SXP

Release master

OpenDaylight Project

Mar 19, 2020

CONTENTS

CHAPTER

ONE

SXP DEVELOPER GUIDE

1.1 Overview

SXP (Scalable-Group Tag eXchange Protocol) project is an effort to enhance OpenDaylight platform with IP-SGT (IP Address to Source Group Tag) bindings that can be learned from connected SXP-aware network nodes. The current implementation supports SXP protocol version 4 according to the Smith, Kandula - SXP IETF draft and grouping of peers and creating filters based on ACL/Prefix-list syntax for filtering outbound and inbound IP-SGT bindings. All protocol legacy versions 1-3 are supported as well. Additionally, version 4 adds bidirectional connection type as an extension of a unidirectional one.

1.2 SXP Architecture

The SXP Server manages all connected clients in separate threads and a common SXP protocol agreement is used between connected peers. Each SXP network peer is modelled with its pertaining class, e.g., SXP Server represents the SXP Speaker, SXP Listener the Client. The server program creates the ServerSocket object on a specified port and waits until a client starts up and requests connect on the IP address and port of the server. The client program opens a Socket that is connected to the server running on the specified host IP address and port.

The SXP Listener maintains connection with its speaker peer. From an opened channel pipeline, all incoming SXP messages are processed by various handlers. Message must be decoded, parsed and validated.

The SXP Speaker is a counterpart to the SXP Listener. It maintains a connection with its listener peer and sends composed messages.

The SXP Binding Handler extracts the IP-SGT binding from a message and pulls it into the SXP-Database. If an error is detected during the IP-SGT extraction, an appropriate error code and sub-code is selected and an error message is sent back to the connected peer. All transitive messages are routed directly to the output queue of SXP Binding Dispatcher.

The SXP Binding Dispatcher represents a selector that will decides how many data from the SXP-database will be sent and when. It is responsible for message content composition based on maximum message length.

The SXP Binding Filters handles filtering of outgoing and incoming IP-SGT bindings according to BGP filtering using ACL and Prefix List syntax for specifying filter or based on Peer-sequence length.

The SXP Domains feature provides isolation of SXP peers and bindings learned between them, also exchange of Bindings is possible across SXP-Domains by ACL, Prefix List or Peer-Sequence filters

1.3 Key APIs and Interfaces

As this project is fairly small, it provides only few features that install and provide all APIs and implementations for this project.

- sxp-api
- spx-core
- sxp-controller
- sxp-cluster-route
- sxp-karaf
- sxp-robot
- sxp-system-tests

1.3.1 sxp-api

Contains data holders and entities

1.3.2 spx-core

Main logic and core features

1.3.3 sxp-controller

RPC request handling

1.3.4 sxp-cluster-route

Performs managing of SXP devices in cluster environment

1.3.5 sxp-karaf

Defines Karaf4 dependencies

1.3.6 sxp-robot

Contains JRobot libraries used in CSIT Robot tests

1.3.7 sxp-system-tests

Contains REST client used for testing purposes

1.4 API Reference Documentation

- RESTCONF Interface and Dynamic Tree
- Specification and Architecture

CHAPTER

TWO

SXP USER GUIDE

2.1 Overview

SXP (Scalable-Group Tag eXchange Protocol) project is an effort to enhance OpenDaylight platform with IP-SGT (IP Address to Source Group Tag) bindings that can be learned from connected SXP-aware network nodes. The current implementation supports SXP protocol version 4 according to the Smith, Kandula - SXP IETF draft and grouping of peers and creating filters based on ACL/Prefix-list syntax for filtering outbound and inbound IP-SGT bindings. All protocol legacy versions 1-3 are supported as well. Additionally, version 4 adds bidirectional connection type as an extension of a unidirectional one.

2.2 SXP Architecture

The SXP Server manages all connected clients in separate threads and a common SXP protocol agreement is used between connected peers. Each SXP network peer is modelled with its pertaining class, e.g., SXP Server represents the SXP Speaker, SXP Listener the Client. The server program creates the ServerSocket object on a specified port and waits until a client starts up and requests connect on the IP address and port of the server. The client program opens a Socket that is connected to the server running on the specified host IP address and port.

The SXP Listener maintains connection with its speaker peer. From an opened channel pipeline, all incoming SXP messages are processed by various handlers. Message must be decoded, parsed and validated.

The SXP Speaker is a counterpart to the SXP Listener. It maintains a connection with its listener peer and sends composed messages.

