NOMBRE: **Isidro Lara Lopez**
CARRERA: **INFRAESTRUCTURA DE REDES DIGITALES**
DOCENTE: **BARRON RODRIGUEZ GABRIEL**
NO. CONTROL: **1221100381**
FECHA: **14 DE DICIEMBRE DEL 2022**
GRUPO: **GIR0441**

Lab – raw NETCONF

Objectives

Part 1: Verify that NETCONF is Running on the IOS XE

Background / Scenario

In this lab, you will learn how to verify that the NETCONF service is running on the device by directly connecting to its port using an SSH client. You will be sending raw NETCONF Remote Procedure Calls encoded in XML structures.

Required Resources

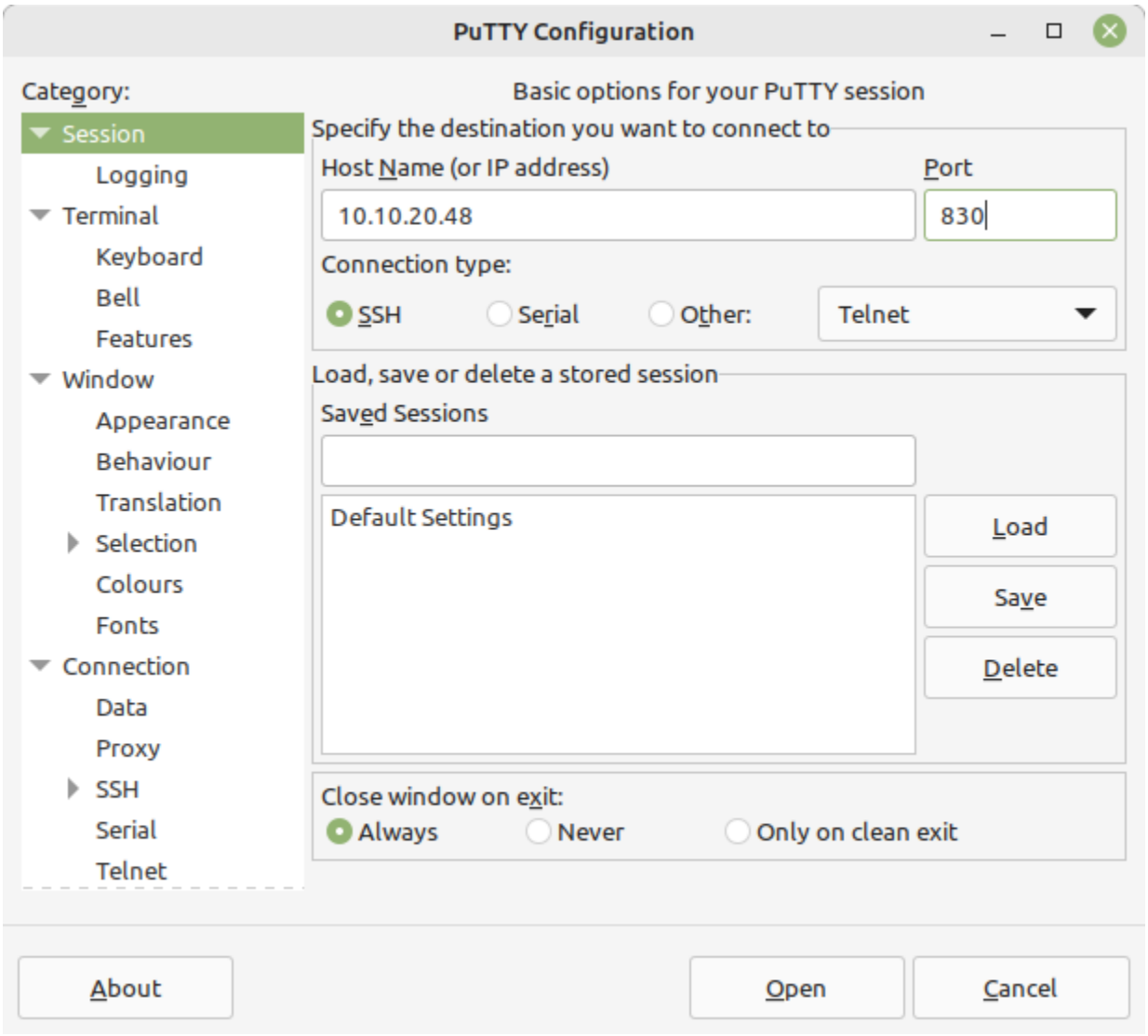
- Access to a router with the IOS XE operating system version 16.6 or higher
- Putty

Part 1: Verify that NETCONF is Running on the IOS XE

Step 1: Use Putty as an SSH client to connect to the NETCONF service.

a. Start Putty.

b. Using Putty, connect to host “192.168.56.101” (Adjust the IP address to match the router’s current address.) and port “830”.



c.

