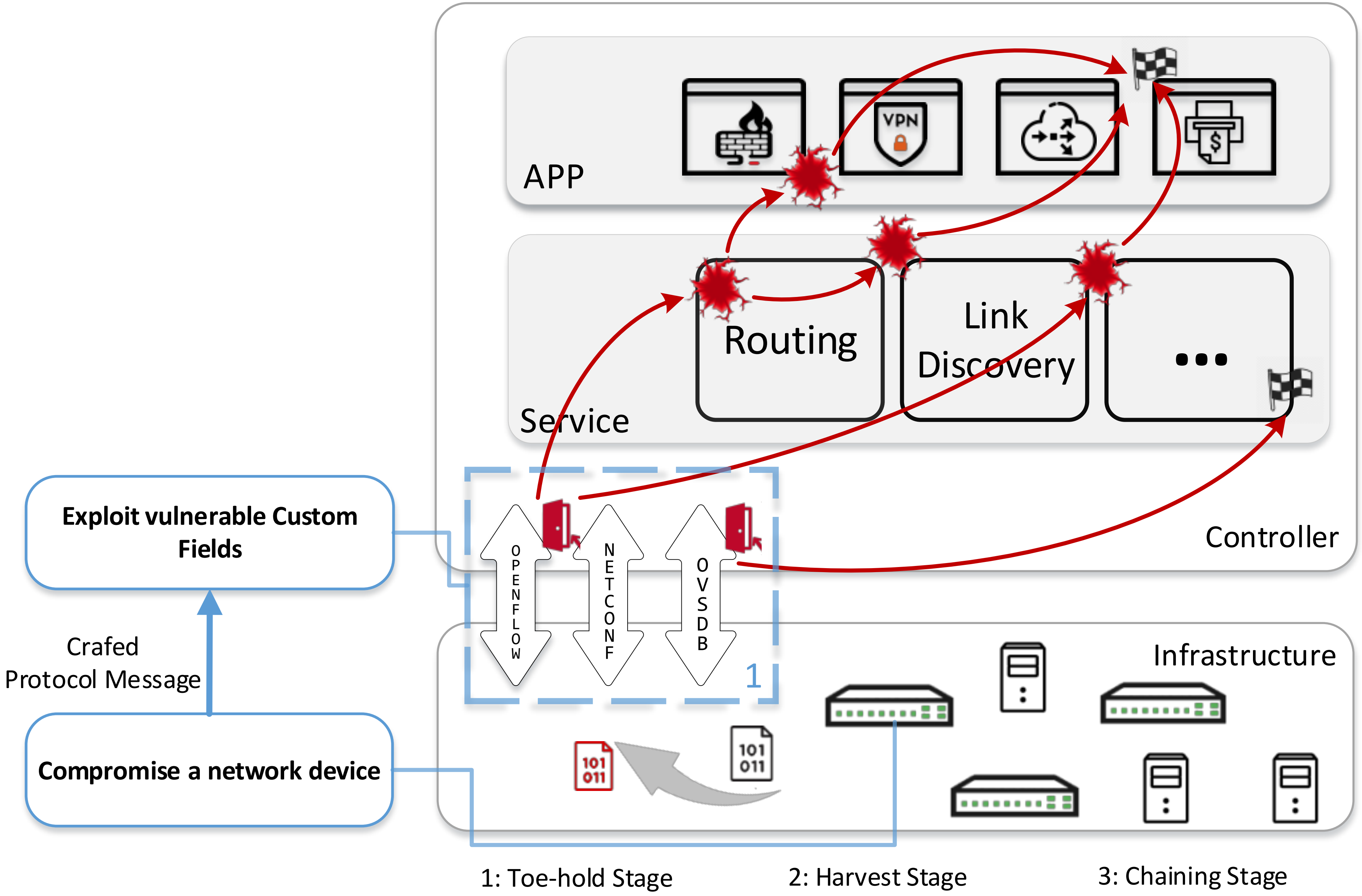


```
<notification
xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2007-07-08T00:10:00Z</eventTime>
    <event xmlns="http://example.com/event/1.0">
        <eventClass>state</eventClass>
        <reportingEntity>
            <card>Ethernet0</card>
        </reportingEntity>
        <operState>enabled</operState>
    </event>
</notification>
```