The SXP Binding Handler extracts the IP-SGT binding from a message and pulls it into the SXP-Database. If an error is detected during the IP-SGT extraction, an appropriate error code and sub-code is selected and an error message is sent back to the connected peer. All transitive messages are routed directly to the output queue of SXP Binding Dispatcher.

The SXP Binding Dispatcher represents a selector that will decides how many data from the SXP-database will be sent and when. It is responsible for message content composition based on maximum message length.

The SXP Binding Filters handles filtering of outgoing and incoming IP-SGT bindings according to BGP filtering using ACL and Prefix List syntax for specifying filter or based on Peer-sequence length.

The SXP Domains feature provides isolation of SXP peers and bindings learned between them, also exchange of Bindings is possible across SXP-Domains by ACL, Prefix List or Peer-Sequence filters

2.3 Configuring SXP

SXP requires no manual configuration.

2.4 Administering or Managing SXP

By RPC (response is XML document containing requested data or operation status):

- Get Connections POST http://127.0.0.1:8181/restconf/operations/sxp-controller:get-connections

```
<input xmlns:xsi="urn:opendaylight:sxp:controller">
 <domain-name>global</domain-name>
 <requested-node>0.0.0.100</requested-node>
</input>
```

- Add Connection POST http://127.0.0.1:8181/restconf/operations/sxp-controller:add-connection

```
<input xmlns:xsi="urn:opendaylight:sxp:controller">
 <requested-node>0.0.0.100</requested-node>
 <domain-name>global</domain-name>
 <connections>
  <connection>
   <peer-address>172.20.161.50</peer-address>
   <tcp-port>64999</tcp-port>
   <!-- Password setup: default | none leave empty -->
   <password>default</password>
   <!-- Mode: speaker/listener/both -->
   <mode>speaker</mode>
   <version>version4</version>
   <description>Connection to ASR1K</description>
   <!-- Timers setup: 0 to disable specific timer usability, the default value will␣
↪be used -->
   <connection-timers>
    <!-- Speaker -->
    <hold-time-min-acceptable>45</hold-time-min-acceptable>
    <keep-alive-time>30</keep-alive-time>
   </connection-timers>
  </connection>
  <connection>
   <peer-address>172.20.161.178</peer-address>
   <tcp-port>64999</tcp-port>
   <!-- Password setup: default | none leave empty-->
   <password>default</password>
   <!-- Mode: speaker/listener/both -->
   <mode>listener</mode>
   <version>version4</version>
   <description>Connection to ISR</description>
   <!-- Timers setup: 0 to disable specific timer usability, the default value will␣
↪be used -->
   <connection-timers>
    <!-- Listener -->
    <reconciliation-time>120</reconciliation-time>
    <hold-time>90</hold-time>
    <hold-time-min>90</hold-time-min>
    <hold-time-max>180</hold-time-max>
```

(continues on next page)

(continued from previous page)

```
   </connection-timers>
  </connection>
 </connections>
</input>
```

- Delete Connection POST http://127.0.0.1:8181/restconf/operations/sxp-controller:delete-connection

```
<input xmlns:xsi="urn:opendaylight:sxp:controller">
 <requested-node>0.0.0.100</requested-node>
 <domain-name>global</domain-name>
 <peer-address>172.20.161.50</peer-address>
</input>
```

- Add/Update Bindings POST http://127.0.0.1:8181/restconf/operations/sxp-controller:add-bindings

```
<input xmlns="urn:opendaylight:sxp:controller">
  <node-id>0.0.0.100</node-id>
  <domain-name>global</domain-name>
  <origin>LOCAL</origin>
  <master-database>
    <binding>
      <sgt>50</sgt>
      <ip-prefix>192.168.2.1/32</ip-prefix>
      <ip-prefix>192.168.2.2/32</ip-prefix>
    </binding>
    <binding>
      <sgt>100</sgt>
      <ip-prefix>192.168.3.1/32</ip-prefix>
      <ip-prefix>192.168.3.2/32</ip-prefix>
    </binding>
  </master-database>
</input>
```

- Delete Bindings POST http://127.0.0.1:8181/restconf/operations/sxp-controller:delete-bindings

```
<input xmlns="urn:opendaylight:sxp:controller">
  <node-id>0.0.0.100</node-id>
  <domain-name>global</domain-name>
  <binding>
    <sgt>50</sgt>
    <ip-prefix>192.168.2.2/32</ip-prefix>
  </binding>
  <binding>
    <sgt>100</sgt>
    <ip-prefix>192.168.3.2/32</ip-prefix>
  </binding>
</input>
```

