OpenFlow Security

Junichi Murakami
Executive Officer, Director of Advanced Development Division

Fourteenforty Research Institute, Inc.
http://www.fourteenforty.jp

ver2.00.02

Agenda

1. Introduction

2. OpenFlow Overview
   - Software Defined Network(SDN) and OpenFlow
   - Background and circumstances
   - Technical basics
   - Controllers and Switches
   - Example of network design and traffic

3. Threat of OpenFlow
   - Threat analysis
   - Flow entry / Network capability / Switch / Controller
   - Conclusions
   - Further research

4. References

Introduction

- This slide describes an overview of OpenFlow technology and its threat analysis under the current specification

- This research focuses on the specification of OpenFlow 1.0

- Threats described in this slide does not always mean the feasibility of attacks on the threats is proven

Software Defined Network(SDN) and OpenFlow

- SDN
  - Usual networks are fixed system, which are defined by each network device's deployment, connections and configurations
  - Virtualizations for servers and storages are in progress in a data center in recent years.
  - A network is not so flexible yet, so it needs to be re-designed and re-configured every time (operation cost is highly increasing)
  - SDN is general concept to define network as software for making it more flexible in terms of its design, control and management
- OpenFlow
  - A kind of technology specifications to realize the SDN

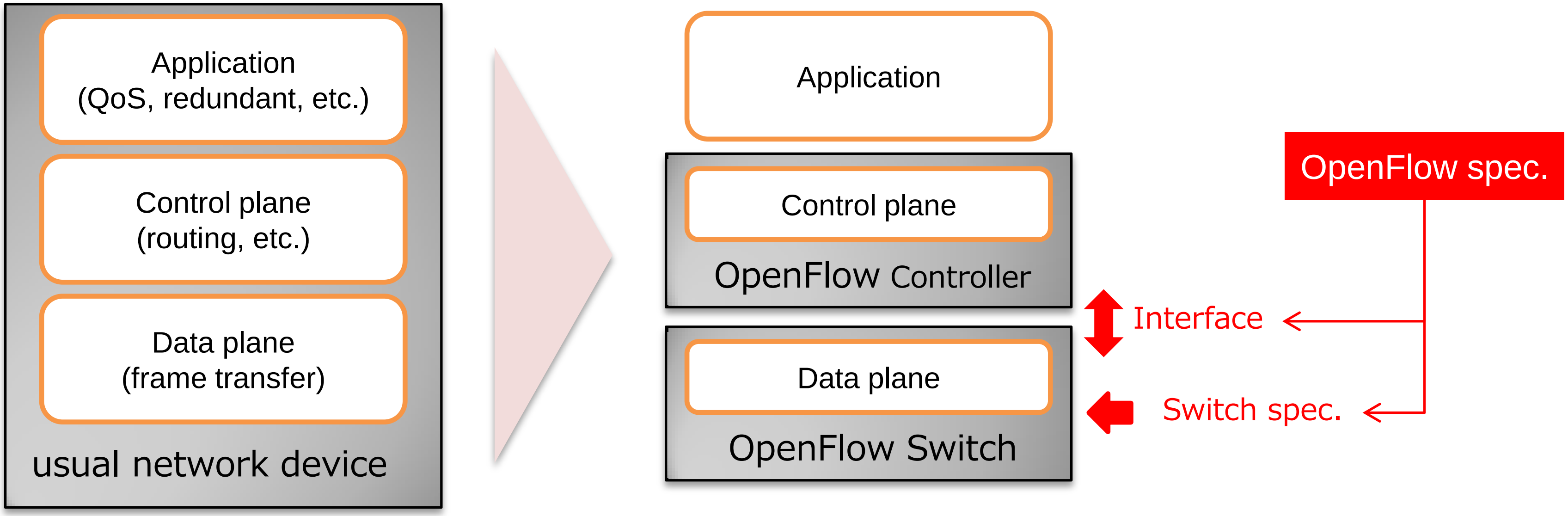
Background and circumstances

- Open Networking Foundation(ONF) draws up the specification
  - https://www.opennetworking.org/
- Board member of ONF is shown as below(4/15/2013)
  - Deutsche Telekom, Facebook, Goldman Sachs, Google, Microsoft, NTT Communications, Verizon, Yahoo!
- Currently most implementations are based on version 1.0