Multi-stage Exploitation



Multi-stage Exploitation

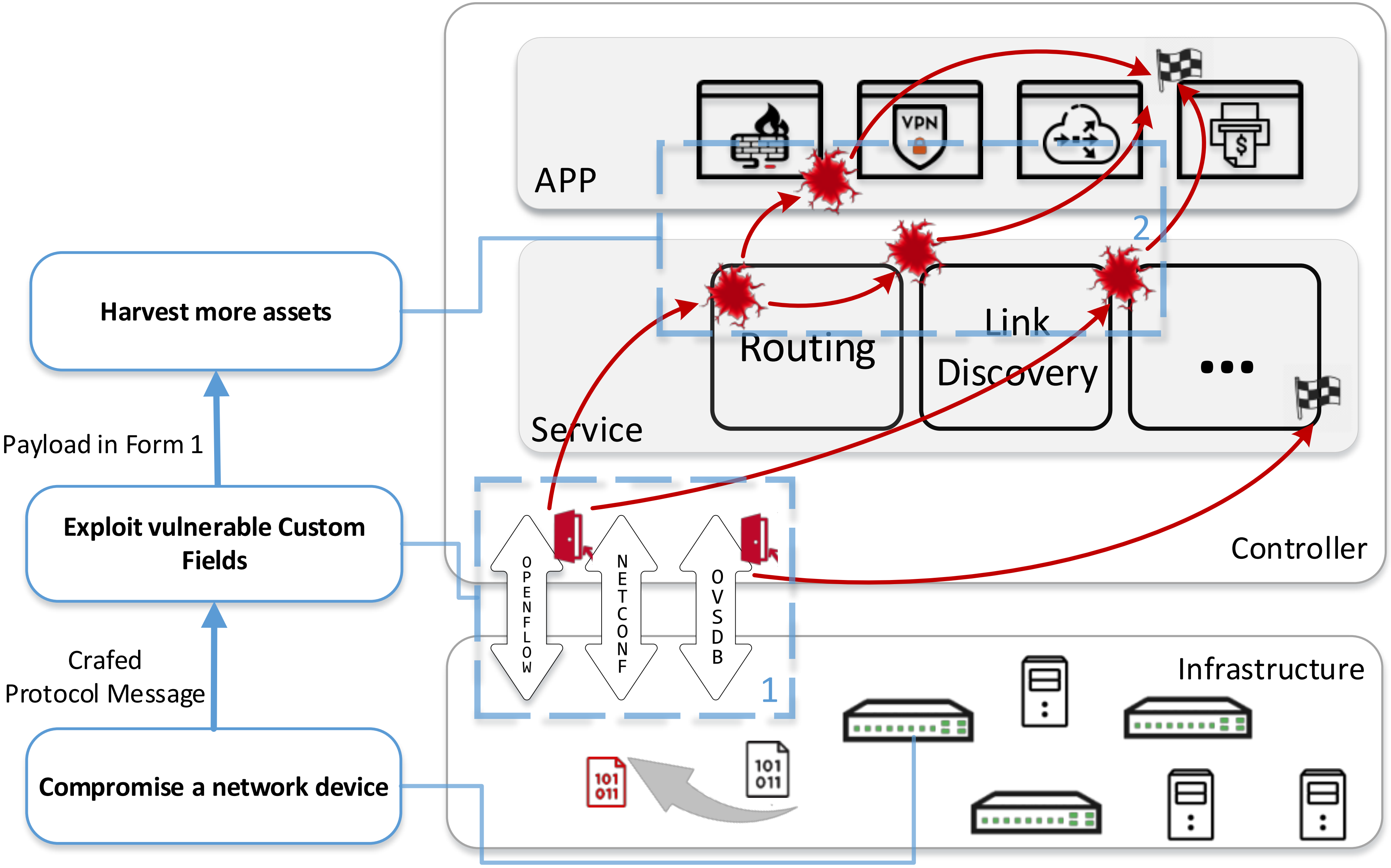


Exploit vulnerable Custom Fields

Crafed Protocol Message

Compromise a network device

Multi-stage Exploitation



Harvest more assets

Payload in Form 1

Exploit vulnerable Custom Fields

Crafted Protocol Message

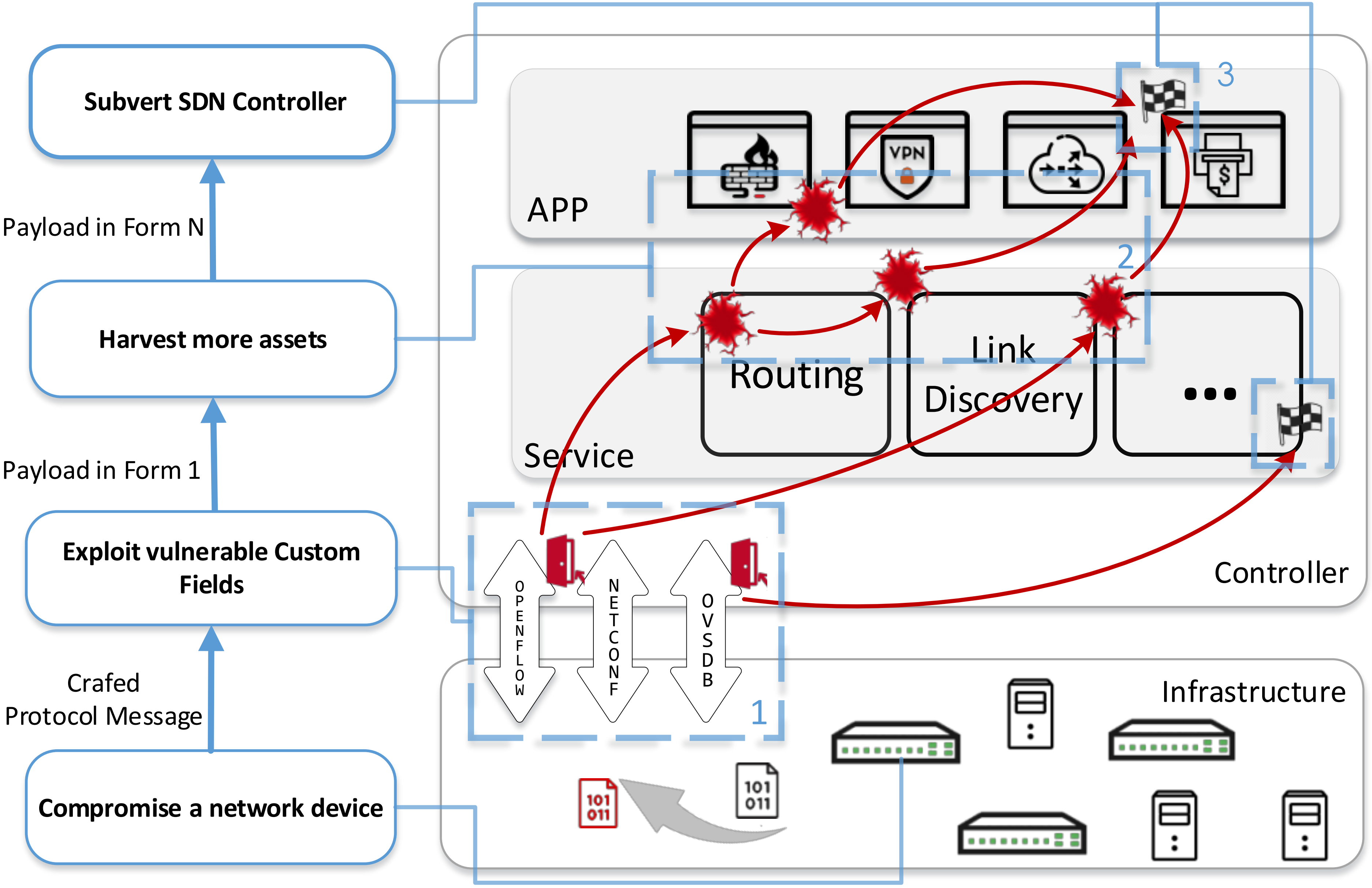
Compromise a network device

1: Toe-hold Stage

2: Harvest Stage

3: Chaining Stage

Multi-stage Exploitation



Subvert SDN Controller

Payload in Form N

Harvest more assets

