

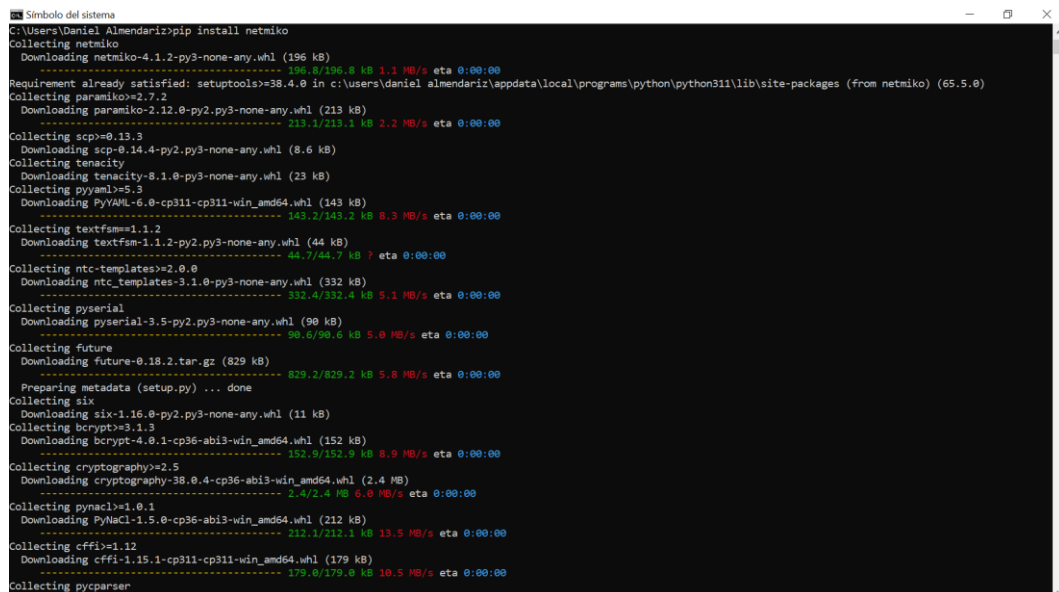
2.2 Lab - CLI Automation with Python using netmiko

Para la automatización de red simple utilizando una línea de comandos remota basada en telnet o ssh, los administradores de red han estado utilizando varias técnicas de raspado de pantalla durante un largo período de tiempo. Inicialmente, los scripts basados en "esperar" se utilizaban para automatizar la introducción de comandos cuando aparecía una cadena esperada específica en la línea de comandos. Con la evolución del lenguaje Python, el módulo netmiko Python ha surgido como un proyecto de código abierto alojado y mantenido en GitHub.com que proporciona una interfaz de automatización de red simple utilizando técnicas similares como los scripts basados en "esperar".

En esta actividad de laboratorio, identificará el potencial, pero también las limitaciones de usar netmiko para transportar comandos CLI para la automatización de redes.