Date	Occurrence
12/31/2009	Version 1.0 published(mainly worked by Stanford University)
2/28/2011	Version 1.1 published
3/21/2011	Open Networking Foundation Founded
12/5/2011	Version 1.2 published
5/25/2012	Version 1.3 published
9/6/2012	Version 1.3.1 published

Technical basis (1/5)

- Basic concept
  - Separate control plane from network devices
  - Build up network with OpenFlow Controllers and OpenFlow Switches
  - The specification mainly defines switch spec. and communication interface between OpenFlow Controllers and OpenFlow Switches



Technical Basis (2/5)

- Flow
  - A unit of traffic handled by OpenFlow

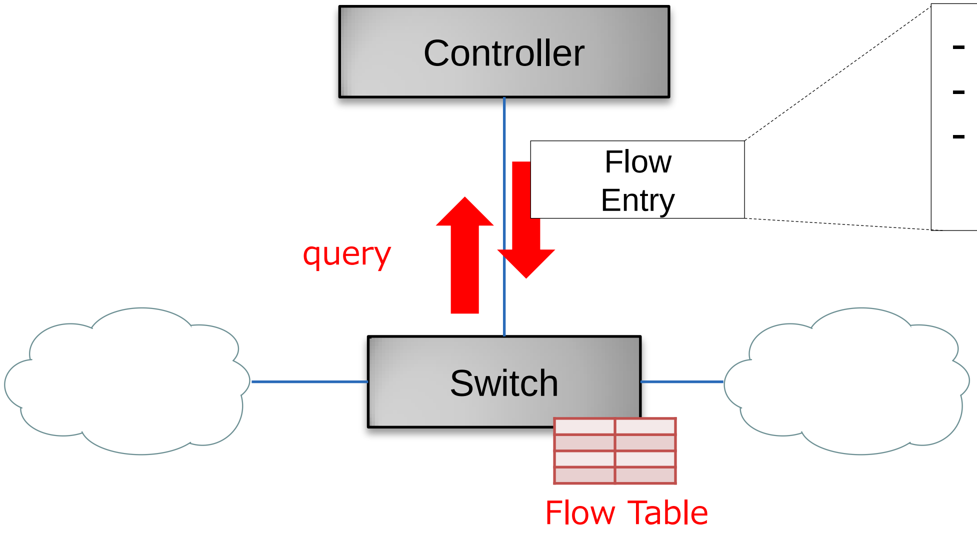
- Flow Entry: management structure of Flow consists of 3 elements below
  - Header Fields : conditions to determine a target flow
  - Instructions: a set of actions which describes how the matched flow being processed
  - Counter : statistics information of the matched flow

Header Fields	
Ingress port	IP src
Ether src	IP dst
Ether dst	IP proto
Ether type	IP ToS bits
VLAN id	TCP/UDP src port
VLAN priority	TCP/UDP dst port

Actions (partial)	
Forward	output the flow to specified port
DROP	discard the flow
Modify-Field	modify specified fields of the flow

Technical Basis (3/5)

- Controller
  - Write a flow entry to a switch
  - Respond to a switch's query (shown as below)
- Switch
  - Keep flow entries on a flow table
  - Process each flow based on a flow table
  - Query to controller if appropriate entry does not exist



* Flow Entry (eg.)