Payload in Form 1

Exploit vulnerable Custom Fields

Crafed Protocol Message

Compromise a network device

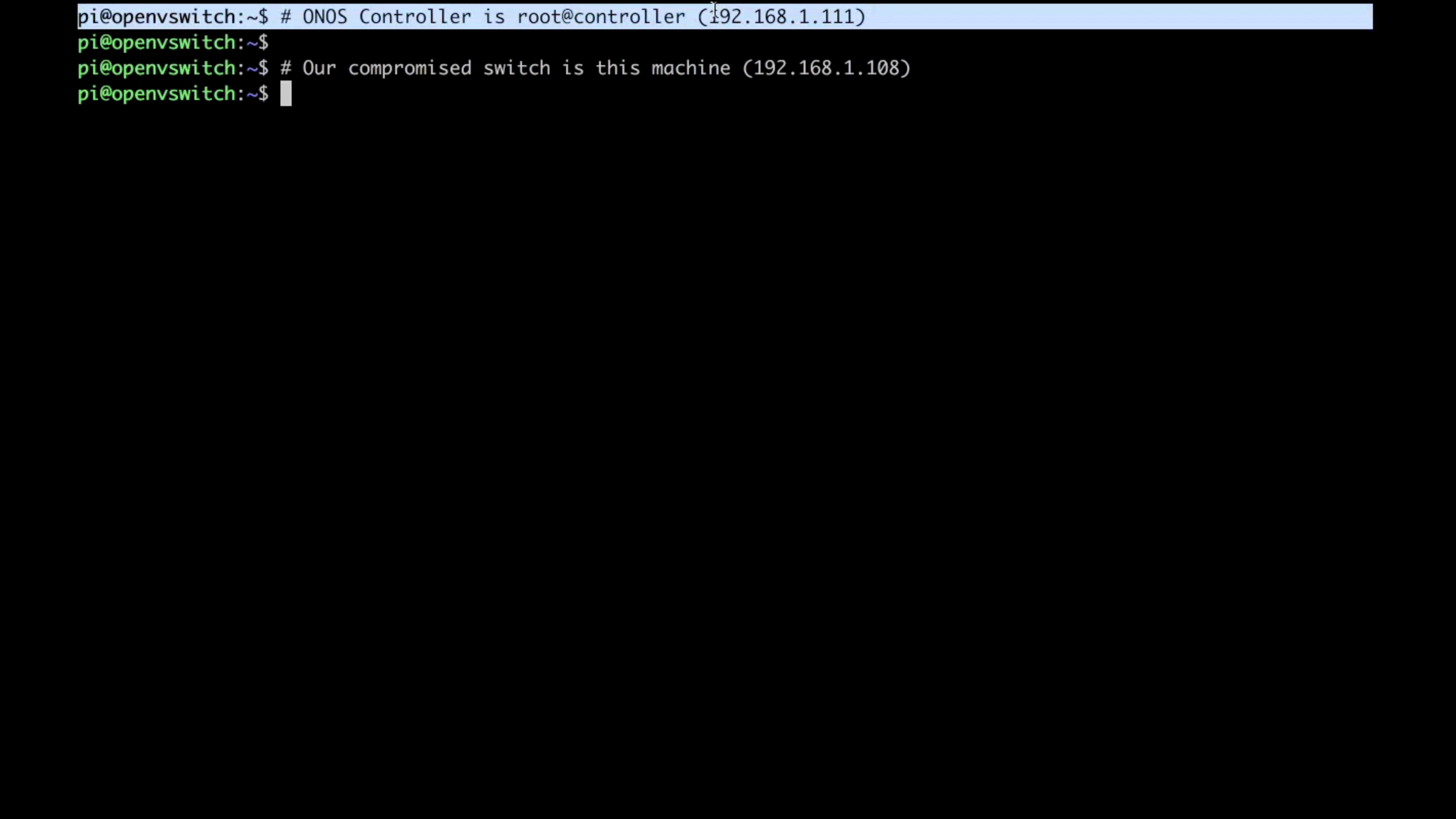
1: Toe-hold Stage

2: Harvest Stage

3: Chaining Stage

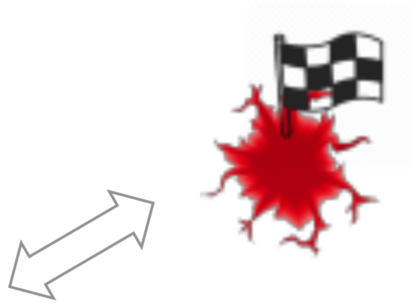
Hack Something Real!

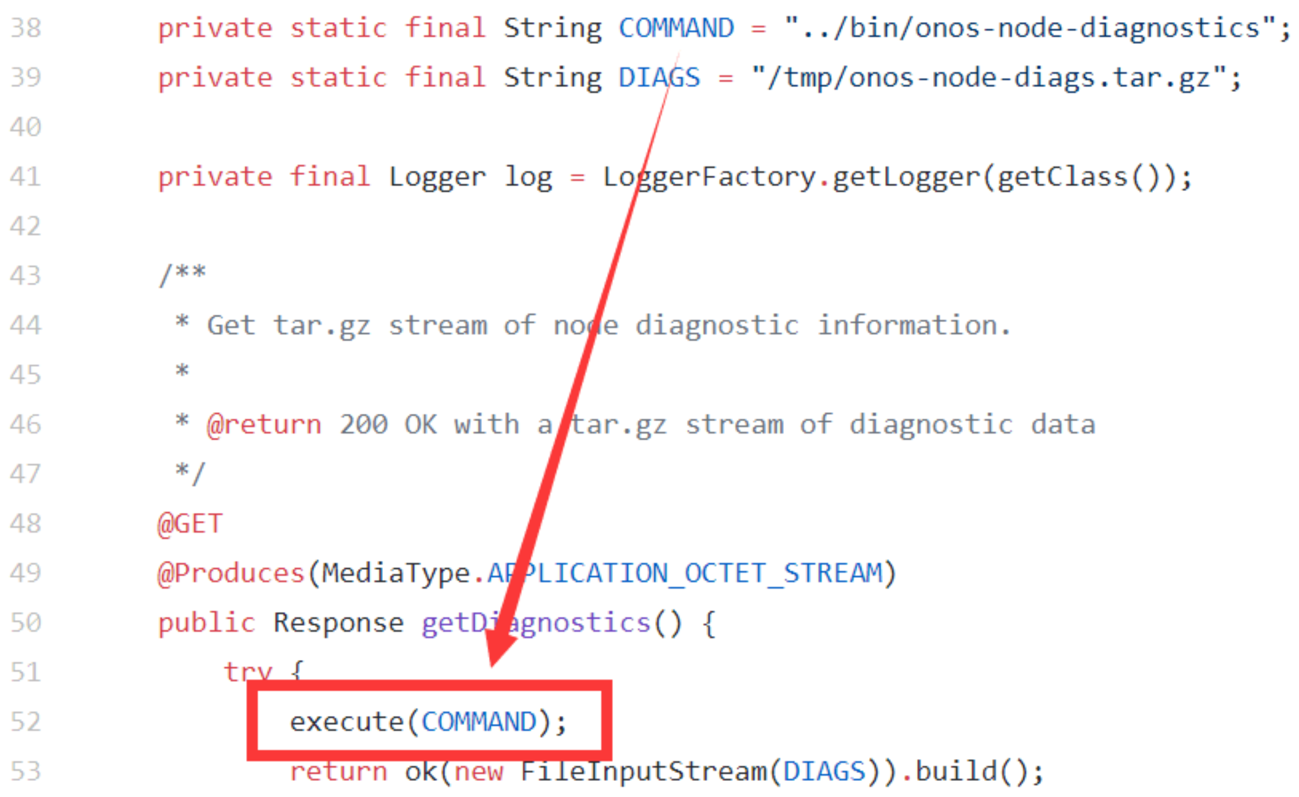
ONOS Remote Command Execution



```
pi@openvswitch:~$ # ONOS Controller is root@controller (192.168.1.111)
pi@openvswitch:~$
pi@openvswitch:~$ # Our compromised switch is this machine (192.168.1.108)
pi@openvswitch:~$
```