Instalar el módulo netmiko Python

- Inicie un nuevo símbolo del sistema de Windows (cmd).
- Instalar netmiko usando pip en el símbolo del sistema de Windows: `pip install netmiko`
- Compruebe que netmiko se ha instalado correctamente. Inicie Python IDLE y en el shell interactivo intente importar el módulo netmiko:
- Importar** Netmiko



```
Símbolo del sistema
C:\Users\Daniel Almendariz>pip install netmiko
Collecting netmiko
  Downloading netmiko-4.1.2-py3-none-any.whl (186 kB)
-----
  126.8/126.8 kB 1.1 MB/s eta 0:00:00
Requirement already satisfied: setuptools>=38.4.0 in c:\users\daniel almendariz\appdata\local\programs\python\python311\lib\site-packages (from netmiko) (65.5.0)
Collecting paramiko>=2.7.2
  Downloading paramiko-2.12.0-py2.py3-none-any.whl (213 kB)
-----
  213.1/213.1 kB 2.2 MB/s eta 0:00:00
Collecting scp>=0.13.3
  Downloading scp-0.14.4-py2.py3-none-any.whl (8.6 kB)
Collecting tenacity
  Downloading tenacity-8.1.0-py3-none-any.whl (23 kB)
Collecting pyyaml>=5.3
  Downloading PyYAML-6.0-cp311-cp311-win_amd64.whl (143 kB)
-----
  143.2/143.2 kB 8.3 MB/s eta 0:00:00
Collecting textfsm>=1.1.2
  Downloading textfsm-1.1.2-py2.py3-none-any.whl (44 kB)
-----
  44.7/44.7 kB ? eta 0:00:00
Collecting ntc-templates>=2.0.0
  Downloading ntc_templates-3.1.0-py3-none-any.whl (332 kB)
-----
  332.4/332.4 kB 5.1 MB/s eta 0:00:00
Collecting pyserial
  Downloading pyserial-3.5-py2.py3-none-any.whl (90 kB)
-----
  90.6/90.6 kB 5.0 MB/s eta 0:00:00
Collecting future
  Downloading future-0.18.2.tar.gz (829 kB)
-----
  829.2/829.2 kB 5.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting bcrypt>=3.1.3
  Downloading bcrypt-4.0.1-cp36-abi3-win_amd64.whl (152 kB)
-----
  152.9/152.9 kB 8.9 MB/s eta 0:00:00
Collecting cryptography>=2.5
  Downloading cryptography-38.0.4-cp36-abi3-win_amd64.whl (2.4 MB)
-----
  2.4/2.4 MB 6.0 MB/s eta 0:00:00
Collecting pynacl>=1.0.1
  Downloading PyNaCl-1.5.0-cp36-abi3-win_amd64.whl (212 kB)
-----
  212.1/212.1 kB 13.5 MB/s eta 0:00:00
Collecting cffi>=1.12
  Downloading cffi-1.15.1-cp311-cp311-win_amd64.whl (179 kB)
-----
  179.0/179.0 kB 10.5 MB/s eta 0:00:00
Collecting pycparser
```

Conéctese al servicio SSH de IOS XE usando netmiko

Conéctese al servicio SSH de IOS XE usando netmiko.

El módulo netmiko proporciona una función `ConnectHandler()` para configurar la conexión ssh remota . Después de una conexión correcta, el objeto devuelto representa la conexión de la cli ssh al dispositivo remoto.

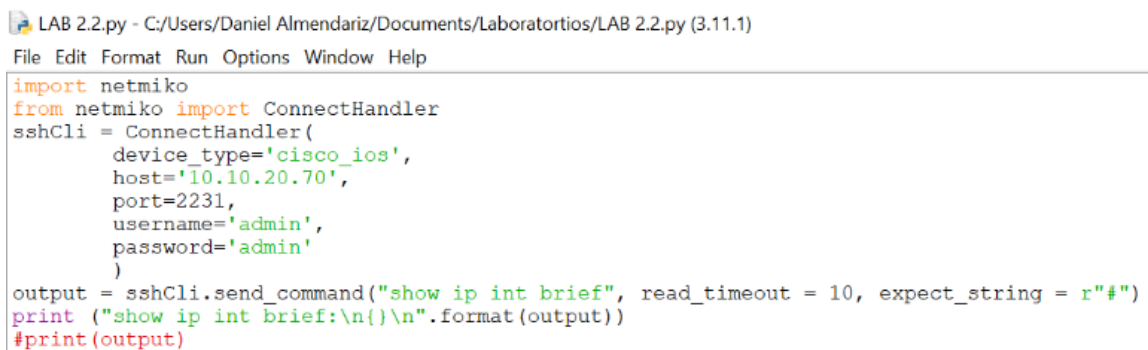
- En Python IDLE, cree un nuevo archivo de script de Python:
- En el nuevo editor de archivos de script de Python, importe la función `ConnectHandler()` desde el módulo netmiko :**desde** netmiko

import ConnectHandler

Configure un objeto de conexión `sshCli` mediante la función `ConnectHandler()` en el dispositivo IOS XE.

Los parámetros de la función `ConnectHandler()` son:

- `device_type`: identifica el tipo de dispositivo remoto
- `host`: la dirección (host o IP) del dispositivo remoto (ajuste la dirección IP "192.168.56.101" para que coincida con la dirección actual de su enrutador)
- `Puerto`: el puerto remoto del servicio SSH
- `nombre de usuario`: nombre de usuario ssh remoto (en este laboratorio "cisco" para eso se configuró en la máquina virtual de ios xe)
- `contraseña`: contraseña ssh remota (en este laboratorio "cisco123!" para eso se configuró en la máquina virtual xe de ios)



```
LAB 2.2.py - C:/Users/Daniel Almendariz/Documents/Laboratortios/LAB 2.2.py (3.11.1)
File Edit Format Run Options Window Help
import netmiko
from netmiko import ConnectHandler
sshCli = ConnectHandler(
    device_type='cisco_ios',
    host='10.10.20.70',
    port=2231,
    username='admin',
    password='admin'
)
output = sshCli.send_command("show ip int brief", read_timeout = 10, expect_string = r"#")
print ("show ip int brief:\n{}\n".format(output))
#print(output)
```

