Monthly Research

# OpenFlowセキュリティ

Fourteenforty Research Institute, Inc.
株式会社 フォティーンフォティ技術研究所
http://www.fourteenforty.jp

ver2.00.02

# Agenda

1. はじめに

2. OpenFlow概要
   - Software Defined Network(SDN)及びOpenFlow
   - 背景及び周辺状況
   - 技術概要
   - コントローラー及びスイッチの実装例
   - OpenFlowネットワークの構成例
   - 実際の通信例

3. OpenFlowネットワークに想定される脅威
   - 脅威分析
   - ①フローエントリー / ②NW環境 / ③スイッチ / ④コントローラー
   - まとめ
   - 今後調査すべき事項

4. 参考情報

5. 謝辞

# はじめに

- 本書では、OpenFlowの概要を明らかにした上で、現行の仕様において想定されるセキュリティ上の脅威を調査、分析した結果を記載する。
- 本調査が対象とするOpenFlowの技術仕様は、1.0を前提とする。
- 想定される脅威は、可能性を示すものであり必ずしも現時点で攻撃の実現性を確認、実証したものではない。

# Agenda

1. はじめに
2. OpenFlow概要
   - Software Defined Network(SDN)及びOpenFlow
   - 背景及び周辺状況
   - 技術概要
   - コントローラー及びスイッチの実装例
   - OpenFlowネットワークの構成例
   - 実際の通信例
3. OpenFlowネットワークに想定される脅威
   - 脅威分析
   - ①フローエントリー / ②NW環境 / ③スイッチ / ④コントローラー
   - まとめ
   - 今後調査すべき事項
4. 参考情報
5. 謝辞

# Software Defined Network(SDN)及びOpenFlow

- SDN
  - 従来のネットワークは、各構成要素（L2/L3スイッチ、ルーター等）の物理的な配置、接続、設定に依存した固定的なシステム
  - データセンター等を中心にサーバ、及びストレージの仮想化が進んでいるが、ネットワークは都度構成変更、機器個別の設定が必要（VMのマシン間移動時等）
    **→　オペレーションの複雑化、運用負荷増大**
  - SDNは、ネットワークをソフトウェアとして定義し、柔軟な構成、制御、管理を実現しようとする概念