- output flow to port3 if IP_DST == 192.168.1.4
- drop all UDP flow
- modify IP_SRC to 192.168.1.5 and output the flow to port5 if IP_DST == 10.1.3.51

Technical Basis (4/5)

- Can control switch behavior based on flow entry
  - repeater, switch, router, load balancer and so on
- Doesn’t need to change physical connections and each device configurations
- Retrieve counters from each switch’s flow table
  - Can manipulate routing appropriately according to flow type and load
- Each flow entry on a flow table has a timeout
  - hard timeout
  - idle timeout

Technical Basis (5/5)

- Secure-Channel : communication interface between switches and controllers
  - following messages are exchanged over TCP or TLS connections

a. Controller to Switch
- Features : to request the capability of a switch
- Configuration : to set and query configuration parameter in a switch
- Modify-State: to add or delete entry in a flow table and modify port configuration
- Read-State : to collect statistics from a switch

b. Switch to Controller (asynchronous)
- Packet-in : to notify an incoming packet which is not matched to any flow entry
- Flow-Removed : to notify a flow has expired and is deleted from a table
- Port-Status : to notify switch's port configuration states has changed (eg. link-down)

c. bidirectional (asynchronous)
- HELLO: messages exchanged when establishing a connection
- ECHO（Request/Reply）: ping/pong over the secure-channel

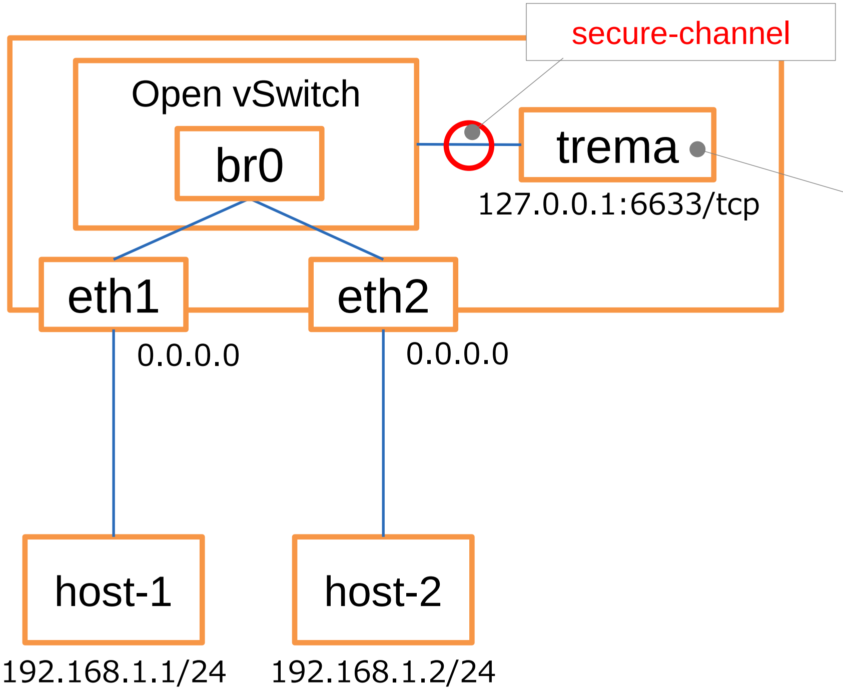
Controllers and Switches

- Both software and hardware implantations are available
- Hardware based switch is a bit expensive yet

	Software	Hardware
Switch	• Open vSwitch(OSS) • Indigo(OSS) • LINC(OSS) • UNIVERGE PF1000(NEC)	• UNIVERGE PF5220/PF5240/ PF5248/PF5820（NEC) • RackSwitch G8264/G8264T(IBM) • Pronto 3290/3780(Pica8) • AS4600-54T/L3(Riava) • HP2920-24G(HP)
Controller	• NOX(OSS) • POX(OSS) • Trema(OSS) • Floodlight(OSS) • Virtual Network Controller Version 2.0 (NTT data) • Ryu(OSS)	• UNIVERGE PF6800(NEC)

Example of network design and traffic(1/4)

- Install Open vSwitch and Trema on Linux box
- Create a bridge device as br0 and activate it
- Run Trema on localhost:6633/tcp, and specify the controller’s address in the switch parameter
- Run the controller code below on Trema which makes the switch act like an repeater