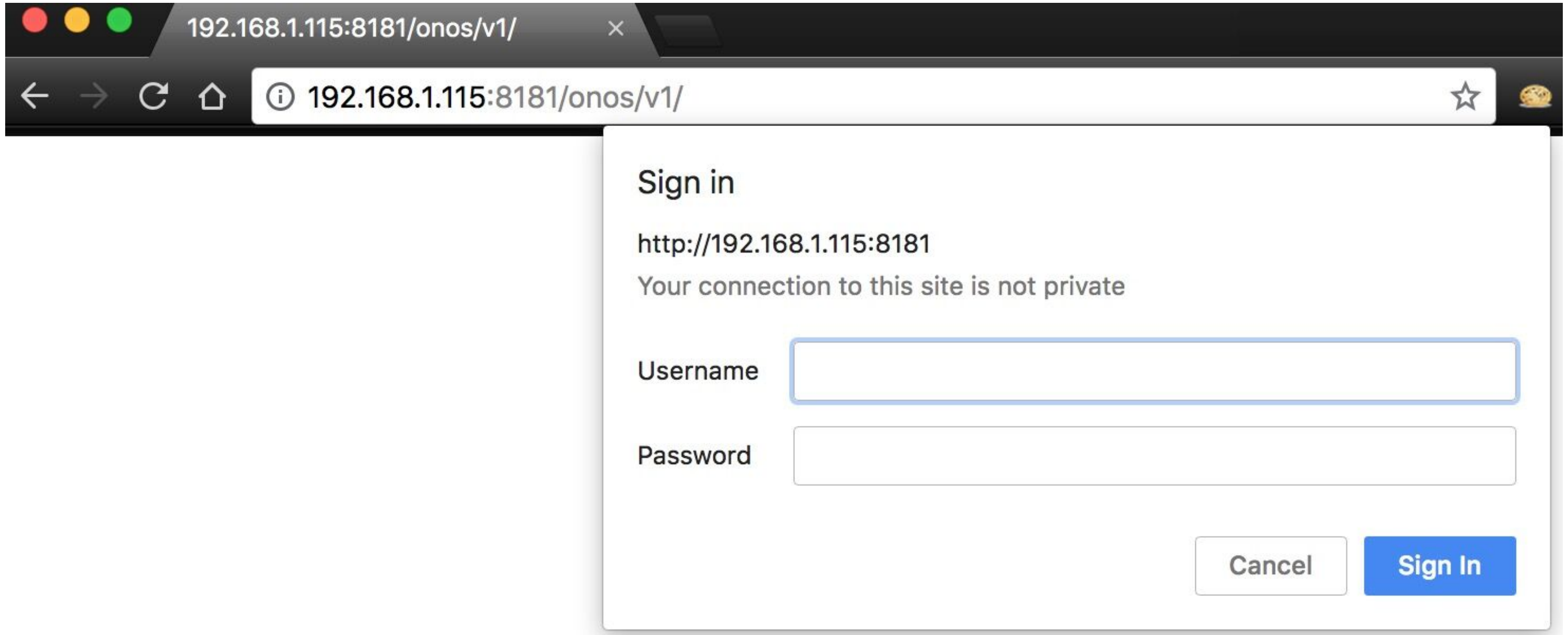
Hack Something Real!





```
    private static final String COMMAND = "../bin/onos-node-diagnostics";
    private static final String DIAGS = "/tmp/onos-node-diags.tar.gz";

    private final Logger log = LoggerFactory.getLogger(getClass());

    /**
     * Get tar.gz stream of node diagnostic information.
     *
     * @return 200 OK with a tar.gz stream of diagnostic data
     */
    @GET
    @Produces(MediaType.APPLICATION_OCTET_STREAM)
    public Response getDiagnostics() {
        try {
            execute(COMMAND);
            return ok(new FileInputStream(DIAGS)).build();
```

Hack Something Real!



192.168.1.115:8181/onos/v1/

192.168.1.115:8181/onos/v1/

Sign in

http://192.168.1.115:8181

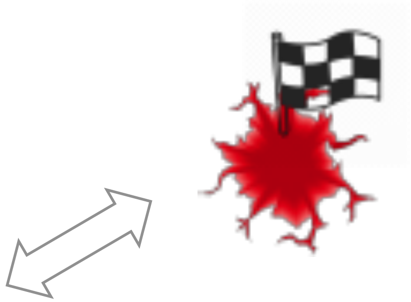
Your connection to this site is not private

