CHAPTER 8

Cisco XML Security

Specific security privileges are required for a client application requesting information from the router.

This chapter contains these sections:

Authentication

User authentication through authentication, authorization, and accounting (AAA) is handled on the router by the transport-specific XML agent and is not exposed through the XML interface.

Authorization

Every operation request by a client application is authorized. If the client is not authorized to perform an operation, the operation is not performed by the router and an error is returned.

Authorization of client requests is handled through the standard AAA “task permissions” mechanism. The XML agent caches the AAA user credentials obtained from the user authentication process, and then each client provides these to the XML infrastructure on the router. As a result, no AAA information needs to be passed in the XML request from the client application.

Each object class in the schema has a task ID associated with it. A client application’s capabilities and privileges in terms of task IDs are exposed by AAA through a **show** command. A client application can use the XML interface to retrieve the capabilities prior to sending configuration requests to the router.

A client application requesting an operation through the XML interface must have the appropriate task privileges enabled or assigned for any objects accessed in the operation:

- <Get> operations require AAA “read” privileges.
- <Set> and <Delete> operations require AAA “write” privileges.

The “configuration services” operations through configuration manager can also require the appropriate predefined task privileges.

If an operation requested by a client application fails authorization, an appropriate <Error> element is returned in the response sent to the client. For “native data” operations, the <Error> element is associated with the specific element or object classes where the authorization error occurred.

Retrieving Task Permissions

A client application’s capabilities and privileges in terms of task permissions are exposed by AAA through CLI **show** commands. A client application can also use the XML interface to programatically retrieve the current AAA capabilities from the router. This retrieval can be done by issuing the appropriate <Get> request to the <AAA> component.

This example shows a request to retrieve all of the AAA configuration from the router:

Sample XLM Request to Retrieve AAA Configuration Information

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <AAA MajorVersion=”2” MinorVersion=”0”/>
    </Configuration>
  </Get>
</Request>
```

Sample XML Response from the Router

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <AAA MajorVersion=”2” MinorVersion=”0”>
        .
        .
        .
         AAA configuration returned here
        .
        .
        .
      </AAA>
    </Configuration>
  </Get>
  <ResultSummary ErrorCount="0"/>
</Response>
```

Task Privileges

A client application requesting a native data operation through the XML interface must have the appropriate task privileges enabled or assigned for any items accessed in the operation:

- <Get>, <GetNext>, and <GetVersionInfo> operations require AAA “read” privileges.
- <Set> and <Delete> operations require AAA “write” privileges.

The “configuration services” operations through the configuration manager can also require the appropriate predefined task privileges.

Task Names

Each object (that is, data item or table) exposed through the XML interface and accessible to the client application has one or more task names associated with it. The task names are published in the XML schema documents as <appinfo> annotations.

For example, the complex type definition for the top-level element in the Border Gateway Protocol (BGP) configuration schema contains this annotation:

```
<xsd:appinfo>
  <ObjectType>Container</ObjectType>
  <MajorVersion>18</MajorVersion>
  <MinorVersion>0</MinorVersion>
  <TaskIDInfo TaskGrouping="Single">
    <TaskName>bgp</TaskName>
  </TaskIDInfo>
  <Parents>
    <Parent>
      <Schema>native_data_operations</Schema>
      <Name>Configuration</Name>
    </Parent>
  </Parents>
</xsd:appinfo>
```

Here is another example from a different component schema. This annotation includes a list of task names.

```
<xsd:appinfo>
  <MajorVersion>1</MajorVersion>
  <MinorVersion>0</MinorVersion>
  <TaskIdInfo TaskGrouping="And">
    <TaskName>ouni</TaskName>
    <TaskName>mpls-te</TaskName>
  </TaskIdInfo>
</xsd:appinfo>
```

Task names indicate what permissions are required to access the data below the object. In the example, the task names `ouni` and `mpls-te` are specified for the object. The task names apply to the object and are inherited by all the descendants of the object in the schema. In other words, the task names that apply to a particular object are the task names specified for the object and the task names of all ancestors for which there is a task name specified in the schema.