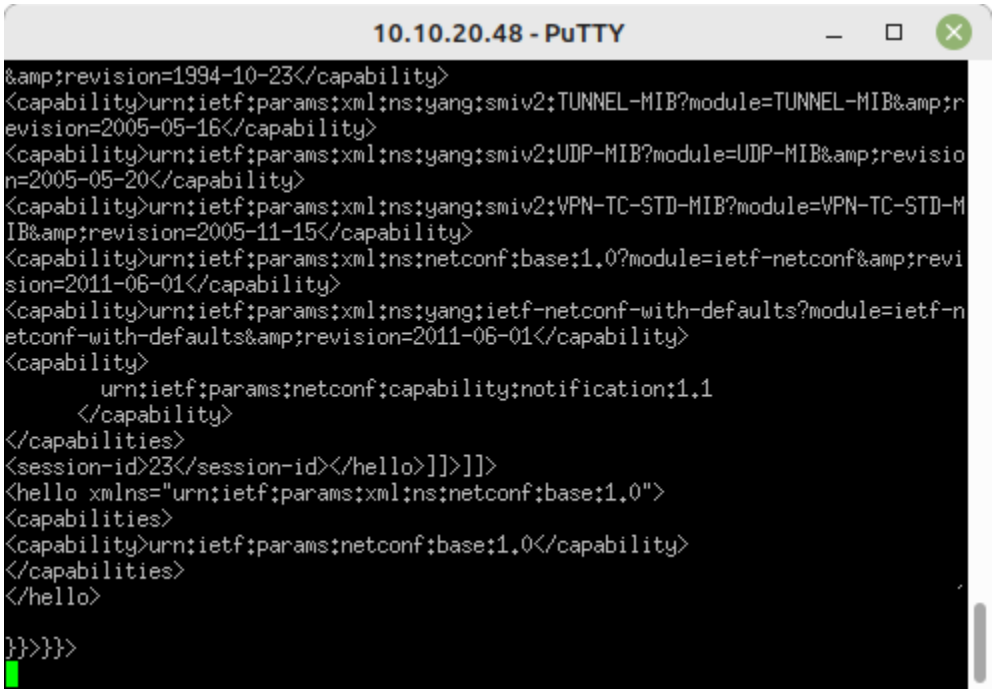
d. Login as “cisco” with the password “cisco123!” that was configured in IOS XE VM.

e. After a successful login to the NETCONF server, you should see a server “hello” message with an XML formatted list of supported YANG models (capabilities).

f. The end of the message is identified with “]]>]]>”.

g. To start a NETCONF session, the client needs to send its own hello message in a response:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
   <capability>urn:ietf:params:netconf:base:1.0</capability>
 </capabilities>
</hello>
]]>]]>
```



h. After the client hello message has been sent, the NETCONF session is ready to process RPC messages. For example, the following XML formatted RPC message will return the ietf-interfaces model data. Please

i. note that the returned XML data are designed to be consumed by an application. By default, this data might be difficult to for humans to read.

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
   </filter>
 </get>
</rpc>
]]>]]>
```

j. To close the NETCONF session, the client needs to send the following message:

```
<rpc message-id="9999999" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <close-session />
</rpc>
]]>]]>
```

Existen diferencias entre las peticiones que realizan los protocolos de NETCONF. Se utilize PuTTy porque es un servicio que nos permite conectarnos a dispositivos remotamente por SSH, Raw, Serial, Telnet, Rlogin.En este laboratorio se utilizaron 3 mensajes, el de hello para responderle el saludo al servidor y así poder enviar mensajes rcp que son comunicación en sistemas cliente-servidor, y para finalizar la sesión se envió un mensaje de cerrar sesión conel comando <close-session />

Otros tipos de mensajes que se pueden enviar son:

1; <rpc> y <rpc-reply>

El elemento <rpc> se utiliza para realizar acciones sobre el servidor que serán respondidas con un mensaje

con el elemento <rpc-reply>.