Usar netmiko para recopilar información del dispositivo

- Usando el objeto `sshCli`, devuelto por la función `ConnectHandler()` que representa la sesión remota de la cli `ssh`, envíe algún comando "show" e imprima la salida. Utilice la función `send_command()` del objeto `sshCli` con un parámetro de cadena que represente el comando que desea ejecutar en el modo `exec`:

```
LAB 2.2.py - C:/Users/Daniel Almendariz/Documents/Laboratorios/LAB 2.2.py (3.11.1)
File Edit Format Run Options Window Help

import netmiko
from netmiko import ConnectHandler
sshCli = ConnectHandler(
    device_type='cisco_ios',
    host='10.10.20.70',
    port=2231,
    username='admin',
    password='admin'
)

output = sshCli.send_command("show ip int brief", read_timeout = 10, expect_string = r"#")
print ("show ip int brief:\n()\n".format(output))
#print(output)
```

Enviar comandos show y mostrar la salida

```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help

Python 3.11.1 (tags/v3.11.1:1a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits()" or "license()" for more information.
>>> import netmiko
>>>

===== RESTART: C:/Users/Daniel Almendariz/Documents/Laboratorios/LAB 2.2.py =====
show ip int brief:

Thu Dec 8 01:54:38.751 UTC

Interface      IP-Address      Status      Protocol Vrf-Name
MgmtEth0/RP0/CPU0/0      192.168.122.31  Up          Up        default
GigabitEthernet0/0/0/0    unassigned      Shutdown    Down      default
GigabitEthernet0/0/0/1    unassigned      Shutdown    Down      default
GigabitEthernet0/0/0/2    unassigned      Shutdown    Down      default
GigabitEthernet0/0/0/3    unassigned      Shutdown    Down      default
GigabitEthernet0/0/0/4    unassigned      Shutdown    Down      default
```

Usar netmiko para modificar la configuración en el dispositivo

```
LAB 2.2 scrip.py - C:/Users/Daniel Almendariz/Documents/Laboratorios/LAB 2.2 scrip.py (3.11.1)
File Edit Format Run Options Window Help

import netmiko
from netmiko import ConnectHandler
sshCli = ConnectHandler(
    device_type='cisco_ios',
    host='10.10.20.70',
    port=2231,
    username='admin',
    password='admin'
)

config_commands = [
    'int loopback 1',
    'ip add 192.168.1.1 255.255.255.0',
    'no shutdown',
    'Prueba laboratorio 2'
]

sshCli.send_command('Configure terminal', expect_string = r"#", read_timeout=150)
sshCli.send_config_set("config_commands")
result = sshCli.send_command('show ip int brief')
print(result)
```

Comandos o variantes que se pueden utilizar:

La variable 'cfg_output' me mostrará lo que ocurrió durante esa sesión SSH. Una vez más, Netmiko hará una serie de cosas automáticamente; en particular, entrará y saldrá del modo de configuración (si es necesario).

Luego puede guardar su configuración en ejecución en la configuración de inicio ejecutando el método 'save_config ()'.

Netmiko es una biblioteca de código abierto diseñada para simplificar la administración de SSH en una amplia gama de dispositivos de red de varios proveedores, incluidos Cisco, Arista y Juniper Networks. Este video es uno de una serie producida por Bombal sobre la automatización de redes usando Python y GNS3.

Establece con éxito una conexión SSH al dispositivo. Simplifique la ejecución de los comandos show y la recuperación de datos de salida. Simplifique la ejecución de los comandos de configuración, incluidas posibles acciones de confirmación.

Asynssh es la biblioteca python ssh utilizada en Suzieq. Se conectó con éxito a las máquinas Juniper MX, Juniper QFX, Cisco 9K, Cumulus, Arista y SONIC sin ningún problema. Usamos textfsm internamente en los datos recopilados, si la salida estructurada no está disponible. Dados nuestros requisitos, esta fue la mejor opción.