ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

Configuración de una red LAN utilizando Python scripting para generar redes definidas por software (SDN).

Configuring of a LAN using Python scripting to generate software-defined networks (SDN).

L. Yulán[1], J. Pazmiño[1] y M. Saavedra[1]

1 Instituto Tecnológico Superior Quito Metropolitano. Carán N3-195 y Calle B (Nueva Tola 2) Quito, Ecuador.

lyulan@itsqmet.edu.ec, jupazmino@itsqmet.edu.ec, lsaavedra@itsqmet.edu.ec

Resumen: La gestión de redes es una tarea esencial en las organizaciones modernas, donde la complejidad de las infraestructuras tecnológicas demanda soluciones eficientes y precisas. Con dispositivos de red de fabricantes líderes como Cisco y Mikrotik. En respuesta a estos desafíos, la automatización de la configuración de dispositivos de red ha ganado popularidad como un enfoque para mejorar la eficiencia y reducir errores. Python, un lenguaje de programación ampliamente adoptado por su versatilidad y facilidad de uso, se ha convertido en una herramienta clave para esta automatización. Dentro de este ecosistema, Netmiko emerge como una librería especializada que facilita la interacción automatizada con dispositivos de red mediante SSH. Netmiko permite a los administradores de redes escribir scripts que pueden conectarse a dispositivos Cisco y Mikrotik, ejecutar comandos de configuración, y recuperar información sobre el estado de la red. Este enfoque no solo acelera el proceso de configuración, sino que también asegura la consistencia y precisión en la implementación de configuraciones, especialmente en entornos con múltiples dispositivos. A medida que las redes continúan creciendo en complejidad, la adopción de herramientas como Netmiko se vuelve esencial para mantener la operatividad y seguridad de las infraestructuras de red.

Palabras clave: Netmiko, Automatización de red, Python scripting, CISCO, MIkrotik.

ÉLITE 2024, VOL. (6). NÚM. (2)
ISSN: 2600-5875

Recibido: 04/09/2024
Revisado: 13/09/2024
Aceptado: 17/09/2024
Publicado: 27/09/2024

Abstract: *Network management is an essential task in modern organizations, where the complexity of technological infrastructures demands efficient and accurate solutions. With network devices from leading manufacturers such as Cisco and Mikrotik. In response to these challenges, automating the configuration of network devices has gained popularity as an approach to improve efficiency and reduce errors. Python, a programming language widely adopted for its versatility and ease of use, has become a key tool for this automation. Within this ecosystem, Netmiko emerges as a specialized library that facilitates automated interaction with network devices using SSH. Netmiko allows network administrators to write scripts that can connect to Cisco and Mikrotik devices, execute configuration commands, and retrieve information about the network status. This approach not only speeds up the configuration process, but also ensures consistency and accuracy in deploying configurations, especially in multi-device environments. As networks continue to grow in complexity, the adoption of tools such as Netmiko becomes essential to maintaining the operability and security of network infrastructures.*

Key words: *Netmiko, Network Automatization, Python scripting, CISCO, Mikrotik.*

ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

INTRODUCCIÓN

En el panorama actual de la administración de redes, la gestión eficiente de dispositivos de red es fundamental para garantizar la continuidad operativa y la seguridad de las infraestructuras tecnológicas en las organizaciones. Los routers, especialmente aquellos fabricados por empresas líderes como CISCO y Mikrotik, desempeñan un papel clave en la gestión del tráfico de datos, la segmentación de redes y la implementación de políticas de seguridad. Tradicionalmente, la configuración de estos dispositivos se ha realizado de manera manual, utilizando interfaces de línea de comandos (CLI), lo que requiere un alto nivel de conocimiento técnico y atención al detalle. Sin embargo, este enfoque manual es cada vez más insostenible en entornos modernos debido a la creciente complejidad de las redes y la necesidad de implementar cambios de manera rápida y precisa.

El uso de herramientas de automatización ha ganado tracción como una solución a estos desafíos. Python, un lenguaje de programación versátil y ampliamente adoptado en la comunidad tecnológica, junto con librerías especializadas como Netmiko, ofrece un enfoque robusto para automatizar tareas repetitivas y propensas a errores, como la configuración de dispositivos de red. Netmiko, en particular, se ha destacado por su capacidad para facilitar la conexión y la gestión de dispositivos de red a través de scripts, permitiendo a los administradores de redes ejecutar configuraciones y obtener información de los dispositivos de manera eficiente y escalable (Nugroho et al., 2020).