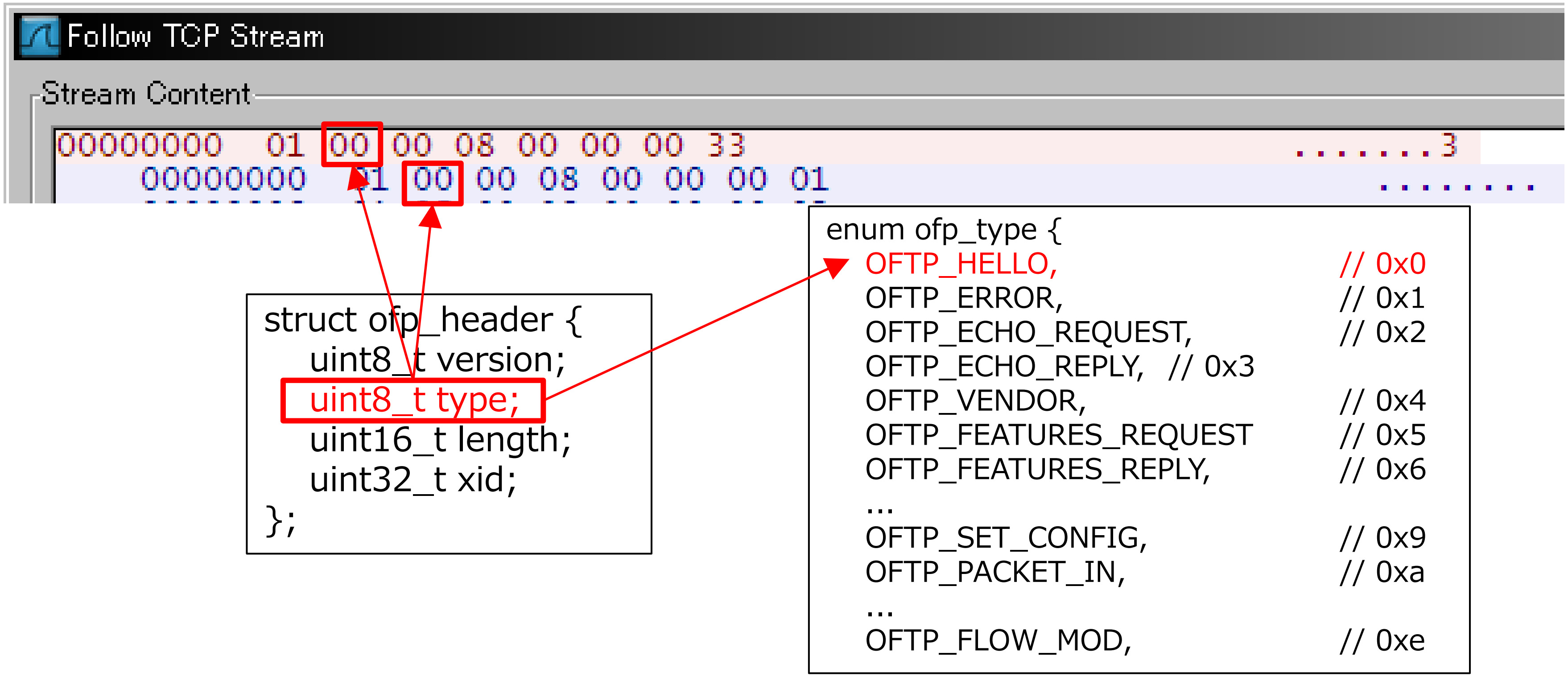
```
class RepeaterHub < Controller
  def packet_in datapath_id, message
    send_flow_mod_add(
      datapath_id,
      :match => ExactMatch.from( message ),
      :actions => ActionOutput.new( OFPP_FLOOD )
    )
    send_packet_out(
      datapath_id,
      :packet_in => message,
      :actions => ActionOutput.new( OFPP_FLOOD )
    )
  end
end
```

src: http://www.trema.info/2012/09/repeater-hub/

Example of network design and traffic(2/4)

- Both controller and switch run on same Linux box
- TCP based Secure-Channel (not TLS)
- Red background on screen is the switch’s traffic