The TaskGrouping attribute specifies the logical relationship among the task names when multiple task names are specified for a particular object. For example, for a client application to issue a <Get> request for the object containing the preceding annotation, the corresponding AAA user credentials must have read permissions set for both the `ouni` and `mpls-te` tasks (and any tasks inherited by the object). The possible values for the TaskGrouping attribute are And, Or, and Single. The value Single is used when there is only a single task name specified for the object.

Authorization Failure

If an operation requested by a client application fails authorization, an appropriate <Error> element is returned in the response sent to the client. For "native data" operations, the <Error> element is associated with the specific element or object where the authorization error occurred.

If a client application issues a <Get> request to retrieve all data below a container object, and if any subsections of that data require permissions that the user does not have, then an error is not returned. Instead, the subsection of data is not included in the <Get> response.

Management Plane Protection

Management Plane Protection (MPP) provides a mechanism for securing management traffic on the router. Without MPP, a management service's traffic can come through any interface with a network address, which could be a security risk.

MPP is effective when XML is configured.

Inband Traffic

To configure the MPP for inband traffic, use the command in this example:

```
RP/0/0/CPU0:router(config)#control-plane management-plane inband interface [interface
type] allow [protocol|all]
```

where the **protocol** is XML.

```
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$Ethernet 0/0/0/0 allow XML ?
  peer  Configure peer address on this interface
  <cr>
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$Ethernet 0/0/0/0 allow XML peer ?
  address  Configure peer address on this interface
  <cr>
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$Ethernet 0/0/0/0 allow XML peer address ?
  ipv4  Configure peer IPv4 address on this interface
  ipv6  Configure peer IPv6 address on this interface
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$Ethernet 0/0/0/0 allow XML peer address
```

Out-of-Band Traffic

To configure the MPP for out-of-band traffic, use the command in this example:

```
RP/0/0/CPU0:router(config)#control-plane management-plane out-of-band interface
[interface type] allow [protocol|all]
```

where the **protocol** is XML.

```
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$gabitEthernet 0/0/0/1 allow XML ?
  peer  Configure peer address on this interface
  <cr>
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$gabitEthernet 0/0/0/1 allow XML peer ?
  address  Configure peer address on this interface
  <cr>
```

```
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$ XML peer address ?
  ipv4  Configure peer IPv4 address on this interface
  ipv6  Configure peer IPv6 address on this interface
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#$ XML peer address
```

VRF

XML agents can be configured to virtual route forwarding (VRF) aware.

- To configure the dedicated agent [ssl] to receive or send messages through VRF, use this command:

  RP/0/0/CPU0:router(config)#**xml agent** [**ssl**] **vrf** <*vrf name*>

- To configure the dedicated [ssl] agent NOT to receive or send messages through the default VRF, use this command:

  RP/0/0/CPU0:Router(config)#**xml agent** [ssl] **vrf default shutdown**

Access Control List

To configure an access control list (ACL) for XML agents, use this command:

RP/0/0/CPU0:router(config)#**xml agent** [**ssl**] **vrf** <*vrf name*> **access-list** <*access-list name*>

IPv6 Access List Example

```
xml agent [ssl]
  vrf <vrf name>
     ipv6 access-list <ipv6 access-list name>
```

IPv4 and IPv6 Access Lists Example

```
xml agent [ssl]
  vrf <vrf name>
    ipv4 access-list <ipv4 access-list name>
    ipv6 access-list <ipv6 access-list name>
  !
!
```

Note This method to configure an IPv4 access-list is still supported (for backward compatibility) but hidden from CLI help.

```
xml agent [ssl]
  vrf <vrf name>
    access-list <ipv4 access-list name>
  !
!
```