A pesar de los avances en la automatización de redes, muchas organizaciones continúan dependiendo de métodos manuales para configurar y administrar routers CISCO y Mikrotik. Este enfoque no solo consume tiempo, sino que también aumenta el riesgo de errores humanos, que pueden llevar a fallos en la red, vulnerabilidades de seguridad y tiempos de inactividad prolongados. En entornos donde la rapidez y la precisión son cruciales, la dependencia de la configuración manual limita la capacidad de las organizaciones para adaptarse a cambios rápidos en la demanda o a nuevas amenazas de seguridad. Además, la falta de estandarización en los procesos de configuración manual puede resultar en configuraciones inconsistentes entre diferentes dispositivos, lo que complica aún más la gestión y el mantenimiento de la red.

Este problema es particularmente evidente en organizaciones con redes extensas y distribuidas, donde la administración centralizada y automatizada se convierte en una necesidad para mantener la operatividad y seguridad. A medida que las redes continúan creciendo en tamaño y complejidad, la necesidad de soluciones de automatización efectivas se vuelve cada vez más apremiante.

La hipótesis central de este estudio es que la implementación de scripts de configuración automatizados reducirá significativamente el tiempo necesario para configurar dispositivos de red, disminuirá la incidencia de errores humanos y mejorará la consistencia en las configuraciones de red. Además, se espera que la automatización permita una respuesta más rápida y eficiente a las demandas cambiantes de la red y a las amenazas de seguridad

ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

emergentes, ofreciendo a las organizaciones una ventaja competitiva en términos de operatividad y seguridad.

Adicional, dentro este estudio realizará un enfoque metodológico en el desarrollo e implementación de un conjunto de scripts en Python, utilizando la librería Netmiko, para automatizar la configuración de routers CISCO y Mikrotik. Estos scripts estarán diseñados para realizar tareas comunes de configuración de manera automática, incluyendo la asignación de direcciones IP, la configuración de interfaces, la implementación de listas de control de acceso (ACL), y la configuración de protocolos de enrutamiento. Con estos resultados, se espera proporcionar una solución práctica y efectiva que mejore la administración de redes en el contexto de routers CISCO y Mikrotik, contribuyendo a futuras investigaciones dentro de las redes definidas por software (Mauboy & Wellem, 2022).

METODOLOGÍA

La metodología de Action Research (Investigación-Acción) es un enfoque que combina la investigación con la acción para resolver problemas en un contexto específico, muy utilizada en desarrollo de tecnologías, implementación de programas o sistemas (Ros-Sánchez et al., 2023). La cual servirá de base en la implementación de script con Python, pues se tiene que realizar varias pruebas y ejecuciones por medio de ciclos para tener un resultado optimo que se ajuste a la necesidad de la red de datos. La metodología investigación-acción es un proceso cíclico y colaborativo que involucra a investigadores y profesionales trabajando conjuntamente para abordar problemas mediante un conjunto de actividades que incluyen la identificación de problemas, la planificación, la implementación de soluciones (acción), su observación, y la reflexión sobre los resultados (Zuluaga Marín et al., 2022).

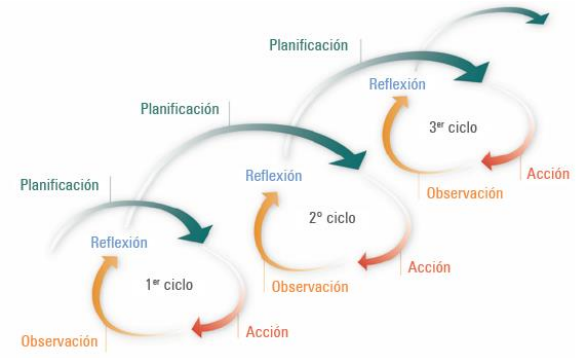

Figura 1. *Etapas de la metodología investigación-acción Fuente: Autoría propia.*

2.1 Fase I: Identificación de problemas

Es necesario establecer un entorno de trabajo controlado para probar las diferentes funcionalidades que se desarrollará dentro de la metodología que se va a aplicar, tales como: asignación de IP, configuración de ACL (Listas de control de Acceso, del inglés Access Control List) y protocolo de enrutamiento. Para ello, existen varios programas de simulación disponibles. En este caso, se utilizarán las IOS de CISCO y Mikrotik, las cuales se pueden descargar desde las plataformas oficiales. Estas emulan el hardware de un dispositivo y permiten ejecutar imágenes reales en un dispositivo virtual utilizando VMware (Choi, 2021).

2.2 Fase II-Planificación: Definición de topología e instalación de herramientas necesarias.

En la figura 2, se presenta el diagrama de los procedimientos que se va a seguir. El propósito principal es configurar de efectiva un equipo de red a lo largo de una red LAN. Para lograrlo, se parte de

ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

implementar una conexión SSH (Secure Shell) que contiene información sobre la conexión del equipo. Durante la validación de acceso al equipo, si el acceso es exitoso, se extraerá información crucial del equipo para construir e implementar los requerimientos de este, pero de forma automática por media de un resumen de comandos preestablecido dentro de un archivo formato txt (Elezi & Karras, 2023).