Username

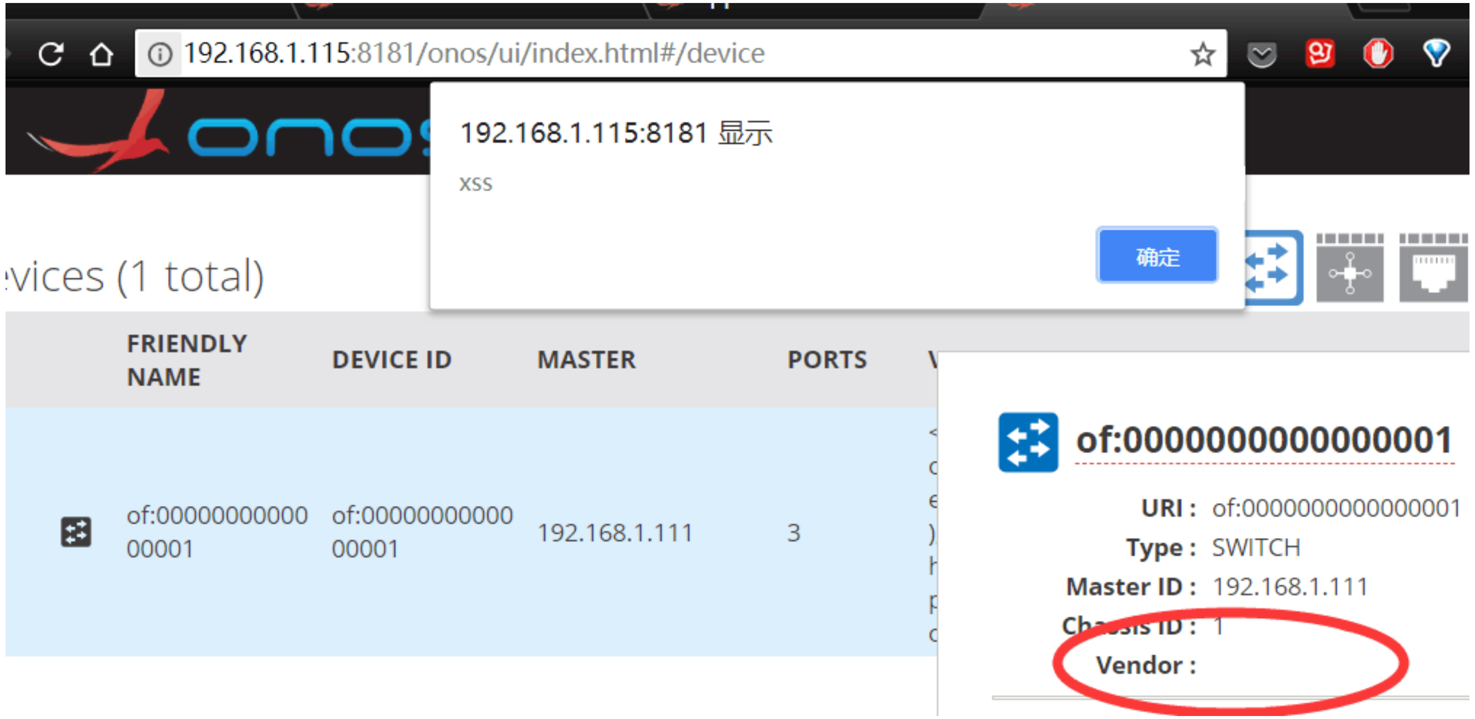
Password

Cancel

Sign In



Hack Something Real!



192.168.1.115:8181/onos/ui/index.html#/device

192.168.1.115:8181 显示

xss

确定

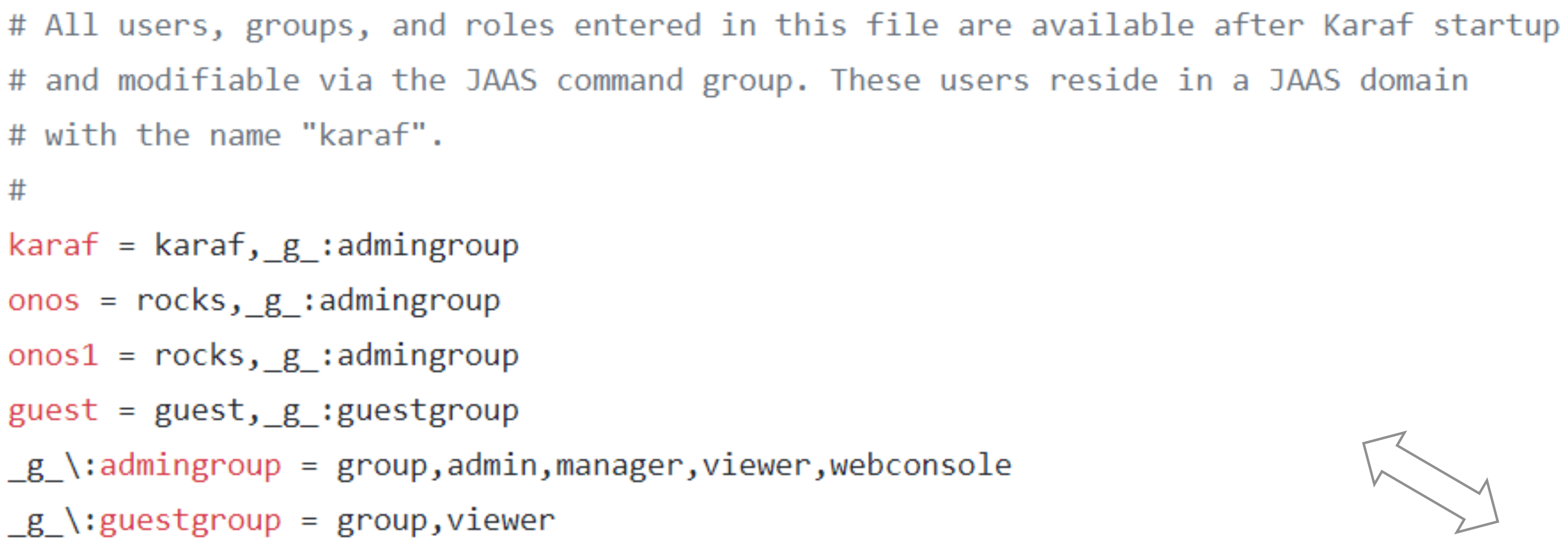
vices (1 total)