- OpenFlow
  - SDNという概念を具現化する技術仕様のひとつ

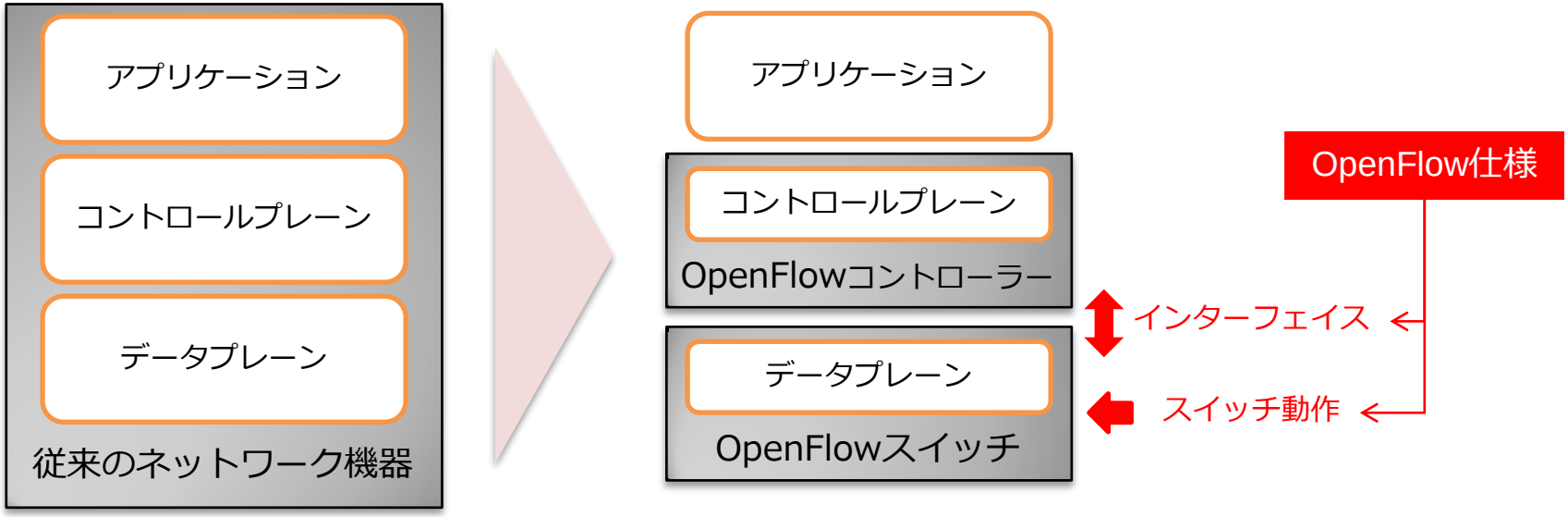
# 背景及び周辺状況

- 現在はOpen Networking Foundation(ONF)が仕様を策定
  - https://www.opennetworking.org/
- ONFのボードメンバーは下記の通り（2013年4月15日現在)
  - Deutsche Telekom, Facebook, Goldman Sachs, Google, Microsoft, NTTコミュニケーションズ, Verizon, Yahoo!
- 現時点では、バージョン1.0に準拠した実装が主流

| 年月日 | 出来事 |
|---|---|
| 2009年12月31日 | スタンフォード大学が中心となりバージョン1.0策定 |
| 2011年2月28日 | バージョン1.1策定 |
| 2011年3月21日 | Open Networking Foundation設立 |
| 2011年12月5日 | バージョン1.2策定 |
| 2012年5月25日 | バージョン1.3.0策定 |
| 2012年9月6日 | バージョン1.3.1策定 |

# 技術概要（1/5）

- OpenFlowの基本的な考え方
  - 従来NW機器に統合されていたコントロールプレーン（経路制御等）、データプレーン（フレーム転送等）を分離
  - NWをOpenFlowスイッチ及びOpenFlowコントローラーで構成
  - 仕様は、主にスイッチの動作及びスイッチ-コントローラー間の通信を規定



# 技術概要（2/5）

- フロー
  - OpenFlowにおける通信の取扱い単位

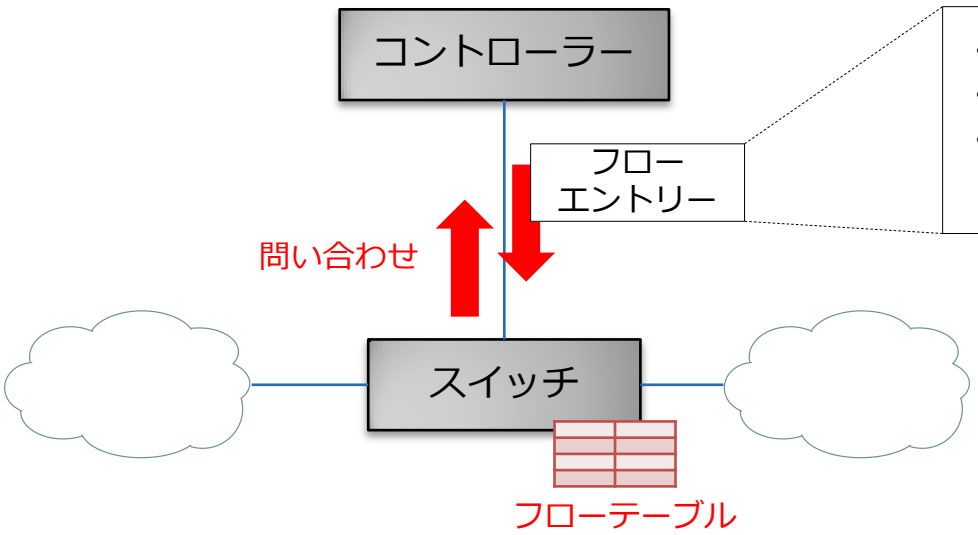
- フローエントリー：下記の3要素から構成されるフローの管理構造
  - ヘッダーフィールド：処理対象のフローを定義
  - アクション：該当フローの処理方法を定義
  - カウンタ：該当フローの統計情報を保持

| ヘッダーフィールドで利用可能な項目 | |
|---|---|
| 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 |

| 利用可能なアクション例 | |
|---|---|
| Forward | 指定したポートからパケットを出力 |
| DROP | パケットを破棄 |
| Modify-Field | 指定したフィールドを書き換え |