En una primera instancia, es necesario instalar Netmiko en un ambiente de Visual Studio Code preinstalado con el comando:

```
pip install Netmiko
```

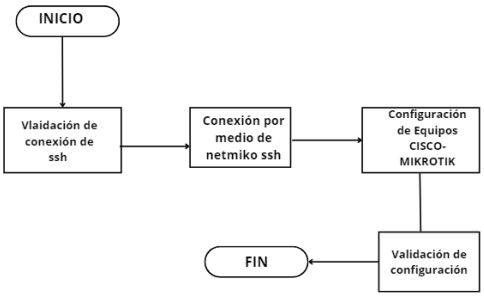

Figura 2. *Diagrama de configuración de equipos con librería Netmiko. Fuente: Autoría propia.*

Para obtener las imágenes de IOS es necesario ir a las páginas oficiales. En el caso de CISCO, es necesario crear una cuenta y para luego descargar la ova CR 10000V que es un router estándar y compatible con VMware:

https://software.cisco.com/download/home

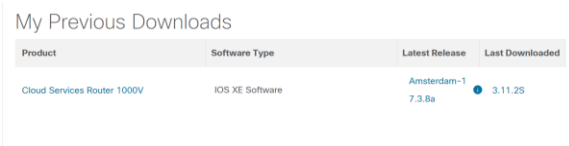

Figura 3 *.Obtención de OVA CISCO desde repositorio oficial. Fuente: Autoría propia.*

La ova Mikrotik de igual forma se la obtiene del siguiente enlace:

https://mikrotik.com/download

Para la descarga es necesario dirigirse al apartado de Cloud Hosted Router y seleccionar ova template, tal como se muestra en la figura 4. La versión seleccionada para este estudio fue 6.49.17 que es la más estable.

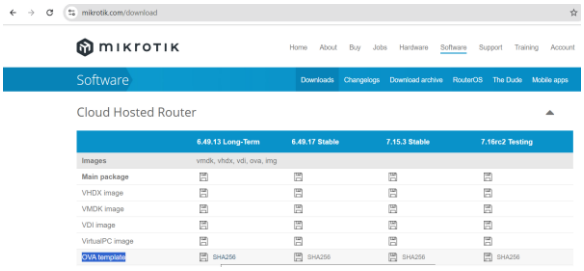

Figura 4. *Obtención de ova mikrotik de repositorio oficial. Fuente: Autoría propia.*

Una vez descargadas las dos OVAs, se procede a montarlas dentro de VMware Pro-17. La máquina en la que se realizarán las pruebas cuenta con las características detalladas en la tabla 1.

Tabla 1. Requerimientos mínimos para poder simular IOS en ambiente virtual.

Item	***Requerimiento***
Sistema Operativo	Windows 11
Procesador	Intel(R) Core (TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
Virtualización	Virtualización habilitada con hipervisor VMware 17 pro.
Memoria	16 GB RAM
Almacenamiento	SSD con al menos 20 GB libre

Una vez que se han obtenido las OVAs en las máquinas locales, estas deben ser montadas dentro del hipervisor. Para ello, es necesario acceder a la opción "Open a Virtual Machine" en VMware, ingresar el nombre y realizar la importación, tal como se ilustra en la figura 5 (Rahardika & Ratama, 2021).

ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

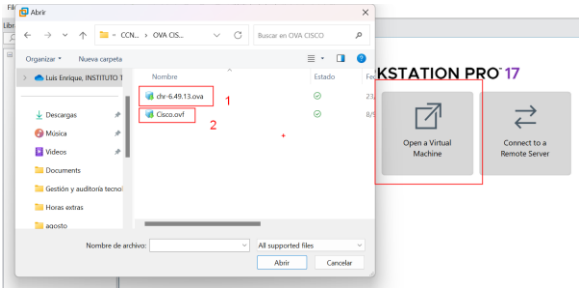

Figura 5 *Importación de ovas a VMware en máquina de prueba. Fuente: Autoría propia.*

Una vez realizado esto se tendrá creado dos máquinas virtuales que simularan el comportamiento de ambos routers. En el caso de Mikrotik, el protocolo SSH está configurado por defecto en su OVA debido a que tiene activo el segmento de SSH-server y la obtención de IP por medio de DHCP Client. Lo que permitiría la configuración por medio de la librería Netmiko de forma inmediata.

En cambio, en la emulación de CISCO, se debe configurar una dirección IP que permita el acceso vía SSH y de igual forma levantar el servicio vía comando a través de CLI. Para ello, se debe ingresar los siguientes comandos:

Router>enable

Este comando cambia el modo de usuario básico

Router#config terminal

Entra en el modo de configuración global (Global Configuration Mode), permitiéndo realizar configuraciones que afectan al sistema en su totalidad.