	FRIENDLY NAME	DEVICE ID	MASTER	PORTS	V
	of:0000000000000001	of:0000000000000001	192.168.1.111	3	

of:0000000000000001

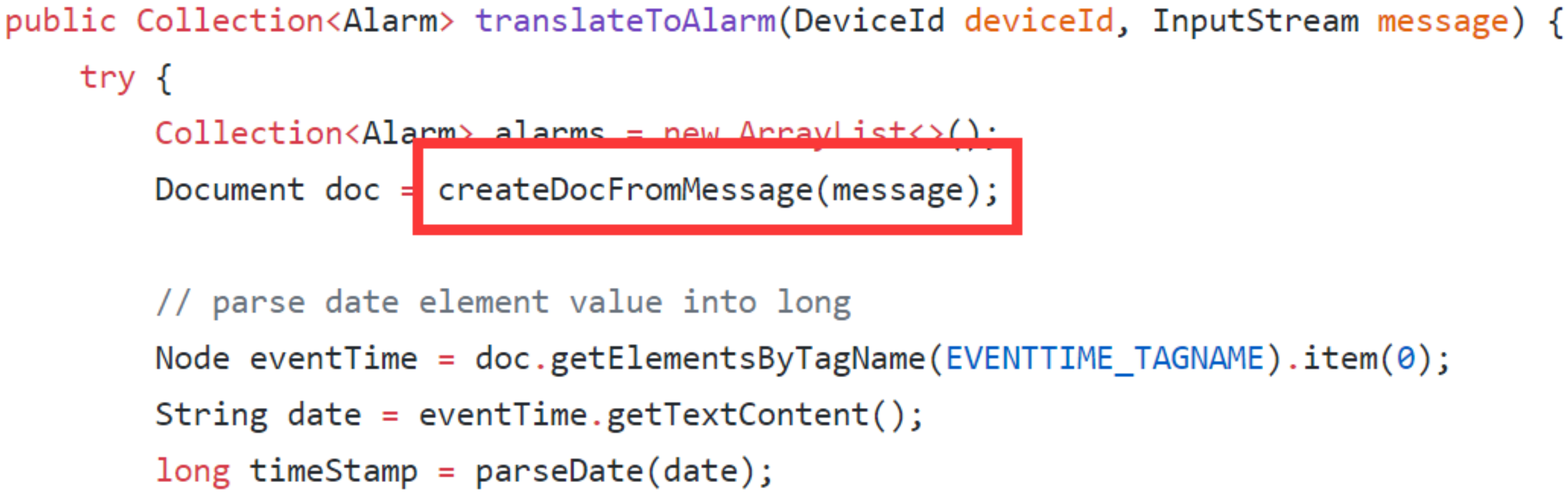
URI : of:0000000000000001
Type : SWITCH
Master ID : 192.168.1.111
Chassis ID : 1
Vendor :

Hack Something Real!



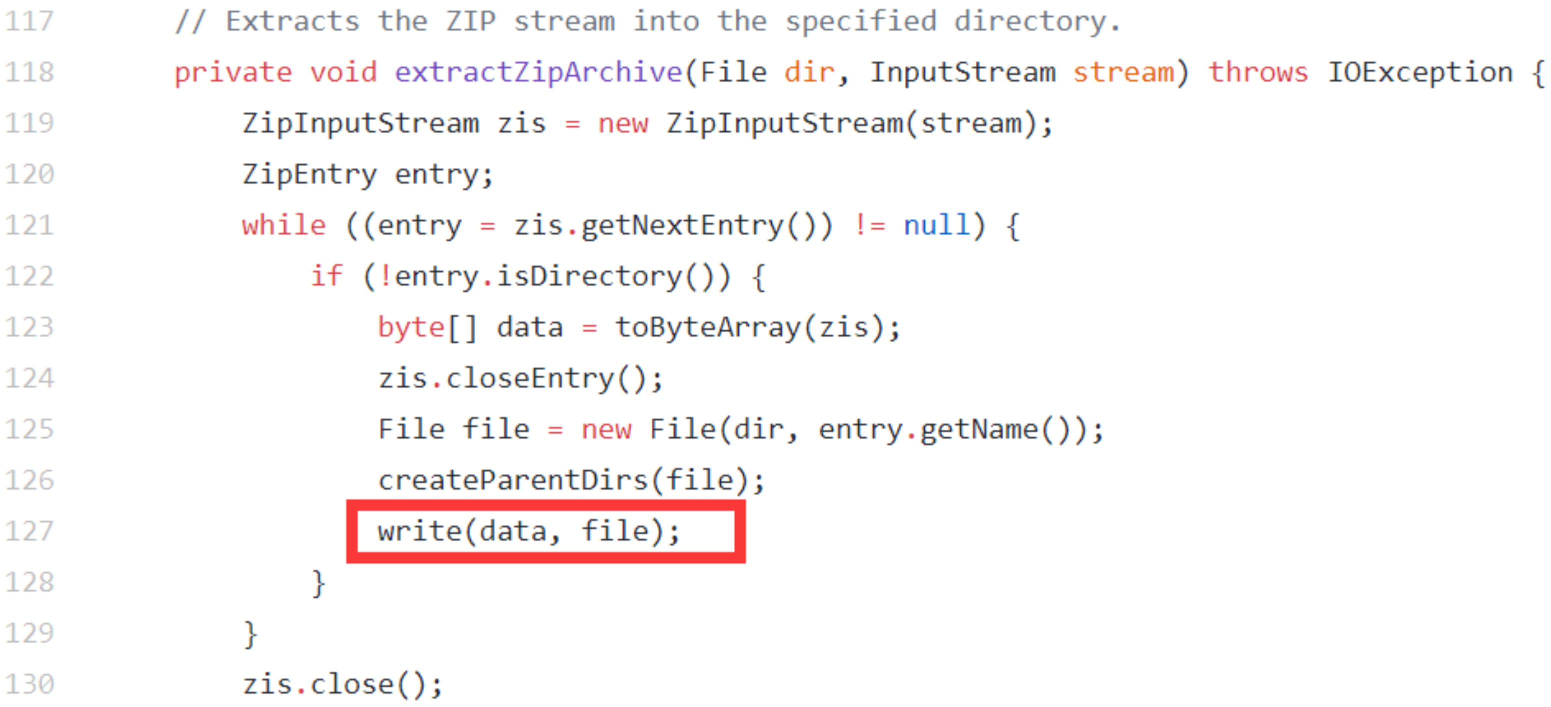
```
# All users, groups, and roles entered in this file are available after Karaf startup
# and modifiable via the JAAS command group. These users reside in a JAAS domain
# with the name "karaf".
#
karaf = karaf,_g_:admingroup
onos = rocks,_g_:admingroup
onos1 = rocks,_g_:admingroup
guest = guest,_g_:guestgroup
_g_\:admingroup = group,admin,manager,viewer,webconsole
_g_\:guestgroup = group,viewer
```

Hack Something Real!



