CONCLUSIÓN DE LABORTATORIOS.

Lab 2.1 - Setting Up the Lab Environment

Para mi caso que soy por el momento usuario 100% Linux se me fue difícil el utilizar le **VPN** de cisco que se utiliza para enlazar con el laboratorio **IOS XE on CSR**, así que tuve que hacer una **VIRTUALIZACIÓN** de una máquina con Windows 10 y para administrarlo y configurar con el servicio **SSH** o **TELNET** usaremos **PuTTy**.

IOS XE on CSR: En este caso describe al nombre del laboratorio que contiene al router CSRV1000 que contiene al SO hecho por cisco que cuenta con los protocolos NETCONF y RESTCONF protocolos creados por IETF y que sirven de herramienta para configurar, administrar y automatizar redes con “model drive programability”.

Virtualización: Mediante virtualbox nos permite la simulación de un sistema operativo real el cual nos permite usar un sistema operativo funcional que en este caso será Windows 10.

VPN: Virtual Private Network en palabra propias es un servicio crear un portal entre un punto a otro en cualquier parte del mundo, que en este caso nos hará ayudará a que podamos interactuar con el laboratorio sin tener los dispositivos físicamente esto no hay que confundirlo con la virtualización ya que elegimos la 3ra. Opción.

TELNET: Protocolo que permite el acceso remoto y traspaso de datos, pero es de una manera más insegura ya que los no se encripta el tránsito de los datos, ya que si aplicamos un análisis del tráfico de la red un ejemplo usando lo que es wireshark podríamos ver toda la información en crudo sin necesidad de realizar algo más, este protocolo funciona por default por el puerto 23.

SSH: Protocolo que permite el acceso remoto y traspaso de datos de manera seguro, la evolución de TELNET ya que este proporciona una encriptación de los datos usando el puerto 22.

PuTTy: Es un programa que permite la interacción con dispositivos mediante varios protocolos como telnet y ssh. Muy usado en la administración de redes y servidores.

2.2 Lab - CLI Automation with Python using netmiko.

Revisando la documentación de Netmiko EN Github se describe como un librería simplificada de conexión de dispositivos de redes CLI Multi-vendor.

Con este se pueden mostrar las configuraciones de dispositivos y modificarla en este caso por medio de scripts en Python.

La sintaxis para conexión y visualización es de la siguiente manera:

from netmiko import ConnectHandler -> Importación del modulo ConnectHandler

desde netmiko

```
sshCli = ConnectHandler {
    device_type= 'cisco_ios', -> Se selecciona el OS del vendor
    host=   '10.10.10.10',    -> Dirección IP del dispositivo a conectar
    username= 'test',         -> Se ingresa el username del dispositivo
    password= 'password',     -> Se ingresa el password del username
    port= 8022,               -> Puerto de conexión SSH o TELNET
    }

Comando para ejecución de comandos:

output = sshCli.send_command("comando disponible según el vendor")
print(output)
```

La sintaxis para conexión y configuración es de la siguiente manera:

from netmiko import ConnectHandler -> Importación del modulo ConnectHandler

desde netmiko

```
sshCli = ConnectHandler {
    device_type= 'cisco_ios', -> Se selecciona el OS del vendor
    host=   '10.10.10.10',    -> Dirección IP del dispositivo a conectar
    username= 'test',         -> Se ingresa el username del dispositivo
    password= 'password',     -> Se ingresa el password del username
    port= 8022,               -> Puerto de conexión SSH o TELNET
    }

config_commands = {
  'int loopback 2',
  'ip address 2.2.2.2 255.255.255.0'
}
```

config_commands: Nos permite realizar una serie de comandos según los predispuestos por el vendor que puedan ser tan complejas como uno tenga las posibilidades.

Comando para ejecución de comandos:

```
output = sshCli.send_command("comando disponible según el vendor")
print(output)
```

2.3 Lab - Explore YANG models using the pyang tool

Pyang: Es un validador, transformador y generador de código YANG, escrito en python. Se puede utilizar para validar módulos YANG y comprobar que son correctos, para transformar módulos YANG a otros formatos y para escribir plugins que generen código a partir de los módulos.

En este laboratorio utilizaremos pyang para analizar el archivo IOS XE VM que se encuentra disponible la descar desde github. Usando el comando del laboratorio le indicamos que nos muestre información con forma de arbol.

Otras comando para aplicar y usar yang son:

- **bin/** Scripts ejecutables.
- **pyang/** Contiene el código de la librería pyang.
- **pyang/init.py** Código de inicialización para la librería pyang.
- **pyang/context.py** Define la clase Context, que representa una sesión de parseo.
- **pyang/repository.py** Define la clase Repository, que se utiliza para acceder a los módulos.
- **pyang/syntax.py** Comprobación sintáctica genérica para sentencias YANG y YIN. Define expresiones regulares para la comprobación de argumentos de sentencias core.
- **pyang/grammar.py** Gramática genérica para YANG e YIN. Define chk_module_statements() que valida un árbol de análisis de acuerdo con la gramática.
- **pyang/statements.py** Define la clase genérica Statement y todo el código de validación.
- **pyang/yang_parser.py** tokenizador y analizador YANG.
- **pyang/yin_parser.py** Analizador sintáctico YIN. Utiliza la librería expat para el parseo XML.