Router(config)#hostname R1

Cambia el nombre del router a "R2". El nombre de host es importante porque se utiliza en la generación de claves RSA para SSH.

R1(config)#ip domain-name itsqmet.edu.ec

Establece el nombre de dominio del router a "Itsqmet.edu.ec". Este nombre de dominio, junto con el nombre de host, se usa para generar el par de claves RSA necesario para SSH.

R1(config)#crypto key generate rsa

Genera un par de claves RSA que se utilizarán para la encriptación SSH. El router te pedirá que especifiques el tamaño de la clave, generalmente recomendada en 1024 o 2048 bits para mayor seguridad.

R1(config)#username lyulan secret Itsqmet1234

Crea un usuario llamado "lyulan" con una contraseña secreta encriptada "Itsqmet1234". Este usuario será utilizado para autenticarse cuando se establezca una conexión SSH.

R1(config)#line vty 0 4

Accede a la configuración de las líneas virtuales de terminal (VTY) del router. Estas son las líneas que se usan para la conexión remota vía SSH.

R1(config-line)#transport input ssh

Configura el router para aceptar únicamente por conexiones SSH a través de las líneas VTY, deshabilitando otros protocolos como Telnet.

R1(config-line)#login local:

Indica al router que utilice la base de datos local de usuarios para la autenticación en las líneas VTY.

R1(config)#ip ssh authentication-retries 2

Especifica que el router permitirá un máximo de 2 intentos de autenticación fallidos antes de cerrar la sesión SSH. Esto ayuda a prevenir ataques de fuerza bruta.

Una vez configurado el protocolo SSH, se realizó una prueba de conexión con la herramienta Putty dentro de la máquina local que contiene la OVA cisco. La conexión se estableció con normalidad, si se siguen los pasos anteriores, tal como se muestra en la figura 6.

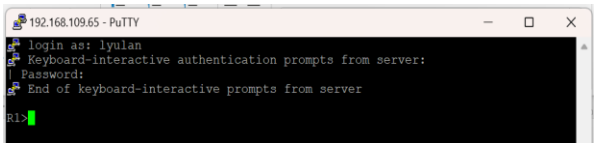

Figura 6. *Prueba de conexión por SSH con OVA cisco. Fuente: Autoría propia.*

ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

2.3 Fase III-Acción: Implementación de scripts en Python

Hay varios lenguajes de programación que se utilizan para la automatización de redes, como C/C++, Java, Python, Perl, entre otros. No obstante, el más destacado es Python, ya que es fácil de aprender y cuenta con numerosas referencias de código disponibles, lo que lo ha convertido en un estándar en la ingeniería de redes. Actualmente, existen herramientas para la gestión de redes desarrolladas en Python, entre las cuales destaca la librería Netmiko, que simplifican la conexión y gestión de dispositivos de red. Por todas estas razones, se decidió utilizar Python para desarrollar los scripts (Pazmiño et al., 2023).

Como primer script, se va a realizar la comunicación sencilla entre Netmiko y ambas marcas de routers.

Cisco

```
from netmiko import ConnectHandler

#JSON con parametros de conexion
cisco_device = {
    'device_type': 'cisco_ios',
    'ip': '192.168.103.79',
    'username': 'lyulan',
    'password': 'Itsqmet1234',
}
#Conexion con R1 via SSH
conexion = ConnectHandler(**cisco_device)
#Ejecución de comando en modo Privilegiado
salida = conexion.send_command('show ip interface brief')
print(salida)
#Cierre de conexion
conexion.disconnect()
```