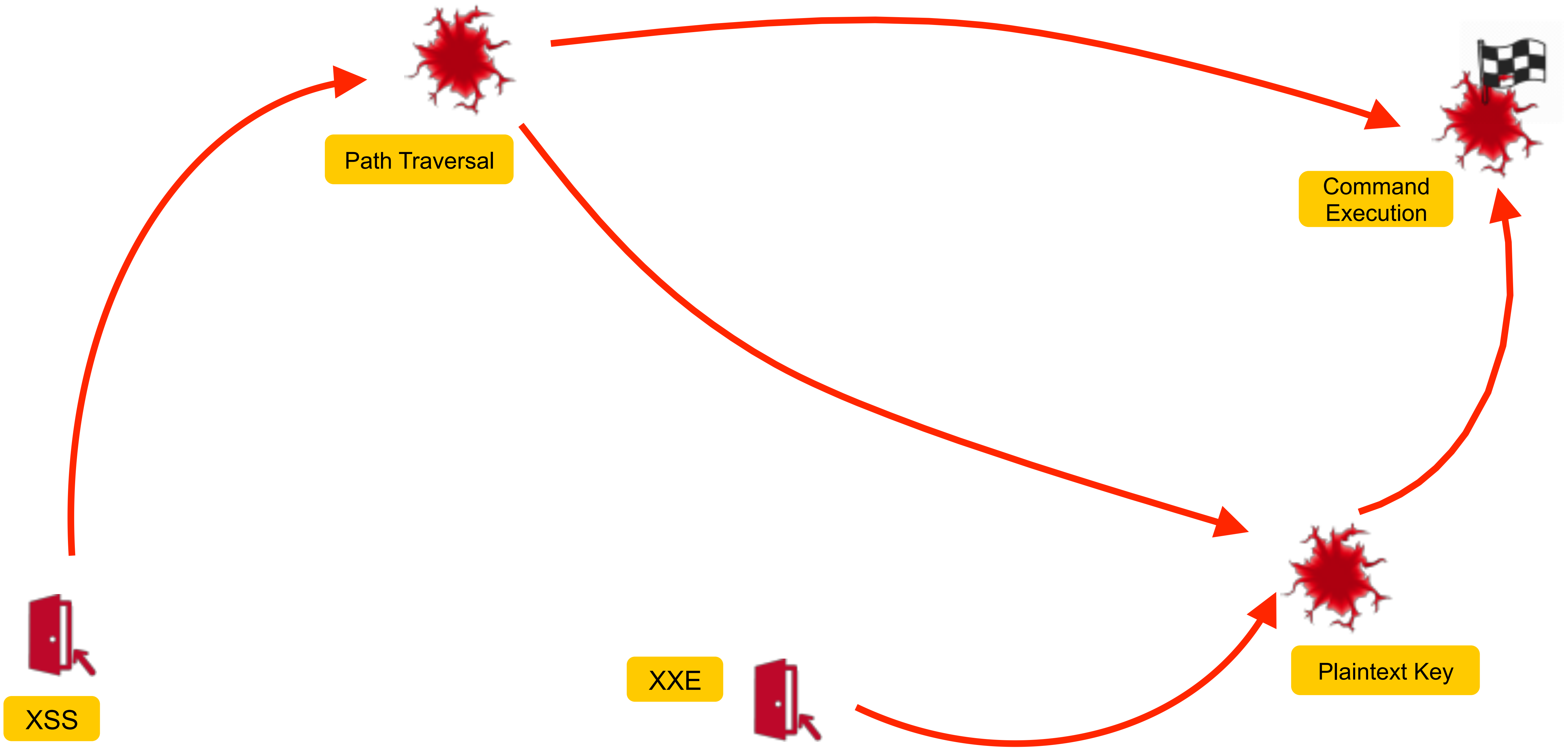
```
public Collection<Alarm> translateToAlarm(DeviceId deviceId, InputStream message) {
    try {
        Collection<Alarm> alarms = new ArrayList<>();
        Document doc = createDocFromMessage(message);

        // parse date element value into long
        Node eventTime = doc.getElementsByTagName(EVENTTIME_TAGNAME).item(0);
        String date = eventTime.getTextContent();
        long timeStamp = parseDate(date);
```

Hack Something Real!



```java
// Extracts the ZIP stream into the specified directory.
private void extractZipArchive(File dir, InputStream stream) throws IOException {
    ZipInputStream zis = new ZipInputStream(stream);
    ZipEntry entry;
    while ((entry = zis.getNextEntry()) != null) {
        if (!entry.isDirectory()) {
            byte[] data = toByteArray(zis);
            zis.closeEntry();
            File file = new File(dir, entry.getName());
            createParentDirs(file);
            write(data, file);
        }
    }
    zis.close();
```

Hack Something Real!



Evaluation

5 popular SDN Controller

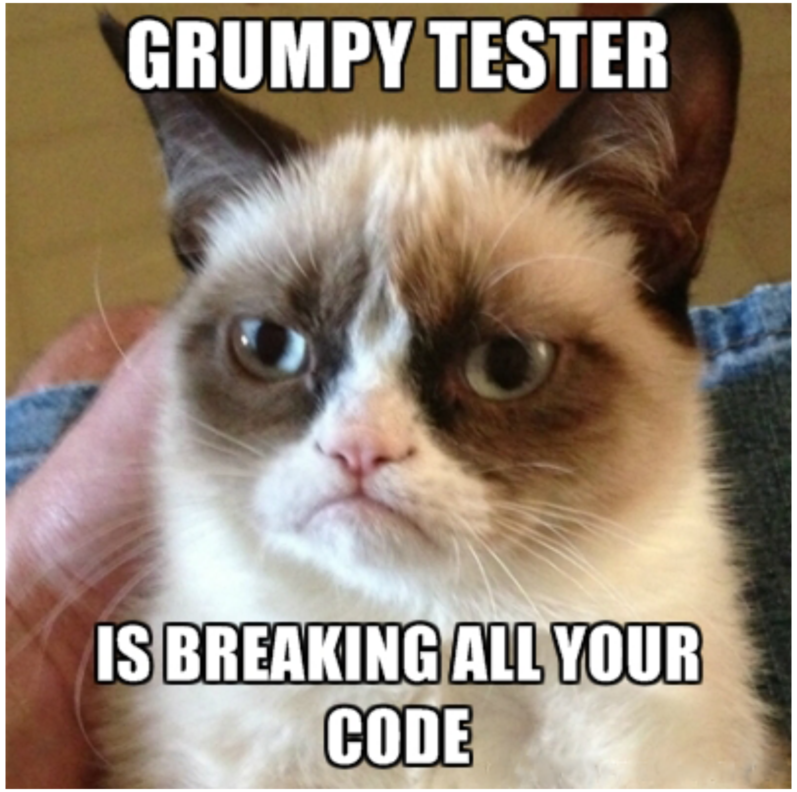
- Three open source projects (White-box)
- Two commercial products (Black-box)