- **pyang/types.py** Contiene código para comprobar los tipos incorporados.
- **pyang/plugin.py** API de plugins. Define la clase PyangPlugin de la que heredan todos los plugins. Todos los manejadores de salida se escriben como plugins.
- **pyang/plugins/** Directorio donde se pueden instalar los plugins. Todos los plugins de este directorio se inicializan automáticamente cuando se inicializa la biblioteca.
- **pyang/translators/** Contiene plugins de salida para la traducción de YANG, YIN y DSDL.

2.4 Lab - RESTCONF with Postman.

Ahora en el laboratorio pasamos a visualizar y configurar los dispositivos pero a diferencia del laboratorio 2.2 usaremos **RESTCONF** mediante el uso de la herramienta **Postman**. La cual mediante una **API** lograremos nuestro resultado. Este laboratorio a mi parecer tampoco fue dificil solo que hay que tener un poco de conocimiento acerca de Postman además que aunque estuviera bien la API y hubiera una conexión con el SDN, había veces que era necesario realizar las peticiones sean GET o PUT que son las dos que utilizamos en esta ocasión pero aparte de eso tenemos más funciones que desglosare en la parte inferior pero son las misma con las que interactuamos en las API's.

RESTCONF: Protocolo creado por IETF a diferencia de NETCONF nos permite realizar cambios en los dispositvos por medio de una API usando http/tls

Postman: Postman es una herramienta .

API: Herramienta de desarrollo web que nos permite interactuar con API's a diferencia de usar python este no nos permitira realizar al automatización de software.

2.5 Lab - RESTCONF with Python.

En este laboratorio a diferencia del anterior se utilizara python para realizar las consultas y no por medio de Postman para esto importaremos los modulos json y request además que deshabilitaremos las advertencias SSL . Igual como lo hicimos en la unidad anterior y con postman deberemos formar el request con la api, los encabezados y el tipo de autorización guardaremos la información proporcionada en una varaible que se le dara el formato de json y la imprimiremos. Luego pasaremos a usar la opción de yangConfig este es una opción que nos da request para realizar configuraciones con Yang. Luego usaremos un if para que dependiendo del resultado lanzado nos mandara el mensaje de repuesta deseado.

SSL: Protocolo para establecer un enlace autenticado y encriptados.

2.6 Lab – raw NETCONF.

En este laboratorio solo nos conectareos de diferente manera por medio de PuTTy cambiando el parametro de puerto de 22 a 830 **.**

2.7 Lab - NETCONF wPython List Capabilities.

En este caso usaremos el modulo de ncclient con el cual vamos a poder realizar peticiones con NETCONF, que es lo que realizamos en este laboratorio y devolviendonos el resultados en xml de manera cruda.

nnclient: ncclient es una libreria Python que facilita el desarrollo de aplicaciones y scripts del lado del cliente en torno al protocolo NETCONF.

Esta nos permite realizar interacción con equipos Cisco, Huawei y Juniper.

2.8 Lab - NETCONF wPython Device Configuration.

Nos quedamos en el anterior laboratorio nos pide copiar la salida y pasarlo al navegador de CodeBeauty con el cual nos mostrara el xml la salida de una forma organizada y en cascada. Esto solo es para mostrar un ejemplo de lo que nos dara el resultado de agregar la función toprettyxml() al código luego con lo que es la función netconf_data podremos ejecutar comandos de configuraciones del dispositivo, los comandos seran los que tenga disponible el dispositivo del vendor y la forma de estructurar esta orden debe de ser en xml. Por ejemplo nostros realizamos la acción de crear un nuevo host y una interfaz loopback pero esto puede ser cualquier comando que en este momento disponible que en este momento el vendor es Cisco. Pero se debe tener un conocimiento acerca de NETCONF. Además de que hablamos del Model Drive Programability una de sus caracteristicas es de las cuales si no acción es erronea con el dispositivo este devolvera los cambios cosa que lo vemos de ejemplo en el ultimo ejercicio.

CodeBeauty: Pagina con la que nos permite realizar un orden a nuestro codigo.

toprettyxml(): Función de nnclient que nos permite realizar un orden en la salida de nuestra petición y ahorrarnos el utilizar servicios de terceros.

Model Driven Programability: Modelo nuevo de redes en la cual define que mediante un APIC-EM pueda implementarse la programación en los dispositvos de la red.

2.9 Lab - NETCONF wPython Get Operational Data.

A diferencia del anterior laboratorio en el cual se nos puso a realizar configuraciones en este solo vamos a visualizarlas en este momento solo usaremos el xmlodict para hacer un filtrado de la información.

xmlodict: Es un módulo de Python que hace que trabajar con XML parezca trabajar con JSON, como en esta "especificación"