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL   PUERTOS

GigabitEthernet3       unassigned      YES NVRAM  administratively down down
PS D:\Itsqmet\archivos_python> & C:/Python310/python.exe d:/Itsqmet/archivos_python/netcisco02.py
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       192.168.103.79  YES DHCP   up                    up
GigabitEthernet2       unassigned      YES NVRAM  administratively down down
GigabitEthernet3       unassigned      YES NVRAM  administratively down down
```

Figura 7. *Realización de consulta vía SSH con Netmiko en router Cisco. Fuente: Autoría propia.*

Tanto en la conexión cisco y MIkrotik, se debe crear un diccionario tipo Json donde se especifica los parámetros de conexión:

- **device_type:** El tipo de dispositivo, en este caso, un dispositivo Cisco IOS.
- **ip:** La dirección IP del dispositivo.
- **username:** El nombre de usuario para la autenticación.
- **password:** La contraseña para la autenticación por medio de SSH.

Métodos y Descripciones

Cuando los algoritmos se conectan a un dispositivo, los comandos que deben enviarse ya están predefinidos. Estos comandos se transmiten a la terminal de cada dispositivo vía SSH con la ayuda de ciertos métodos. La información resultante de cada comando se puede visualizar en la consola de Visual Studio Code. A continuación, se detallan cada uno de los métodos para la ejecución de comandos vía remota.

ConnectHandler(device):** Es utilizado para inicializar la comunicación y reconocer el tipo de dispositivo a configurar.

send_command(): Permite el envío de un comando a cualquier dispositivo de la red a través de la comunicación vía SSH.

send_config_set(): Permite la configuración de un dispositivo con órdenes más específicas. Es decir, realiza una configuración de un bloque de comandos, por ejemplo: Configuración de una interfaz, ACL o nateo.

send_config_from_file(): Permite el envío de comandos de configuración a los dispositivos remotos, cargados desde un archivo.

save_config(): Permite guardar la configuración en ejecución en la configuración inicial.

enable(): Permite ingresar al modo privilegiado en cisco

disconnect(): Realiza el cierre de la conexión una vez consultada la configuración o ejecutada el bloque de comandos.

ÉLITE **ÉLITE 2024** VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

Para la configuración de MIkrotik, se utiliza el Json especificando el modelo de software que es RouterOS junto con parámetros de usuario y password. Para realizar una consulta se utiliza el comando send_comand() que da como resultado la consulta de las direcciones IP mediante el ingreso del comando *ip address print*. El resultado de la consulta al dispositivo MIkrotik, se puede visualizar en la figura 8.

MIkrotik

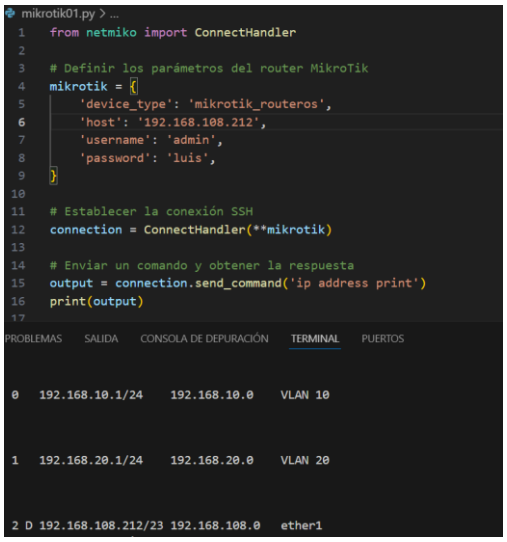

Figura 8. *Realización de consulta a router MIkrotik vía SSH con Netmiko. Fuente: Autoría propia.*

Configuración por bloques

Un router necesita realizar distintas configuraciones entre ellas: manipulación de interfaces de red, bloqueo de servicio, configuración de DHCP y salida a Internet por medio de un nat. Esto implicar ingreso de diferentes líneas de código para poder desplegar una configuración de forma correcta. Para, ello se requiere el uso del método **send_config_set()** que extraerá configuración de un arreglo. En los siguientes ejemplos se creará un servidor de DHCP en las interfaces de LAN en los routers cisco y MIkrotik.

Cisco

Figura 9. *Configuración de DHCP en router cisco. Fuente: Autoría propia.*

Como se puede observar en la figura 9, es necesario ingresar la contraseña de secret dentro de Json para poder ingresar al modo privilegiado, y una vez dentro de este se procede a ingresar al modo global mediante el método config_mode(). Para configurar un grupo de comandos es necesario la implementación de un arreglo en Python que será recorrido y a su vez va configurando dentro del router mediante el método sen_config_set(). Los resultados, se pueden observar tanto en la salida de cmd de Visual Studio Code como dentro de la configuración del router con el comando *show running-config*.

MIkrotik

Dentro de la configuración de MIkrotik, es un poco más sencillo pues no tiene diferentes modos de configuración. Este tiene un único modo general el cual acepta los comandos mediante bloque a través del

ÉLITE **ÉLITE 2024** VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

arreglo Python con método **send_config_set()**, tal como se muestra en la figura 10.

```python
from netmiko import ConnectHandler

# Definir los parámetros del router MikroTik
mikrotik = {
    'device_type': 'mikrotik_routeros',
    'host': '192.168.108.196',
    'username': 'admin',
    'password': 'luis',
}
# Establecer la conexión SSH
connection = ConnectHandler(**mikrotik)
connection.enable()
# Comandos para configurar DHCP
comandos = [
    'ip pool add name=dhcppool ranges=192.168.10.10-192.168.10.50',
    'ip dhcp-server network add address=192.168.10.0/24 gateway=192.168.10.1 dns-server=8.8.8.8',
    'ip dhcp-server add name=dhcpserver interface=ether2 address-pool=dhcppool disabled=no'
]