■Exchanging HELLO messages between the switch and the controller



Follow TCP Stream

Stream Content

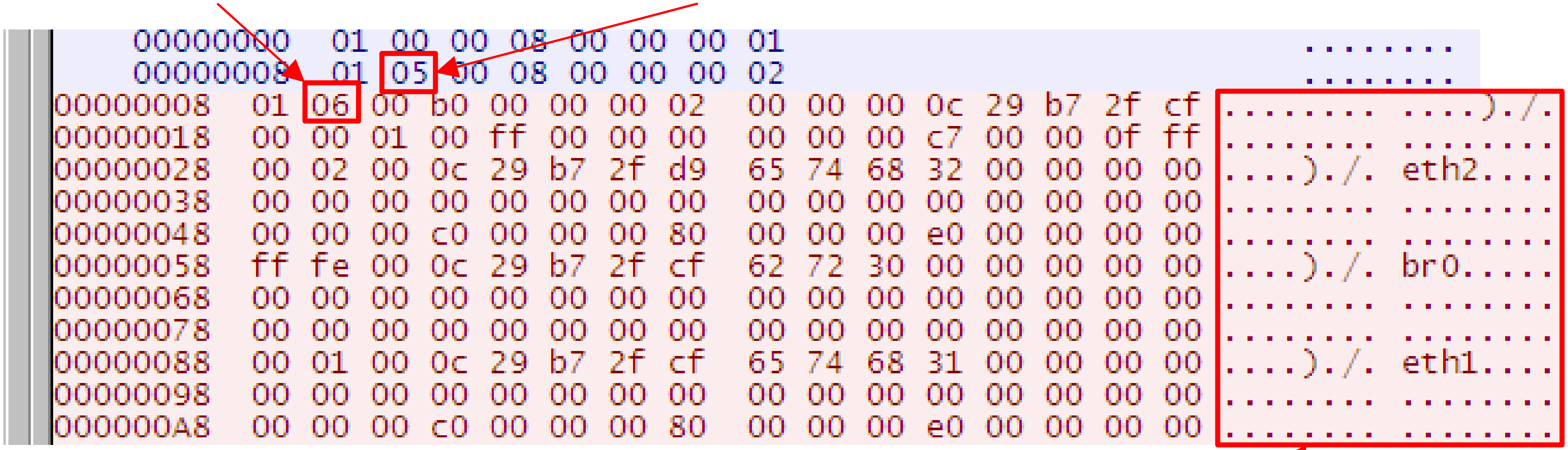
```
00000000  01 00 00 08 00 00 00 33                          .......3
    00000000  01 00 00 08 00 00 00 01                          ........
```

```
struct ofp_header {
    uint8_t version;
    uint8_t type;
    uint16_t length;
    uint32_t xid;
};
```

```
enum ofp_type {
    OFTP_HELLO,                    // 0x0
    OFTP_ERROR,                    // 0x1
    OFTP_ECHO_REQUEST,             // 0x2
    OFTP_ECHO_REPLY,   // 0x3
    OFTP_VENDOR,                   // 0x4
    OFTP_FEATURES_REQUEST          // 0x5
    OFTP_FEATURES_REPLY,           // 0x6
    …
    OFTP_SET_CONFIG,               // 0x9
    OFTP_PACKET_IN,                // 0xa
    …
    OFTP_FLOW_MOD,                 // 0xe
```

Example of network design and traffic (3/4)