- Get Node Bindings

  This RPC gets particular device bindings. An SXP-aware node is identified with a unique Node-ID. If a user requests bindings for a Speaker 20.0.0.2, the RPC will search for an appropriate path, which contains 20.0.0.2 Node-ID, within locally learnt SXP data in the SXP database and replies with associated bindings. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:get-node-bindings

```
<input xmlns:xsi="urn:opendaylight:sxp:controller">
 <requested-node>20.0.0.2</requested-node>
```

(continues on next page)

(continued from previous page)

```
 <bindings-range>all</bindings-range>
 <domain-name>global</domain-name>
</input>
```

- Get Binding SGTs POST http://127.0.0.1:8181/restconf/operations/sxp-controller:get-binding-sgts

```
<input xmlns:xsi="urn:opendaylight:sxp:controller">
 <requested-node>0.0.0.100</requested-node>
 <domain-name>global</domain-name>
 <ip-prefix>192.168.12.2/32</ip-prefix>
</input>
```

- Add PeerGroup with or without filters to node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:add-peer-group

```
<input xmlns="urn:opendaylight:sxp:controller">
 <requested-node>127.0.0.1</requested-node>
 <sxp-peer-group>
  <name>TEST</name>
  <sxp-peers>
  </sxp-peers>
  <sxp-filter>
   <filter-type>outbound</filter-type>
   <acl-entry>
    <entry-type>deny</entry-type>
    <entry-seq>1</entry-seq>
    <sgt-start>1</sgt-start>
    <sgt-end>100</sgt-end>
   </acl-entry>
   <acl-entry>
    <entry-type>permit</entry-type>
    <entry-seq>45</entry-seq>
    <matches>1</matches>
    <matches>3</matches>
    <matches>5</matches>
   </acl-entry>
  </sxp-filter>
 </sxp-peer-group>
</input>
```

- Delete PeerGroup with peer-group-name from node request-node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:delete-peer-group

```
<input xmlns="urn:opendaylight:sxp:controller">
 <requested-node>127.0.0.1</requested-node>
 <peer-group-name>TEST</peer-group-name>
</input>
```

- Get PeerGroup with peer-group-name from node request-node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:get-peer-group

```
<input xmlns="urn:opendaylight:sxp:controller">
 <requested-node>127.0.0.1</requested-node>
 <peer-group-name>TEST</peer-group-name>
</input>
```

- Add Filter to peer group on node request-node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:

add-filter

```
<input xmlns="urn:opendaylight:sxp:controller">
 <requested-node>127.0.0.1</requested-node>
 <peer-group-name>TEST</peer-group-name>
 <sxp-filter>
  <filter-type>outbound</filter-type>
  <acl-entry>
   <entry-type>deny</entry-type>
   <entry-seq>1</entry-seq>
   <sgt-start>1</sgt-start>
   <sgt-end>100</sgt-end>
  </acl-entry>
  <acl-entry>
   <entry-type>permit</entry-type>
   <entry-seq>45</entry-seq>
   <matches>1</matches>
   <matches>3</matches>
   <matches>5</matches>
  </acl-entry>
 </sxp-filter>
</input>
```

- Delete Filter from peer group on node request-node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:delete-filter

```
<input xmlns="urn:opendaylight:sxp:controller">
 <requested-node>127.0.0.1</requested-node>
 <peer-group-name>TEST</peer-group-name>
 <filter-type>outbound</filter-type>
</input>
```

- Update Filter of the same type in peer group on node request-node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:update-filter

```
<input xmlns="urn:opendaylight:sxp:controller">
 <requested-node>127.0.0.1</requested-node>
 <peer-group-name>TEST</peer-group-name>
 <sxp-filter>
  <filter-type>outbound</filter-type>
  <acl-entry>
   <entry-type>deny</entry-type>
   <entry-seq>1</entry-seq>
   <sgt-start>1</sgt-start>
   <sgt-end>100</sgt-end>
  </acl-entry>
  <acl-entry>
   <entry-type>permit</entry-type>
   <entry-seq>45</entry-seq>
   <matches>1</matches>
   <matches>3</matches>
   <matches>5</matches>
  </acl-entry>
 </sxp-filter>
</input>
```