# Enviar un comando y obtener la respuesta
output = connection.send_config_set(comandos)
print(output)
# Cerrar la conexión
connection.disconnect()
```

Figura 10. *Configuración de DHCP en Router MIkrotik. Fuente: Autoría propia.*

Configuración por medio de archivo de texto

Python ofrece diversas maneras de manipular archivos de texto, principalmente a través de la función open (). Con esta función, es posible abrir un archivo en diferentes modos, como lectura ('r'), escritura ('w') o anexado ('a'). Una vez que el archivo está abierto, se puede leer su contenido utilizando métodos como read() o readlines(), y escribir en él con write() o writelines(). Es crucial cerrar el archivo después de manipularlo para liberar recursos, lo cual se puede lograr con close(). Sin embargo, una práctica recomendada es utilizar la declaración with, que cierra el archivo automáticamente al finalizar el bloque de código (Pazmiño et al., 2023). A continuación, se detalla un ejemplo de como abrir un archivo para obtener su contenido dentro de un arreglo.

```
with open('archivo.txt', 'r') as archivo:
    contenido = archivo.read()
    print(contenido)
```

Este arreglo permite ejecutar comandos de manera ordena dentro de la configuración de un script. Debido a que no se tiene sobrecargar de código visual y se puede ordenar sentencias dentro del archivo de texto por medio de varios bloques de comandos. Por lo tanto, hay que especificar el archivo de texto para que este se ajuste con las interfaces y comandos necesarios de acuerdo con la marca y modelo de un router.

Cisco

```python
from netmiko import ConnectHandler

cisco_01={

    'device_type': 'cisco_ios',
    'ip': '192.168.103.79',
    'username': 'lyulan',
    'password': 'Itsqmet1234',
    'secret': 'cisco'
}
#Conexion por ssh router cisco
connection = ConnectHandler(**cisco_01)
#Comando para pasar al modo Global
connection.config_mode()
#Extración de
with open('ejemploglobalnat.txt', 'r') as fichero:
    config_comandos = fichero.read().splitlines()

salida= connection.send_config_set(config_comandos)
print(salida)
connection.disconnect()
```

(variable) cisco_01: dict[str, str]

ejemploglobalnat.txt

```
hostname R1
ip dhcp excluded-address 192.168.10.1 192.168.10.10
ip dhcp pool LAN10
 network 192.168.10.0 255.255.255.0
 default-router 192.168.10.1
 dns-server 8.8.8.8
 exit
interface GigabitEthernet2
ip address 192.168.10.1 255.255.255.0
ip nat inside
exit
interface GigabitEthernet1
ip nat outside
no shutdown
exit
access-list 100 deny ip 192.168.10.0 0.0.0.255 192.168.20.0 0.0.0.255
access-list 100 permit ip 192.168.10.0 0.0.0.255 any
ip nat inside source list 100 interface GigabitEthernet1 overload
ip route 0.0.0.0 0.0.0.0 192.168.1.1
```

Figura 11 . *Ejecución de bloque de código para salida a Internet en router cisco. Fuente: Autoría propia.*

Dentro de la figura 11, se configuró un router desde un archivo de texto donde se especifica de forma correcta los bloques de comando; es decir su entrada a configuración junto con su salida de ser necesario al ejecutar un exit. La configuración de este abarcó desde el nombre “R1” y estableció un servidor DHCP que excluye las direcciones IP del rango 192.168.10.1 a 192.168.10.10. Se creó un pool DHCP llamado “LAN10” con la red 192.168.10.0/24, asignando la dirección 192.168.10.1 como puerta de enlace predeterminada y el servidor DNS 8.8.8.8. La interfaz GigabitEthernet2 se configura con la dirección IP

ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

192.168.10.1/24 y se designa como “ip nat inside” que indica que es la parte LAN. La interfaz GigabitEthernet1 se configura como “ip nat outside” comprendida como WAN y se activó el comando “no shutdown” en ambas interfaces para desplegar la configuración. Se estableció una lista de acceso (ACL), que permite todo el tráfico restante desde 192.168.10.0/24. Finalmente, se configura la traducción de direcciones de red (NAT) para usar la lista de acceso 100 en la interfaz GigabitEthernet1 con sobrecarga, y se añade una ruta estática predeterminada hacia 192.168.103.1.

MIkrotik

```python
from netmiko import ConnectHandler

# Definir los parámetros del router MikroTik
mikrotik = {
    'device_type': 'mikrotik_routeros',
    'host': '192.168.103.221',
    'username': 'admin',
    'password': 'luis',
}
# Establecer la conexión SSH
connection = ConnectHandler(**mikrotik)
connection.enable()
# Comandos para configurar mediante archivo
with open('confmk.txt', 'r') as fichero:
    comandos = fichero.read().splitlines()