- Features request and reply



OFTP_FEATURES_REPLY

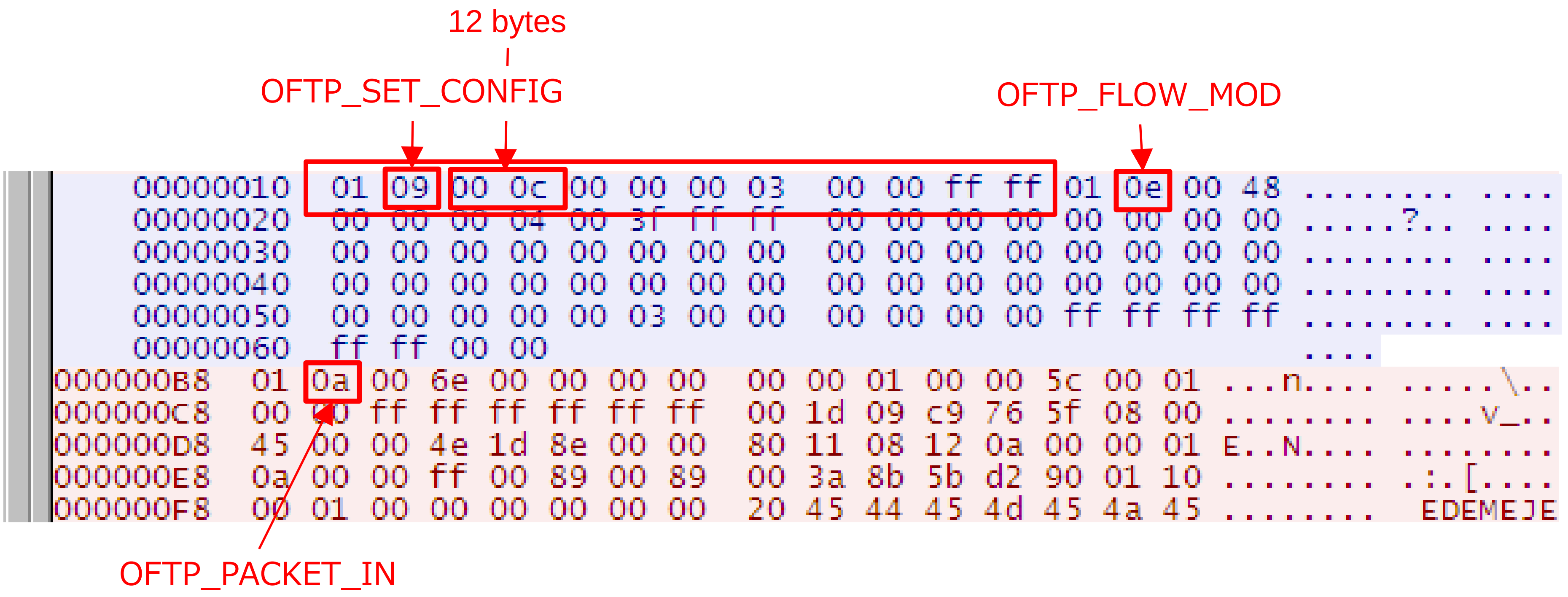
OFTP_FEATURES_REQUEST

```
    00000000  01 00 00 08 00 00 00 01                            ........
    00000008  01 05 00 08 00 00 00 02                            ........
00000008  01 06 00 b0 00 00 00 02  00 00 00 0c 29 b7 2f cf  ........ ....)./.
00000018  00 00 01 00 ff 00 00 00  00 00 00 c7 00 00 0f ff  ........ ........
00000028  00 02 00 0c 29 b7 2f d9  65 74 68 32 00 00 00 00  ....)./. eth2....
00000038  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00000048  00 00 00 c0 00 00 00 80  00 00 00 e0 00 00 00 00  ........ ........
00000058  ff fe 00 0c 29 b7 2f cf  62 72 30 00 00 00 00 00  ....)./. br0.....
00000068  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00000078  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00000088  00 01 00 0c 29 b7 2f cf  65 74 68 31 00 00 00 00  ....)./. eth1....
00000098  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
000000A8  00 00 00 c0 00 00 00 80  00 00 00 e0 00 00 00 00  ........ ........
```

replying port configurations

Example of network design and traffic (4/4)

- Initial configuration and writing flow entry from controller / PACKET_IN message from switch