# 技術概要（3/5）

- コントローラー
  - フローエントリーをスイッチに書き込む
  - スイッチからの問い合わせ（下記）に対応
- スイッチ
  - フローエントリーをフローテーブルに保持
  - フローテーブルを参照し、パケットを処理する
  - 対応がフローテーブルに存在しない場合、コントローラーに問い合わせ




■フローエントリーの例

・192.168.1.4宛のパケットは3番ポートから出力
・UDPパケットは全てドロップ
・10.1.3.51宛のパケットは、送信元IPアドレスを192.168.1.5に書き換えて5番ポートから出力、等

# 技術概要（4/5）

- スイッチの動作、役割はフローエントリー次第
  - リピータ、スイッチングハブ、ルーター、ロードバランサー等

- NWの構成、設定変更が必要になった場合、コントローラー側から各スイッチのフローテーブルを変更することで一元的に制御可能
  → 物理的な構成変更、機器個別の設定が不要

- スイッチのフローテーブルからカウンタ値を取得可能
  - フローの種類/流量を考慮した経路制御等が可能

- フローテーブル上のフローエントリーに生存時間を設定可能
  - ハードタイムアウト：フローエントリーが書き込まれた時間を起点として指定時間経過した際にエントリーを削除する

  - アイドルタイムアウト：最後に当該フローがマッチした時間を起点として指定時間経過した際にエントリーを削除する（フローがマッチしたら経過時間を初期化）

# 技術概要（5/5）

- セキュアチャネル：スイッチ-コントローラー間の通信インターフェイス
  - TCPまたはTLSによるコネクション上で下記の通信を行う

a. コントローラーからスイッチへの通信
   - Features：スイッチがサポートしている機能の問い合わせ
   - Configuration：スイッチのパラメーター設定・取得
   - Modify-State：スイッチのフローエントリ変更（追加、削除）
     ポートの構成情報変更
   - Read-State：スイッチのフローテーブルから統計情報を取得

b. スイッチからコントローラーへの非同期通信
   - Packet-in：フローテーブルに該当しないパケットの受信をコントローラーに通知
   - Flow-Removed：タイムアウトによるフローエントリー削除通知
   - Port-Status：ポートの状態変更通知（Link-Down等）

c. 双方向での非同期通信
   - HELLO：セキュアチャネル確立初期に双方で交換するメッセージ
   - ECHO（Request/Reply）：セキュアチャネル上での死活確認メッセージ

