SDN-based Security Services using I2NSF

(draft-jeong-i2nsf-sdn-security-services-02)

http://datatracker.ietf.org/doc/draft-jeong-i2nsf-sdn-security-services/

성균관대학교 SUNG KYUN KWAN UNIVERSITY

Jaehoon (Paul) Jeong
pauljeong@skku.edu

ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute

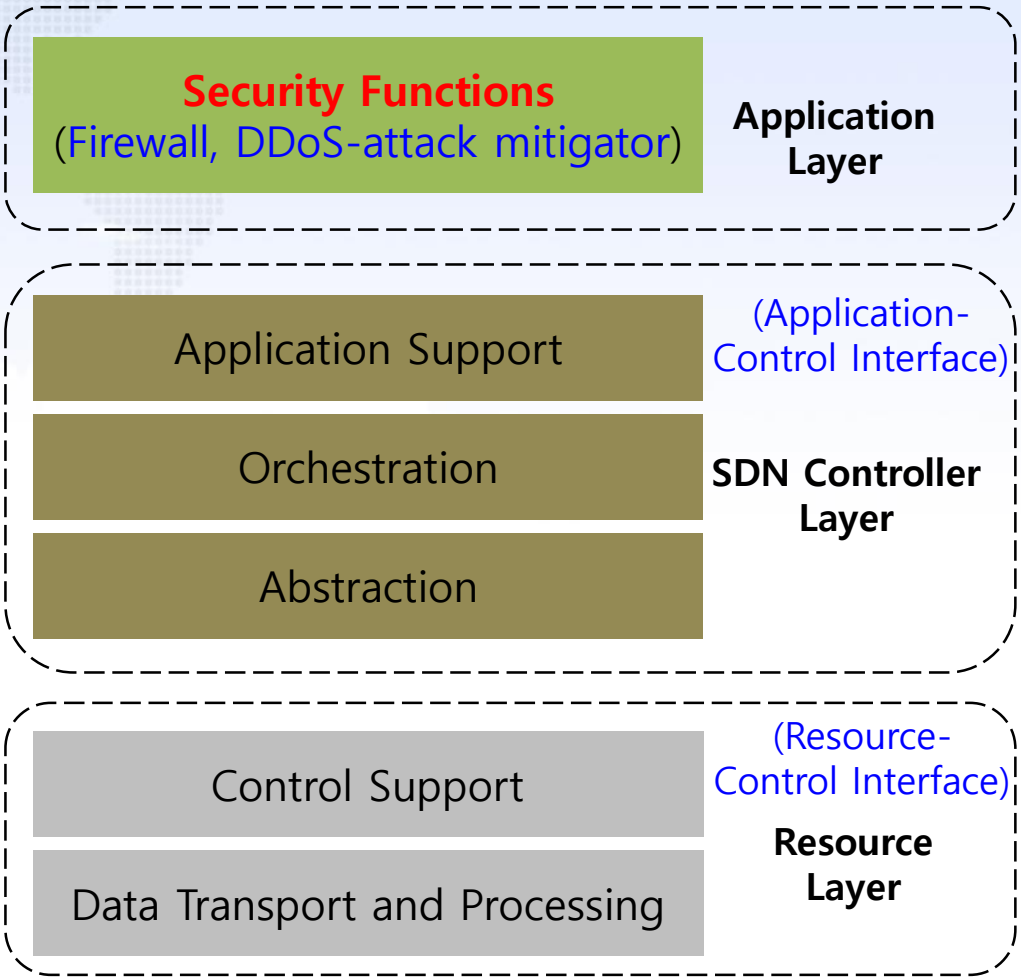
Contents

Architecture for SDN-based Security Services

Architecture (1/2)

❖ High-level Architecture for SDN-based Security Services

- An administrator enforces security policies for the security services.
- Access control rules are applied to network by SDN controller.
- Network resources (e.g., switches) act to mitigate network attacks.
  - ➢ e.g., dropping packets with suspicious patterns



Architecture (2/2)