Cada elemento <rpc> o <rpc-reply> contiene un atributo <message-id> que identifica el

mensaje en esa

conexión. Una petición <rpc> con un determinado <message-id> recibirá un <rpc-reply>

con ese mismo valor

2; <rpc-error>

En caso de que ocurra algún error durante la ejecución en el servidor de una petición

<rpc>, el elemento

<rpc-reply> contendrá uno o varios elementos <rpc-error>

3; <ok>

El elemento <ok> se incluye dentro de <rpc-reply> cuando el mensaje no tiene que enviar ningún dato de respuesta

4; <lock> y <unlock>

Para esta operación, el elemento <lock> define un parámetro obligatorio, <target>, que sirve para identificar el almacén de datos de configuración (datastore) que queremos bloquear. Este parámetro existe de la misma forma para el elemento <unlock> y su uso es similar.

5; <get-config>

Esta operación se utiliza para obtener toda o parte de la configuración almacenada. Contiene dos elementos, el primero, <source>, hace referencia al almacén de datos de configuración al que nos referimos y el segundo, <filter>, se utiliza para indicar qué parte de la configuración queremos obtener.

Si no existe el elemento <filter>, obtendremos toda la información del fichero.

6; <copy-config>

Crea o reemplaza un almacén de datos de configuración con el contenido de otro.

7: <delete-config>

Elimina un almacén de datos de configuración.

8: <get>

Esta operación se utiliza para obtener información de estado del servidor y los datos de configuración actual del servidor. A diferencia de <get-config>, con <get> no podemos obtener datos de configuración de otros almacenes de datos de configuración

9: <close-session>

Operación para finalizar la sesión actual.

En esta práctica de laboratorio, aprendimos a verificar que el servicio NETCONF se esté ejecutando en el dispositivo conectándose directamente un cliente SSH. Envia llamadas de procedimiento remoto NETCONF sin procesar codificadas en estructuras XML.

Primero usamos putty para conectarnos al puerto 830 netconf del dispositivo.

Después enviamos el mensaje de saludo del cliente, la sesión NETCONF está lista para procesar mensajes RPC. Por ejemplo, el siguiente mensaje RPC con formato XML devolverá los datos del modelo yang de interfaces ietf. los datos XML devueltos están diseñados para ser consumidos por una aplicación de una forma predeterminada, estos datos pueden ser difíciles de leer para los humanos.

a. Los
b. mensajes RPC indican el tipo de mensaje que se está enviando, existen tres tipos, <rpc>, <rpc-reply> y <rpc-error>.
c. Una vez iniciada la conexión, a través de la capa de enlace, cliente y servidor se intercambian mensajes utilizando el elemento <hello>.
d. el elemento <rpc> se utiliza para realizar acciones sobre el servidor que serán respondidas con un mensaje
e. con el elemento <rpc-reply>.

f. Cada elemento <rpc> o <rpc-reply> contiene un atributo <message-id> que identifica el mensaje en esa
g. conexión. Una petición <rpc> con un determinado <message-id> recibirá un <rpc-reply> con ese mismo valor.

   Este ejemplo invoca la operación <get> sin parámetros.

h. En caso de que ocurra algún error durante la ejecución en el servidor de una petición <rpc>, el elemento
i. <rpc-reply> contendrá uno o varios elementos <rpc-error>.
j. Este mensaje <rpc-reply> puede contener varios elementos, <rpc-error>, describiendo cada uno de los
k. errores que se hayan producido, más solo es obligatorio que contenga uno de ellos.
l. El mensaje de error (rpc-error) contiene, a través de etiquetas, información de la capa en la que se ha
m. producido el error (error-type), de la causa del error(error-tag) y de la importancia (error-severity). Adicio-
n. nalmente existen etiquetas, error-app-tag, error-path, error-message y error-info que añaden información extra.
o. error-severity tiene dos posibles valores, error y warning, de momento solo se utiliza error y warning se reserva para un uso futuro