12 bytes

OFTP_SET_CONFIG

OFTP_FLOW_MOD

OFTP_PACKET_IN

```
    00000010  01 09 00 0c 00 00 00 03  00 00 ff ff 01 0e 00 48 ........ ....
    00000020  00 00 00 04 00 3f ff ff  00 00 00 00 00 00 00 00 .....?.. ....
    00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ....
    00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ....
    00000050  00 00 00 00 00 03 00 00  00 00 00 00 ff ff ff ff ........ ....
    00000060  ff ff 00 00                                               ....
000000B8  01 0a 00 6e 00 00 00 00  00 00 01 00 00 5c 00 01 ...n.... .....\..
000000C8  00 00 ff ff ff ff ff ff  00 1d 09 c9 76 5f 08 00 ........ ....v_..
000000D8  45 00 00 4e 1d 8e 00 00  80 11 08 12 0a 00 00 01 E..N.... ........
000000E8  0a 00 00 ff 00 89 00 89  00 3a 8b 5b d2 90 01 10 ........ .:.[....
000000F8  00 01 00 00 00 00 00 00  20 45 44 45 4d 45 4a 45 ........  EDEMEJE
```

Threat analysis

[premises]

- Assets: a)Flow entry in switch, b)Network capability offered by OpenFlow
- Information system: c)Switch, d)Controller
- Analyze assets' threat against CIA and the others are against CIAAAR
  - CIAAAR: ISO/IEC TR 13335(GMITS)

	Assets		Information system	
	Flow entry	Network capability	Switch	Controller
Confidentiality				
Integrity				
Availability				
Authenticity	▨	▨		
Accountability	▨	▨		
Reliability	▨	▨		

region for analysis

Flow entry

	Assumed threat	Countermeasure and comment
C	Information leaking on the network	Using TLS for Secure-Channel
	Information leaking from switches	Hardening switches
	Information leaking from controllers	Hardening controllers
I	Tampering on the network	Using TLS for Secure-Channel
	Tampering in switches	Hardening switches
	Tampering from controllers	Hardening controllers
A	Flooding a table using spoofed packet	Applying flow entry to prohibit address spoofing （References 2.c）
	Flushing a flow table in switches	Hardening switches

Network capability

	Assumed threat	Countermeasure and comment
C	Information leaking by corrupted flow entry	Hardening switches and controllers
I	Traffic tampering by corrupted flow entry	Hardening switches and controllers
	Integrity loss by secure channel disconnection	making secure-channel redundant
A	Denial of service by corrupted flow entry	Hardening switches and controllers
	Denial of service by switches and controllers failure	Hardening switches and controllers
	Network failure by secure channel disconnection	making secure-channel redundant