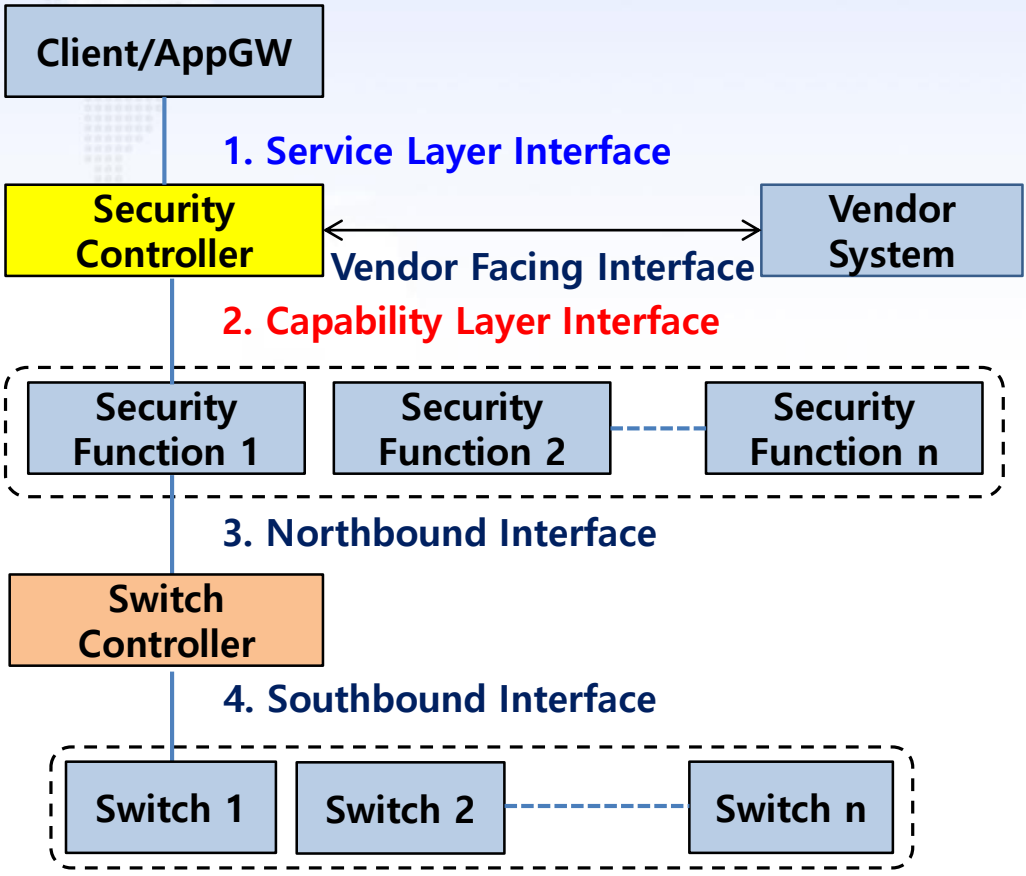
A framework to support SDN-based security services using I2NSF

1. **Client/AppGW** asks for security services with high-level security policies to Security Controller via **Service Layer Interface.**