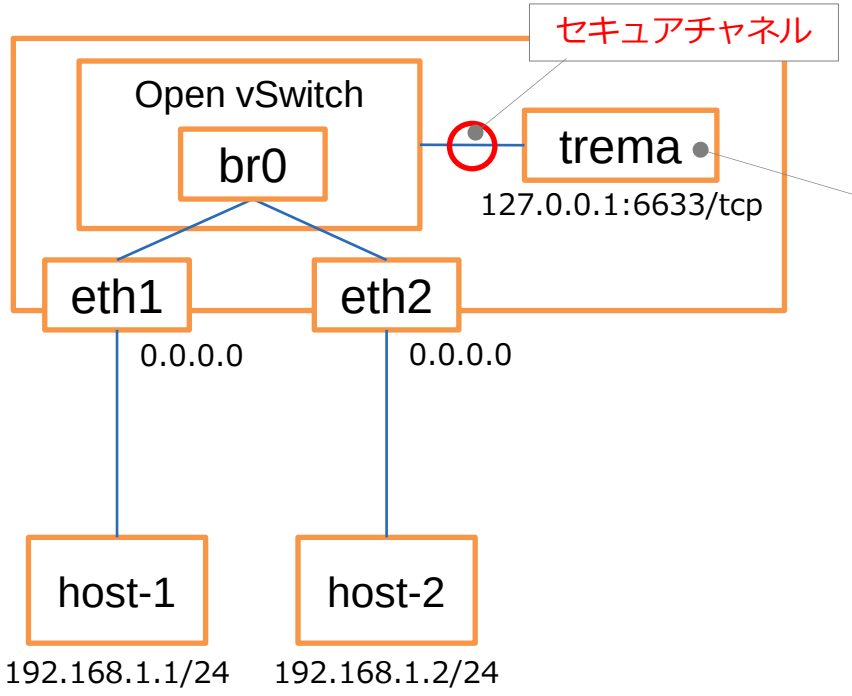
# コントローラー及びスイッチの実装例

- スイッチ、コントローラーともにソフトウェア、ハードウェア実装が存在
- ハードウェアベースのスイッチは現時点ではいずれも高価（数十～数百万円）

| | ソフトウェア | ハードウェア |
|---|---|---|
| スイッチ | ・Open vSwitch(OSS)<br>・Indigo(OSS)<br>・LINC(OSS)<br>・UNIVERGE PF1000(NEC) | ・UNIVERGE PF5220/PF5240/PF5248/PF5820（NEC)<br>・RackSwitch G8264/G8264T（IBM)<br>・Pronto 3290/3780（Pica8）<br>・AS4600-54T/L3（Riava）<br>・HP2920-24G（HP） |
| コントローラー | ・NOX（OSS)<br>・POX（OSS）<br>・Trema（OSS）<br>・Floodlight（OSS）<br>・バーチャルネットワークコントローラバージョン2.0（NTTデータ）<br>・Ryu(OSS) | ・UNIVERGE PF6800（NEC) |

# OpenFlowネットワークの構成例

- Linuxマシン上にOpen vSwitch及びTremaをインストール、実行
- Open vSwitchを利用してeth1、eth2を含むブリッジIF(br0)を作成、立ち上げ
  - eth1、eth2はアドレス無し、インターフェイスの立ち上げのみ
- localhost:6633/tcpでTremaを起動し、スイッチにコントローラーのアドレスを設定
- Trema上でリピーター相当のコントローラーコードを実行




```
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
```

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

# 実際の通信例（1/3）

- Open vSwitch、Tremaによる検証環境を構築
- スイッチ、コントローラーともに同一ホスト(Linux)で実行
- localhost:6633/tcpで通信（非TLS）※画面上の赤背景がスイッチによる通信

■スイッチ-コントローラー間のHELLO通信

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
```

# 実際の通信例(2/3)

■コントローラーからのFeatures要求及びスイッチからの応答

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  ........ ........
```

物理ポートの構成情報状態等を通知

# 実際の通信例（3/3）

■コントローラーからスイッチへの初期設定及びフローエントリーの書き込み、スイッチからのPACKET_IN

12 bytes

OFTP_SET_CONFIG

OFTP_FLOW_MOD

```
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
```

OFTP_PACKET_IN

# Agenda

1. はじめに

2. OpenFlow概要
   - Software Defined Network(SDN)及びOpenFlow
   - 背景及び周辺状況
   - 技術概要
   - コントローラー及びスイッチの実装例
   - OpenFlowネットワークの構成例
   - 実際の通信例

3. OpenFlowネットワークに想定される脅威
   - 脅威分析
   - ①フローエントリー / ②NW環境 / ③スイッチ / ④コントローラー
   - まとめ
   - 今後調査すべき事項

4. 参考情報

5. 謝辞

# 脅威分析

[前提]

- ①「スイッチ上のフローエントリー」、②「OpenFlowによるNW環境」を資産と定義
- ③「スイッチ」及び④「コントローラー」を情報システムと定義
- 資産に対してはCIA（機密性/完全性/可用性)、情報システムに対してはCIAAAR（CIA+真正性/責任追跡性/信頼性)に基づいて脅威分析を実施
  - CIAAARの定義は、ISO/IEC TR 13335（GMITS)に準拠

| | 資産 | | 情報システム | |
|---|---|---|---|---|
| | ①フローエントリー | ②NW環境 | ③スイッチ | ④コントローラー |
| 機密性 | | | | |
| 完全性 | 本領域について分析を実施 | | | |
| 可用性 | | | | |
| 真正性 | | | | |
| 責任追跡性 | | | | |
| 信頼性 | | | | |

# ①フローエントリー

| | 想定される脅威 | 対策/備考 |
|---|---|---|
| 機密性 | 通信経路上での漏えい | セキュアチャネルにTLSを用いることで防御可能 |
| | スイッチからの漏えい | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |
| | コントローラーからの漏えい | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |
| 完全性 | 通信経路上での改竄 | セキュアチャネルにTLSを用いることで防御可能 |
| | スイッチ上での改竄 | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |
| | コントローラーからの改竄 | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |
| 可用性 | 外部からのフローテーブルの溢れ（Spoofingしたパケット等による） | アドレス詐称を禁止するフローエントリーの作成、適用（参考情報 2.c参照） |
| | スイッチ上でのフローテーブルのリセット | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |

# ②NW環境

| | 想定される脅威 | 対策/備考 |
|---|---|---|
| 機密性 | 改竄されたフローエントリーによる漏えい（不正なパケットの複製、転送等による） | スイッチ及びコントローラーのセキュリティ確保 |
| 完全性 | 改竄されたフローエントリーによる改竄 | スイッチ及びコントローラーのセキュリティ確保 |
| | セキュアチャネルの切断によるフローエントリーの完全性欠如 | セキュアチャネルの可用性確保、冗長化 |
| 可用性 | 改竄されたフローエントリーによる通信阻害 | スイッチ及びコントローラーのセキュリティ確保 |
| | スイッチ、コントローラーの障害による通信阻害 | スイッチ及びコントローラーのセキュリティ確保 |
| | セキュアチャネルの切断による通信障害 | セキュアチャネルの可用性確保、冗長化 |

# ③スイッチ

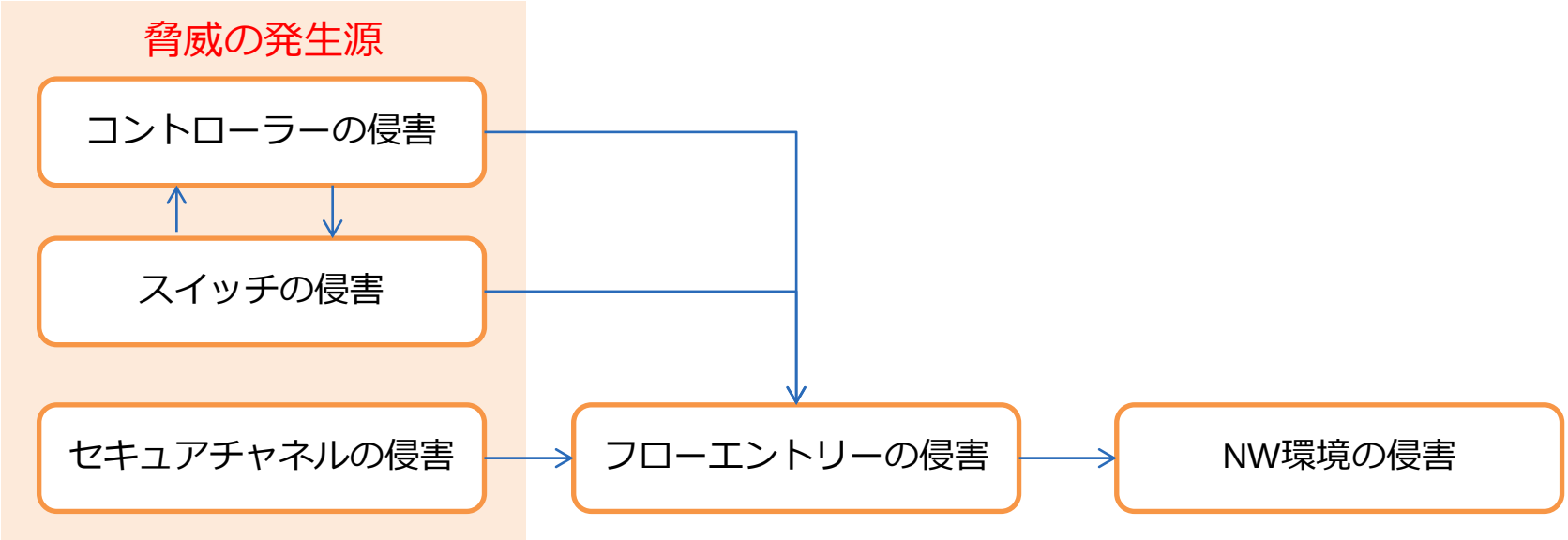
| | 想定される脅威 | 対策/備考 |
|---|---|---|
| 機密性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |
| 完全性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |
| 可用性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |
| | コントローラーからのDoS攻撃 | コントローラーのセキュリティ確保<br>（コントローラーが侵害されている前提） |
| | 他ノードからのDoS攻撃 | 攻撃を考慮したフローエントリーの作成、適用 |
| | ハードウェア/ソフトウェア障害 | システムの冗長化 |
| 真正性 | なりすましによるシステムへの侵入 | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |
| | 偽コントローラーへの誘導<br>（ARP Poisoning等による） | TLSによる証明書に基づいた接続先の認証<br>（適切な運用管理が必要） |
| 責任追跡性 | 各種ログの抹消<br>（システム侵害による） | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |
| 信頼性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | スイッチのセキュリティ確保（堅牢化、パッチ適用等） |