Switch

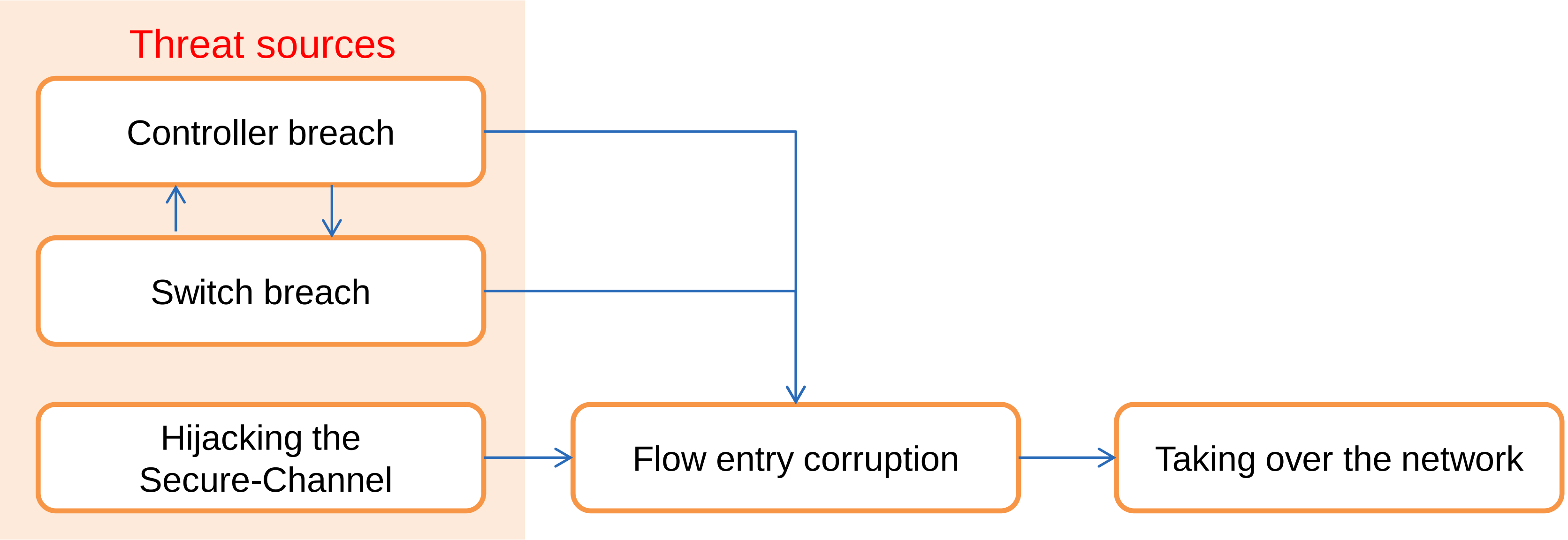
	Assumed threat	Countermeasure and comment
Co	Hacking a system (eg. exploiting, password cracking)	Hardening switches
In	Hacking a system (eg. exploiting, password cracking)	Hardening switches
Av	Hacking a system (eg. exploiting, password cracking)	Hardening switches
	DoS attack from controllers	Hardening controllers (premise: controllers compromise)
	Dos attack from others	Applying the flow entry considered such attack
	Hardware/Software failure	Making a system redundant
Au	Hacking a system (eg. password cracking, identity theft)	Hardening switches
	Redirection to fake controllers (eg. ARP Poisoning)	Authenticating controllers using TLS based on certifications
Ac	Tampering logs by hacking	Hardening switches
Re	Hacking a system (eg. exploiting, password cracking)	Hardening switches

Controller

	Assumed threat	Countermeasure and comment
Co	Hacking a system (eg. exploiting, password cracking)	Hardening controllers
In	Hacking a system (eg. exploiting, password cracking)	Hardening controllers
Av	Hacking a system (eg. exploiting, password cracking)	Hardening controllers
	DoS attack from switches	Hardening switches (premise: switches compromise)
	DoS attack from others	Applying the flow entry considered such attack
	Hardware/Software failure	Making a system redundant
Au	Hacking a system (eg. password cracking, identity theft)	Hardening controllers
	Redirection to fake switches (eg. ARP Poisoning)	Authenticating switches using TLS based on certifications
Ac	Tampering logs by system hacking	Hardening controllers
Re	Hacking a system (eg. exploiting, password cracking)	Hardening controllers

Conclusions

- Hardening switches and controllers and TLS for Secure-Channel are required (depends on where both devices be deployed)
- Both switches and controllers have software component in the system
  - usual countermeasures are important technically and operationally
- Especially, should be careful about controllers as it might be an SPOF



Further research

- Any way to make a DoS situation to a controller by sending special crafted packet like smurf(#1) attack and DNS Amp(#2) attack?
  - Packet-in flood
  - Flow-Removed flood
  - Port-Status flood
- remote flow entry detection by various probing packets
- Auditing each individual device’s design and implementation
- Security problem from actual environment and operations
  - Logic error in flow entry

#1 http://www.ipa.go.jp/security/ciadr/crword.html#S
#2 http://www.ipa.go.jp/security/vuln/documents/2008/200812_DNS.html

References

1. Books (Japanese)
   a. クラウド時代のネットワーク技術 OpenFlow実践入門
   （ISBN-10: 4774154652）

2. Online resources
   a. Openflow Networking Foundation
   https://www.opennetworking.org/

   b. OpenFlow Switch Specifications version 1.0.0
   http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf

   c. SDNのセキュリティ / Inter-Domain Routing Security 23 (Japanese)
   http://irs.ietf.to/wiki.cgi?page=IRS23