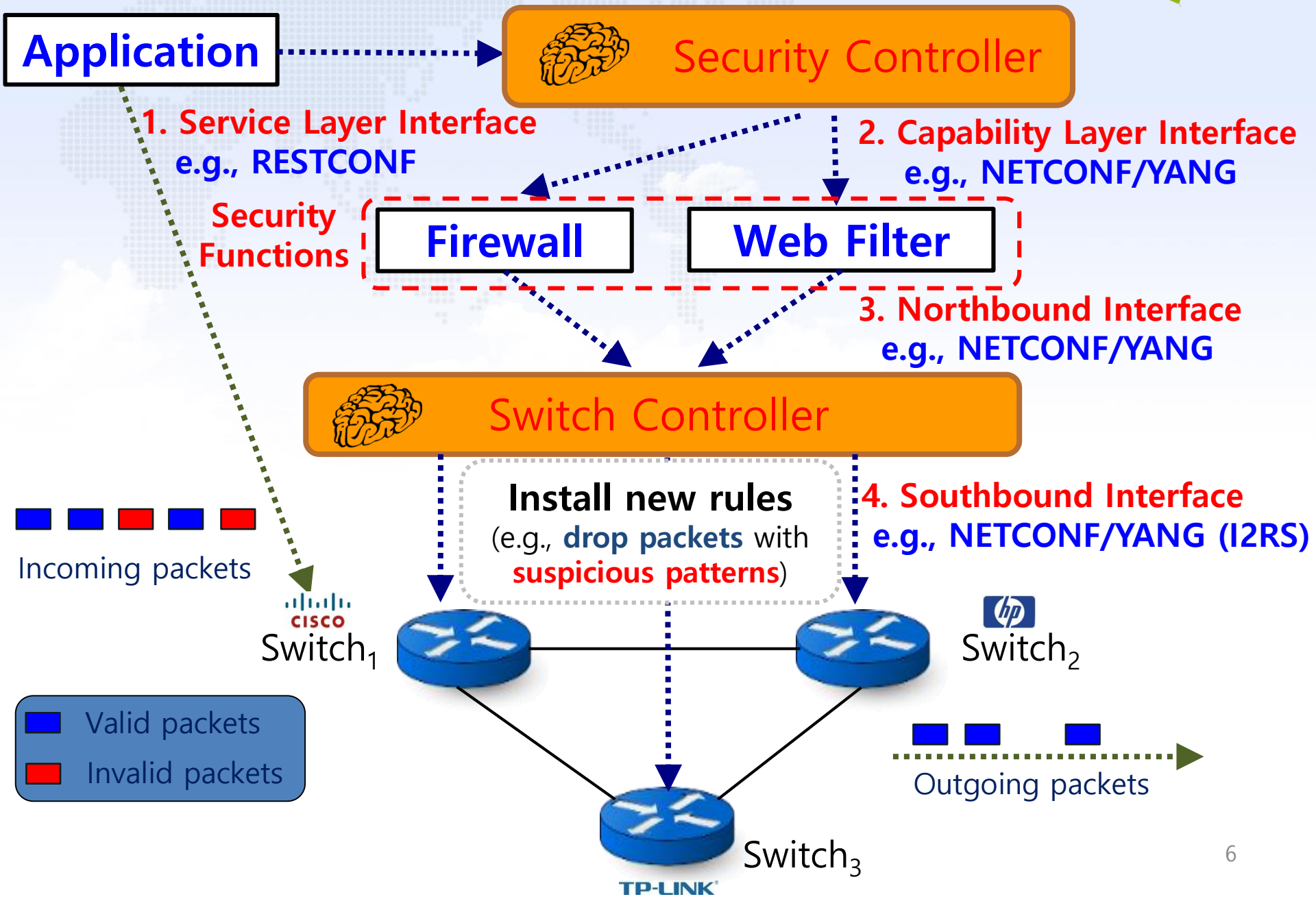
2. **Security Controller** calls function-level security services via **Capability Layer Interface.**

3. **Security Function** tells Switch Controller its required security services via **Northbound Interface.**

4. **Switch Controller** sets up forwarding rules for the security services on Switches via **Southbound Interface**.