54 apps

- Analyze 12 protocols
- Identify 476 dangerous function calls

18 zero-day vulnerabilities

- Construct 24 sophisticated exploit chains



Impact Analysis

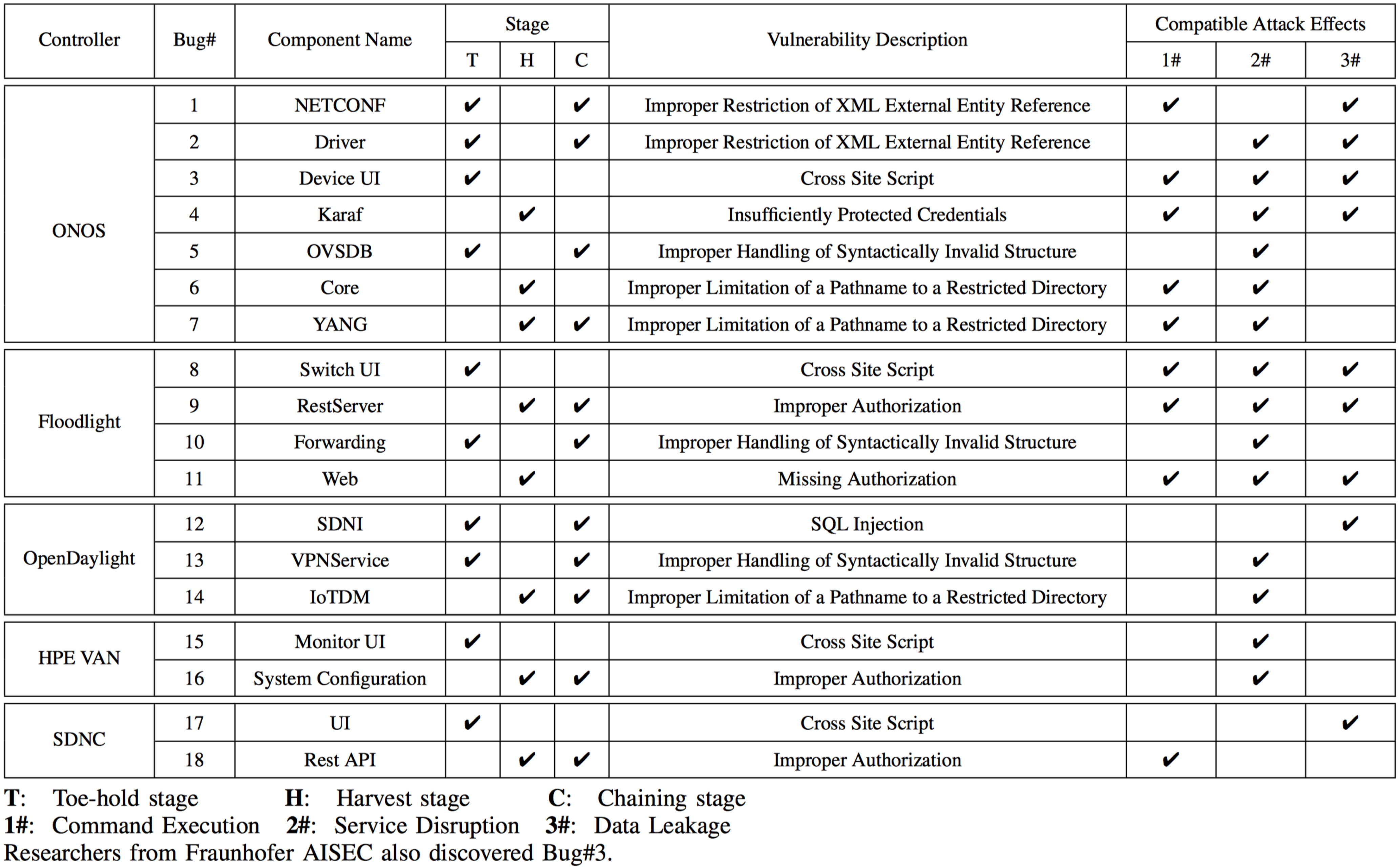
Get System Shell (1 of them)

Execute Arbitray SDN Commands (5 of them)

Steal Confidential Data (7 of them)

Crash/Disrupt Service (11 of them)

0day Profile



Controller	Bug#	Component Name	Stage			Vulnerability Description	Compatible Attack Effects		
			T	H	C		1#	2#	3#
ONOS	1	NETCONF	✔		✔	Improper Restriction of XML External Entity Reference	✔		✔
	2	Driver	✔		✔	Improper Restriction of XML External Entity Reference		✔	✔
	3	Device UI	✔			Cross Site Script	✔	✔	✔
	4	Karaf		✔		Insufficiently Protected Credentials	✔	✔	✔
	5	OVSDB	✔		✔	Improper Handling of Syntactically Invalid Structure		✔	
	6	Core		✔		Improper Limitation of a Pathname to a Restricted Directory	✔	✔	
	7	YANG		✔	✔	Improper Limitation of a Pathname to a Restricted Directory	✔	✔	
Floodlight	8	Switch UI	✔			Cross Site Script	✔	✔	✔
	9	RestServer		✔	✔	Improper Authorization	✔	✔	✔
	10	Forwarding	✔		✔	Improper Handling of Syntactically Invalid Structure		✔	
	11	Web		✔		Missing Authorization	✔	✔	✔
OpenDaylight	12	SDNI	✔		✔	SQL Injection			✔
	13	VPNService	✔		✔	Improper Handling of Syntactically Invalid Structure		✔	
	14	IoTDM		✔	✔	Improper Limitation of a Pathname to a Restricted Directory		✔	
HPE VAN	15	Monitor UI	✔			Cross Site Script		✔	
	16	System Configuration		✔	✔	Improper Authorization		✔	
SDNC	17	UI	✔			Cross Site Script			✔
	18	Rest API		✔	✔	Improper Authorization	✔		

T: Toe-hold stage **H:** Harvest stage **C:** Chaining stage
1#: Command Execution **2#:** Service Disruption **3#:** Data Leakage
Researchers from Fraunhofer AISEC also discovered Bug#3.

Thanks!

Email : f3ixiao@gmail.com

Homepage: http://fxiao.me

Twitter: @f3ixiao