# ④コントローラー

| | 想定される脅威 | 対策/備考 |
|---|---|---|
| 機密性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |
| 完全性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |
| 可用性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |
| | スイッチからのDoS攻撃 | スイッチのセキュリティ確保<br>（スイッチが侵害されている前提） |
| | 他ノードからのDoS攻撃 | 攻撃を考慮したフローエントリーの作成、適用 |
| | ハードウェア/ソフトウェア障害 | システムの冗長化 |
| 真正性 | なりすましによるシステムへの侵入 | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |
| | 偽スイッチへの誘導<br>（ARP Poisoning等による） | TLSによる証明書に基づいた接続先の認証<br>（適切な運用管理が必要） |
| 責任追跡性 | 各種ログの抹消<br>（システム侵害による） | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |
| 信頼性 | システムへの侵入<br>（脆弱性攻撃、なりすまし等による） | コントローラーのセキュリティ確保（堅牢化、パッチ適用等） |

# まとめ

- スイッチ及びコントローラーのセキュリティ確保、セキュアチャネルのTLS利用は必須
  - スイッチ、コントローラの設置個所によってはリスクになり得る
- スイッチ、コントローラーともに（ハードウェア実装の場合でも）ソフトウェア部分が存在するため従来通り技術、運用面での対策が重要
- 特にコントローラーは、単一障害点になるため留意が必要



# 今後調査すべき項目

- Smurf攻撃(#1)やDNSアンプ攻撃(#2)のように細工したパケットをNW上に送信することで意図的にコントローラーへのDoS状態が発生する状況を作り出せないか？
  - 大量のPacket-inの生成
  - 大量のFlow-Removedの生成
  - 大量のPort-Statusの生成、等
- 様々なパケットの送信によるフローエントリーのリモート推測
- スイッチ、コントローラー個別の実装の調査（設計、実装等）
- 実際の運用から見たセキュリティ上の問題、課題
  - フローエントリーにおけるロジックエラー等

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

# Agenda

1. はじめに

2. OpenFlow概要
   - Software Defined Network(SDN)及びOpenFlow
   - 背景及び周辺状況
   - 技術概要
   - コントローラー及びスイッチの実装例
   - OpenFlowネットワークの構成例
   - 実際の通信例

3. OpenFlowネットワークに想定される脅威
   - 脅威分析
   - ①フローエントリー / ②NW環境 / ③スイッチ / ④コントローラー
   - まとめ
   - 今後調査すべき事項

4. 参考情報

5. 謝辞

# 参考情報

1. 書籍
   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. オンライン
   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
      http://irs.ietf.to/wiki.cgi?page=IRS23

# Agenda

1. はじめに

2. OpenFlow概要
   - Software Defined Network(SDN)及びOpenFlow
   - 背景及び周辺状況
   - 技術概要
   - コントローラー及びスイッチの実装例
   - OpenFlowネットワークの構成例
   - 実際の通信例

3. OpenFlowネットワークに想定される脅威
   - 脅威分析
   - ①フローエントリー / ②NW環境 / ③スイッチ / ④コントローラー
   - まとめ
   - 今後調査すべき事項

4. 参考情報

5. 謝辞

# 謝辞

本調査に当り協力・助言頂いた下記の方にこの場をお借りして
心より御礼申し上げます（ヒアリング実施順に記載）

NTTコミュニケーションズ（株）畑田充弘様、渡辺渉様、古澤徹様
NTTアドバンステクノロジ（株）深澤友雄様、伊藤光恭様、大嶋寛様、酒井清隆様
日本電信電話（株）NTTセキュアプラットフォーム研究所一同様