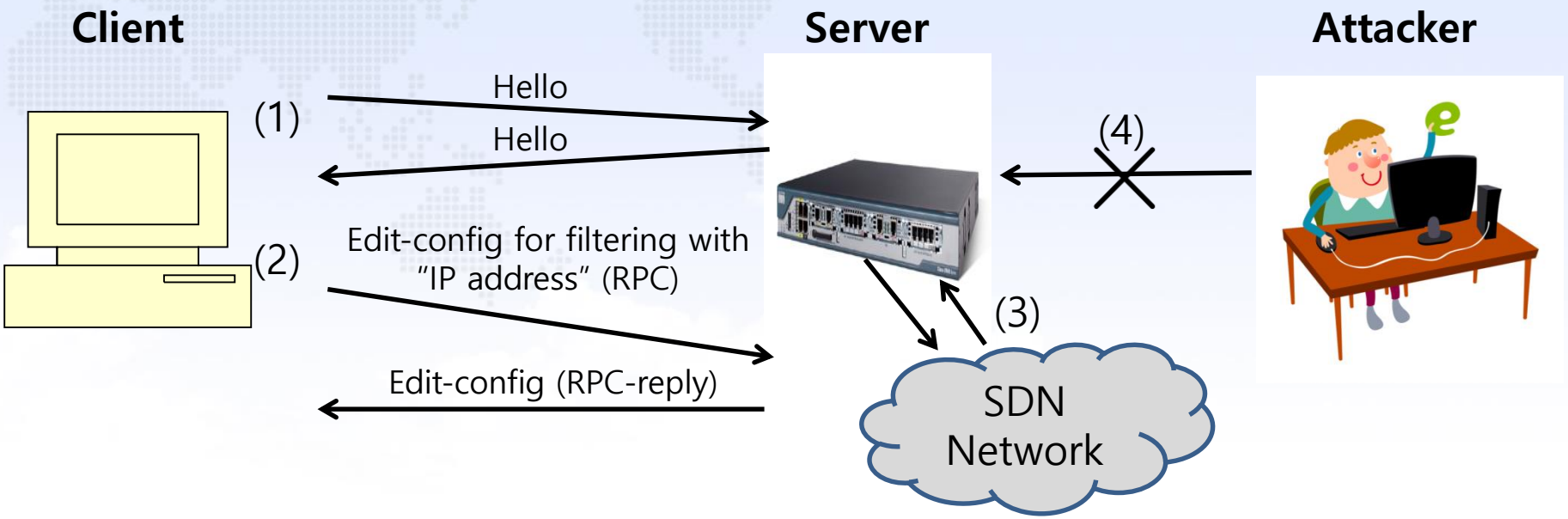


SDN-based Security Services using I2NSF




Program Execution for
Firewall Filtering

Procedure for SDN-based Firewall Filtering



1. Client and Server make a session by using NETCONF/YANG.

2. Client configures the **firewall table** of Server to block specific IP addresses.

3. Server (as Security Function in virtual machine) asks firewall filtering to be set up in Switches through Switch Controller.

4. After the configuration of the firewall table, packets from an attacker is dropped.

YANG Data Modeling for IP Address Filtering

IP Address Filtering

```
module filter {
  namespace "http://skku.com/cps/example/filter";
  prefix filter;

  import ietf-inet-types {
    prefix inet;
  }

  import tailf-common {
        prefix tailf;
  }


  /* A set of filtering structures  */
  container filters {
    tailf:callpoint hcp;

    list filter {
      key identification;
      max-elements 64;
      leaf identification {
        type string;
      }
      leaf where {
        type string;
        mandatory true;
      }
      leaf ip {
        type inet:ip-address;
        mandatory true;
      }
    }
  }
}
```

NETCONF Command for IP Address Filtering (1/2)

IP Address Filtering For Malicious Node 1

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <filters xmlns="http://skku.com/cps/example/filter"
               xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <filter nc:operation="create">
          <identification>Malicious_Node_1</identification>
          <where>Source</where>
          <ip>115.145.178.166</ip>
        </filter>
      </filters>
    </config>
  </edit-config>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <close-session/>
</rpc>
]]>]]>
```

NETCONF Command for IP Address Filtering (2/2)

IP Address Filtering For Malicious Node 2

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <filters xmlns="http://skku.com/cps/example/filter"
               xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <filter nc:operation="create">
          <identification>Malicious_Node_2</identification>
          <where>Source</where>
          <ip>115.145.178.167</ip>
        </filter>
      </filters>
    </config>
  </edit-config>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <close-session/>
</rpc>
]]>]]>
```

IP Addresses for Filtering

```
> show
   IP Malicious_Node_1     Source 115.145.178.166
   IP Malicious_Node_2     Source 115.145.178.167
```

Next Steps

- We will work for an **IETF Internet Draft of Capability Layer Interface** for SDN-based Security Services using I2NSF.
  - **Data Modeling** for Security Policies using **YANG**
    - **Compliant with draft-xia-i2nsf-capability-interface-im-02**
  - **Command Definitions** for **NETCONF**

- We will implement **Service Layer Interface** and **Capability Layer Interface for IETF I2NSF**
  - with **RESTCONF** and **NETCONF/YANG**, respectively.
  - **Service Layer Interface** will refer to **High-level Policy**, defined by **SUPA BoF**
    - For **defining an Interface** for Simplified Use of Policy Abstractions.