# Enviar un comando y obtener la respuesta
output = connection.send_config_set(comandos)
print(output)
# Cerrar la conexión
connection.disconnect()
```

```
ROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL   PUERTOS

rade disabled=no
admin@MikroTik] > ip firewall nat add chain=srcnat src-address=192.168
sabled=no
S D:\Itsqmet\archivos_python>
```

confmk.txt

```
interface set ether1 name=INET
interface set ether2 name=LAN
ip dhcp-client add interface=INET disabled=no
ip address add interface=LAN address=192.168.10.1 netmask 255.255.255.0 disabled=no
ip pool add name=DHCP ranges=192.168.10.10-192.168.10.100
ip dhcp-server add name=DHCP interface=LAN address-pool=DHCP lease-time=259200 bootp-lease-time=forever authoritative
ip dhcp-server network add address=192.168.10.0/24 gateway=192.168.10.1 dns-server=8.8.8.8
ip firewall nat add chain=srcnat src-address=192.168.10.0/24 dst-address=0.0.0.0/0 out-interface=INET action=masquera
```

Figura 12. *Ejecución de bloque de código para dar salida a Internet en MIkrotik. Fuente: Autoría propia.*

Como se puede observar en la figura 12, la configuración se ejecuta correctamente y da salida a Internet a una red 192.168.10.0/24 de la misma forma que el router cisco, pero esta vez utilizando la acción de **masquerade** que es propia de un sistema Linux que tiene como base el RouteOS de MIkrotik.

2.4 Fase IV: Observación y reflexión de resultados

El uso de códigos para automatizar configuraciones en dispositivos de red, como MIkrotik y Cisco, mediante herramientas como Netmiko y archivos TXT, es altamente eficiente. Durante la elaboración de códigos y su utilización para dar salida a Internet se pudo evidenciar que, al ejecutar múltiples comandos de manera rápida y precisa, se ha reducido significativamente los errores humanos y el tiempo necesario para la configuración manual. Además, esto facilitaría la gestión de grandes redes al aplicar configuraciones consistentes en múltiples dispositivos en tan poco tiempo, mejorando la escalabilidad y la eficiencia operativa. Sin embargo, es crucial manejar estos archivos con cuidado para evitar problemas de seguridad y mantenerlos actualizados con las mejores prácticas y cambios en la red tomando en cuenta el modelo y versión de las IOS para cada uno de los routers.

RESULTADOS Y DISCUSIÓN

Netmiko es suficientemente rápido para la mayoría de las tareas de configuración automatizadas. Permite la ejecución de múltiples comandos en secuencia y es capaz de manejar múltiples conexiones simultáneas, lo que puede ser útil en redes más grandes. Como resultados se tiene que el tiempo necesario para realizar la misma tarea manualmente varía según el modelo del equipo y la habilidad de cada trabajador al momento de configurar una salida a Internet por equipo. Después de realizar pruebas de gestión manual durante esta investigación, se determinó el tiempo promedio por modelo de equipo para obtener los

ÉLITE 2024 VOL 6. NUM 2. ISSN: 2600-5875
DOI: https://doi.org/10.69603/itsqmet.vol6.n2.2024.87

mismos 4 parámetros, tales como: Configurar Interfaces, creación de DHCP en un router, reglas de control por ACL y nateo hacia la interfaz de WAN tardan en configurar manual un aproximado de 25 min. Comparando este tiempo versus la ejecución que le toma al script que en aproximado es 12 segundos, se puede medir la eficiencia que con la ecuación siguiente:

$$\%Ef=\left(1-\frac{Tiempo\ de\ ejecución\ script}{Tiempo\ de\ ejecución\ manual}\right)x100\%$$

$$\%Ef=\left(1-\frac{12\ seg}{25mim.\frac{60\ seg}{1\ mim}}\right)x100\%$$

$$\%Ef=(1-0{,}008)\ x100\%$$

$$\%EF=99{,}2\ \%$$

Se demuestra que la ejecución de un script es un 99,2% más rápido que el proceso manual. Al extrapolar estos resultados y aplicarlos a la problemática de la empresa proveedora de servicios de internet relacionada con este esta investigación y aplicando las mismas configuraciones; que son las de dar salida a Internet a una red LAN.

Con los resultados obtenidos, se puede afirmar que los códigos desarrollados reducirán el tiempo necesario para realizar el mismo proceso manual en al menos 10 veces. La eficiencia ha dado una gran visión que permita elaborar topologías de red más complejas donde se involucren protocolos, redes VPN y configuración de infraestructura por medio de MPLS. Donde se vería mucho más el beneficio la reducción del tiempo de ejecución (Ibañez Moreno & Pazmiño Quiñonez, 2020).