- Add new SXP aware Node POST http://127.0.0.1:8181/restconf/operations/sxp-controller:add-node

```
<input xmlns="urn:opendaylight:sxp:controller">
  <node-id>1.1.1.1</node-id>
  <source-ip>0.0.0.0</source-ip>
  <timers>
    <retry-open-time>5</retry-open-time>
    <hold-time-min-acceptable>120</hold-time-min-acceptable>
    <delete-hold-down-time>120</delete-hold-down-time>
    <hold-time-min>90</hold-time-min>
    <reconciliation-time>120</reconciliation-time>
    <hold-time>90</hold-time>
    <hold-time-max>180</hold-time-max>
    <keep-alive-time>30</keep-alive-time>
  </timers>
  <mapping-expanded>150</mapping-expanded>
  <security>
    <password>password</password>
  </security>
  <tcp-port>64999</tcp-port>
  <version>version4</version>
  <description>ODL SXP Controller</description>
</input>
```

- Delete SXP aware node POST http://127.0.0.1:8181/restconf/operations/sxp-controller:delete-node

```
<input xmlns="urn:opendaylight:sxp:controller">
 <node-id>1.1.1.1</node-id>
</input>
```

- Add SXP Domain on node request-node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:add-domain

```
<input xmlns="urn:opendaylight:sxp:controller">
  <node-id>1.1.1.1</node-id>
  <domain-name>global</domain-name>
</input>
```

- Delete SXP Domain on node request-node. POST http://127.0.0.1:8181/restconf/operations/sxp-controller:delete-domain

```
<input xmlns="urn:opendaylight:sxp:controller">
 <node-id>1.1.1.1</node-id>
 <domain-name>global</domain-name>
</input>
```

- Add Route Adds route to leader Node. PUT http://127.0.0.1:8181/restconf/config/sxp-cluster-route:sxp-cluster-route/

```
<sxp-cluster-route xmlns="urn:opendaylight:sxp:cluster:route">
  <routing-definition>
    <ip-address>80.12.43.2</ip-address>
    <interface>eth1:0</interface>
    <netmask>255.255.255.0</netmask>
  </routing-definition>
</sxp-cluster-route>
```

2.4.1 Use cases for SXP

Cisco has a wide installed base of network devices supporting SXP. By including SXP in OpenDaylight, the binding of policy groups to IP addresses can be made available for possible further processing to a wide range of devices, and applications running on OpenDaylight. The range of applications that would be enabled is extensive. Here are just a few of them:

OpenDaylight based applications can take advantage of the IP-SGT binding information. For example, access control can be defined by an operator in terms of policy groups, while OpenDaylight can configure access control lists on network elements using IP addresses, e.g., existing technology.

Interoperability between different vendors. Vendors have different policy systems. Knowing the IP-SGT binding for Cisco makes it possible to maintain policy groups between Cisco and other vendors.

OpenDaylight can aggregate the binding information from many devices and communicate it to a network element. For example, a firewall can use the IP-SGT binding information to know how to handle IPs based on the group-based ACLs it has set. But to do this with SXP alone, the firewall has to maintain a large number of network connections to get the binding information. This incurs heavy overhead costs to maintain all of the SXP peering and protocol information. OpenDaylight can aggregate the IP-group information so that the firewall need only connect to OpenDaylight. By moving the information flow outside of the network elements to a centralized position, we reduce the overhead of the CPU consumption on the enforcement element. This is a huge savings - it allows the enforcement point to only have to make one connection rather than thousands, so it can concentrate on its primary job of forwarding and enforcing.

OpenDaylight can relay the binding information from one network element to others. Changes in group membership can be propagated more readily through a centralized model. For example, in a security application a particular host (e.g., user or IP Address) may be found to be acting suspiciously or violating established security policies. The defined response is to put the host into a different source group for remediation actions such as a lower quality of service, restricted access to critical servers, or special routing conditions to ensure deeper security enforcement (e.g., redirecting the host's traffic through an IPS with very restrictive policies). Updated group membership for this host needs to be communicated to multiple network elements as soon as possible; a very efficient and effective method of propagation can be performed using OpenDaylight as a centralized point for relaying the information.

OpenDaylight can create filters for exporting and receiving IP-SGT bindings used on specific peer groups, thus can provide more complex maintaining of policy groups.

Although the IP-SGT binding is only one specific piece of information, and although SXP is implemented widely in a single vendor's equipment, bringing the ability of OpenDaylight to process and distribute the bindings, is a very specific immediate useful implementation of policy groups. It would go a long way to develop both the usefulness of OpenDaylight and of policy groups.