Para futuras investigaciones, se puede incluir el uso de redes neuronales que consulten el estado de otros routers y en base a eso se reconfiguren para tener un mejor enrutamiento de paquetes. También, se podría crear notificaciones de alerta vía Telegram donde se indique fallos en la red de datos o si un servidor no tiene respuesta. Permitiendo, reducir los tiempos de respuestas a incidente.

CONCLUSIONES

Netmiko es una herramienta eficiente y bastante efectiva para la configuración de routers en la mayoría de los casos. Su simplicidad y versatilidad lo hacen ideal para tareas de automatización cotidianas y para redes de tamaño pequeño a mediano. Sin embargo, para entornos de red muy grandes, altamente sensibles al rendimiento o con requisitos muy específicos, puede que sea necesario considerar otras herramientas o enfoques más especializados.

En la parte práctica, específicamente en el primer script, se pueden incluir directamente los comandos necesarios para configurar y visualizar los elementos de la red que se deseen ajustar o verificar. Sin embargo, en el ámbito profesional, es más común y recomendable lo realizado en el segundo y tercer script que son vía archivo de configuración. En este caso, se tiene los comandos de configuración de red básicos para dar salida a Internet a una red LAN. Además, el tercer script utiliza comandos que permiten la ejecución desde archivo, el cual pueden contener tantos comandos como el usuario considere necesario.

El tiempo necesario para obtener los parámetros de red, y configurar los equipos se pudo evidenciar que se redujo en al menos un 99% del tiempo de ejecución manual en un proceso sencillo. La librería presenta una buena respuesta en tiempo de ejecución.

CONTRIBUCIONES DE LOS AUTORES:

La elaboración de esta investigación contó con la colaboración de varios autores durante todo el proceso de estructuración del artículo. A continuación, se detalla la contribución de cada uno de los autores relevantes en este trabajo:

En la parte de conceptualización, interpretación y metodología, contribuyeron, J. Pazmiño y M. Saavedra; software y validación, L. Yulán y J. Pazmiño, para la parte de análisis formal, investigación y recursos, M. Saavedra y L. Yulán; curación de datos.

FINANCIAMIENTO:

Cabe mencionar que esta investigación no se tuvo financiamiento externo para la realización de esta.

CONFLICTOS DE INTERÉS:

Los autores declaran no tener conflicto de interés al utilizar esta investigación para la elaboración de otras investigaciones siempre y cuando este sea citado de forma correcta.

REFERENCIAS

- Choi, B. (2021). Python Network Automation Labs: SSH paramiko and netmiko. In *Introduction to Python Network Automation*. https://doi.org/10.1007/978-1-4842-6806-3_14
- Elezi, A., & Karras, D. (2023). On automating network systems configuration management. *CRJ*. https://doi.org/10.59380/crj.v1i1.639
- Ibañez Moreno, C. D., & Pazmiño Quiñonez, J. C. (2020). *Diseño de una red con DMVPN sobre la red MPLS de Puntonet*.
- Mauboy, L. G., & Wellem, T. (2022). Studi Perbandingan Library Untuk Implementasi Network Automation Menggunakan Paramiko Dan Netmiko Pada Router Mikrotik. *JURIKOM (Jurnal Riset Komputer)*, *9*(4). https://doi.org/10.30865/jurikom.v9i4.4420
- Nugroho, K., Abrariansyah, A. D., & Ikhwan, S. (2020). Performance Comparison between Paramiko and Netmiko Libraries in Network Automation Process. *InfoTekJar : Jurnal Nasional Informatika Dan Teknologi Jaringan*, *5*(1).
- Pazmiño, J., Suarez, D., Saavedra, M., & Perez, A. (2023). Enhancement of comprehensive early childhood development 1 Instituto Tecnológico Superior Quito Metropolitano. Carán N3-195 y Calle B (Nueva Tola 2) Quito. *ÉLITE 2023 VOL 5. NUM 2. ISSN: 2600-5875* , *5*(2).
- Rahardika, D., & Ratama, N. (2021). Implementasi Network Automation untuk Konfigurasi Jaringan Baru dengan Netmiko. *Journal of Artificial Intelligence and Innovative Applications (JOAIIA)*, *2*(3).
- Ros-Sánchez, T., Abad-Corpa, E., López-Benavente, Y., & Lidón-Cerezuela, M. B. (2023). Participatory Action Research on empowerment in older women: A theoretical-methodological analysis. *Enfermería Clínica (English Edition)*, *33*(2). https://doi.org/10.1016/j.enfcle.2021.10.008
- Zuluaga Marín, M., Botero Suaza, J. C., Martínez Romero, A. M., & Lopera Ortega, Y. (2022). Neurodidáctica y pensamiento crítico: perspectivas para la educación actual. *Educación y Educadores*, *25*(2). https://doi.org/10.5294/edu.2022.25.2.2