Instituto Politécnico Nacional

IPN

Escuela Superior de Cómputo

ESCOM

Academia de Redes Aplicaciones para comunicaciones en red

6CV2

Práctica 17

“Ejemplo Telnet en Python”

Alumna:

Navarrete Becerril Sharon Anette

Fecha de entrega: “25 - Noviembre - 2024”

Profesor: Ojeda Santillan Rodrigo

OBJETIVO

Desarrollar un cliente y un servidor Telnet en Python utilizando la biblioteca socket para comprender el funcionamiento básico de este protocolo en la autenticación y la administración remota de sistemas. Esta práctica busca implementar un sistema donde el servidor reciba y valide credenciales enviadas por el cliente, mostrando un mensaje de "Autenticación exitosa" o "Autenticación fallida" según corresponda.

INTRODUCCIÓN

Telnet es un protocolo de red que permite el acceso remoto a sistemas mediante una conexión de texto simple, proporcionando una interfaz de línea de comandos para interactuar con el servidor. Aunque su uso ha sido reemplazado en gran medida por protocolos más seguros, como SSH, Telnet sigue siendo una herramienta útil para el aprendizaje en redes y administración remota. En este ejemplo, se emplea la biblioteca socket de Python, la cual facilita la creación de conexiones de red a través del modelo de sockets.

La biblioteca socket permite la comunicación entre aplicaciones, ya sea en la misma máquina o entre diferentes máquinas conectadas a la red. En este contexto, se utilizará para construir tanto el servidor Telnet, que escucha las conexiones y recibe las credenciales, como el cliente Telnet, que envía los datos al servidor para autenticarse. Al realizar esta implementación, se busca comprender cómo funciona Telnet en un entorno de bajo nivel, manejando el intercambio de datos en texto y la autenticación mediante Python.

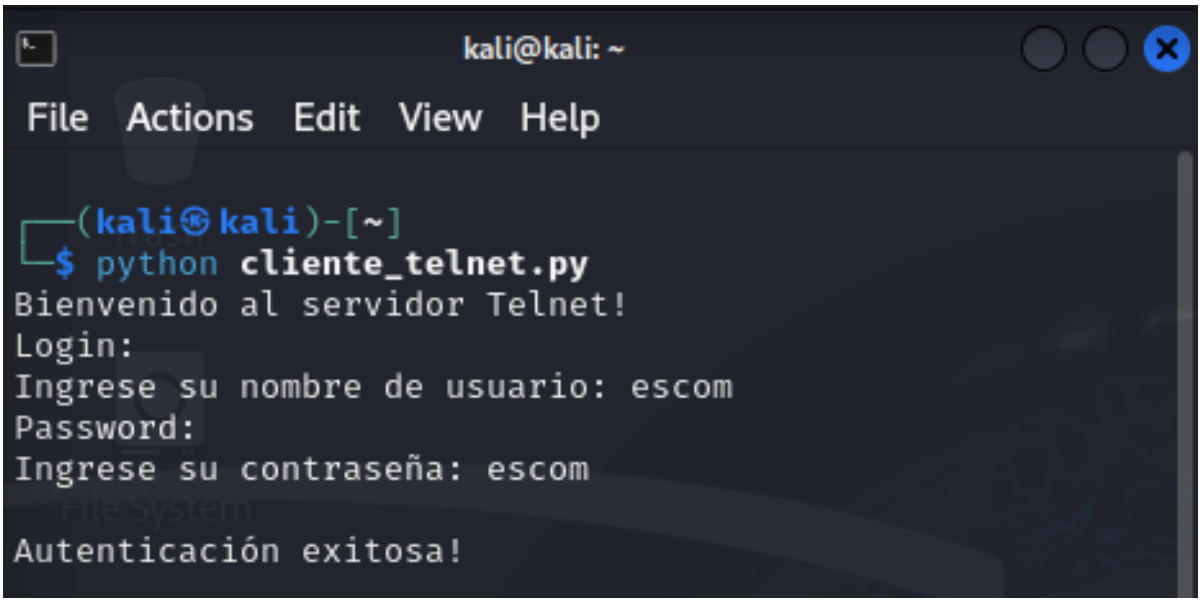
DESARROLLO

Para esta práctica, se agrega la siguiente lógica del lado del cliente_telnet.py:

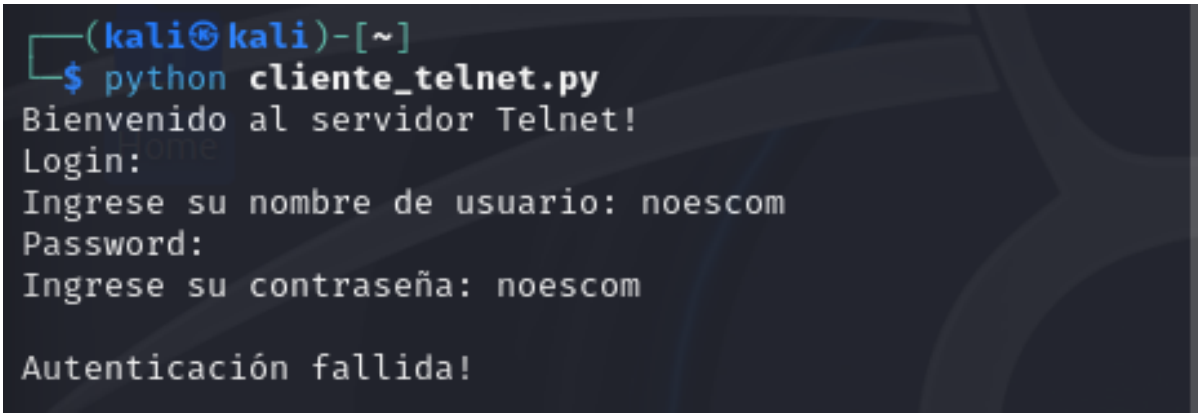
```
import socket

# Configuración del cliente
HOST ='127.0.0.0'  # IP de tu servidor>
PORT = 5000  # Puerto configurado>

# Conectar al servidor
client_socket = socket.socket(socket.AF_INET, socket.S>
client_socket.connect((HOST, PORT))

# Recibir el mensaje de bienvenida
print(client_socket.recv(1024).decode('utf-8'))

# Enviar nombre de usuario
username = input("Ingrese su nombre de usuario: ")
client_socket.sendall(username.encode('utf-8') + b"\n">

# Recibir la solicitud de contraseña
```

```
print(client_socket.recv(1024).decode('utf-8'))

# Enviar la contraseña
password = input("Ingrese su contraseña: ")
client_socket.sendall(password.encode('utf-8') + b"\n">

# Recibir el resultado de la autenticación
print(client_socket.recv(1024).decode('utf-8'))

client_socket.close()
```

Al ejecutar y realizar las pruebas se puede ver el siguiente mensaje del lado del cliente para el caso de autenticación exitosa:

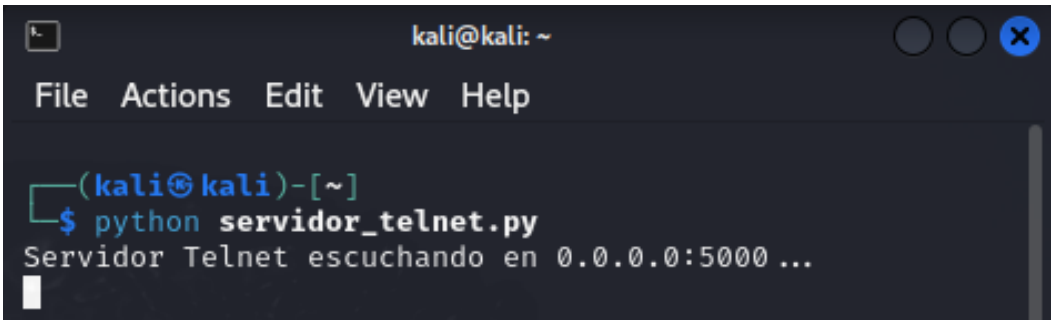


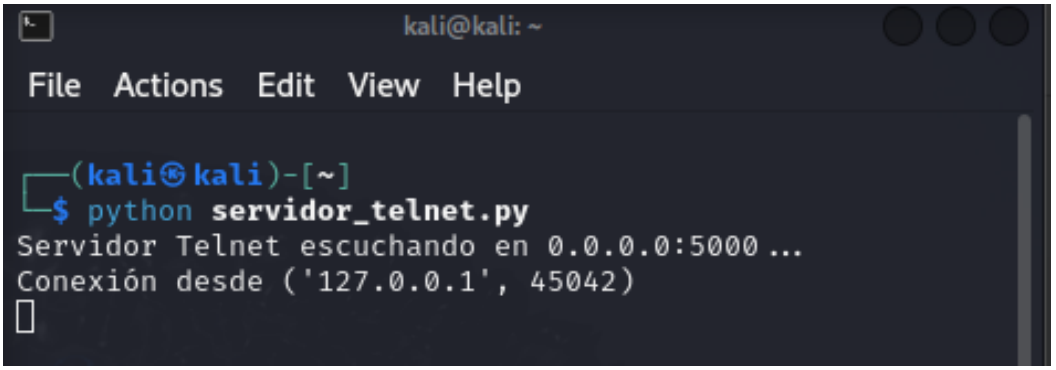
Para el caso de la autenticación fallida se puede ver el siguiente mensaje:



Para el servidor_telnet.py se va a agregar la siguiente lógica para poder recibir las credenciales y auténticas:

```
import socket

# Configuración del servidor
HOST = '0.0.0.0'  # Escuchar en todas las interfaces d>
PORT = 5000       # Puerto arbitrario que no requiere >

# Crear el socket del servidor
server_socket = socket.socket(socket.AF_INET, socket.S>
server_socket.bind((HOST, PORT))
server_socket.listen(1)
print(f"Servidor Telnet escuchando en {HOST}:{PORT}...>

while True:
    client_socket, client_address = server_socket.acce>
    print(f"Conexión desde {client_address}")

    client_socket.sendall("Bienvenido al servidor Teln>

    # Recibir nombre de usuario
    username = client_socket.recv(1024).strip().decode>

    # Pedir contraseña
    client_socket.sendall("Password: ".encode('utf-8')>
    password = client_socket.recv(1024).strip().decode>

    # Autenticación simple
    if username == 'escom' and password == 'escom':
        client_socket.sendall("\nAutenticación exitosa>
    else:
        client_socket.sendall("\nAutenticación fallida>

    client_socket.close()
```





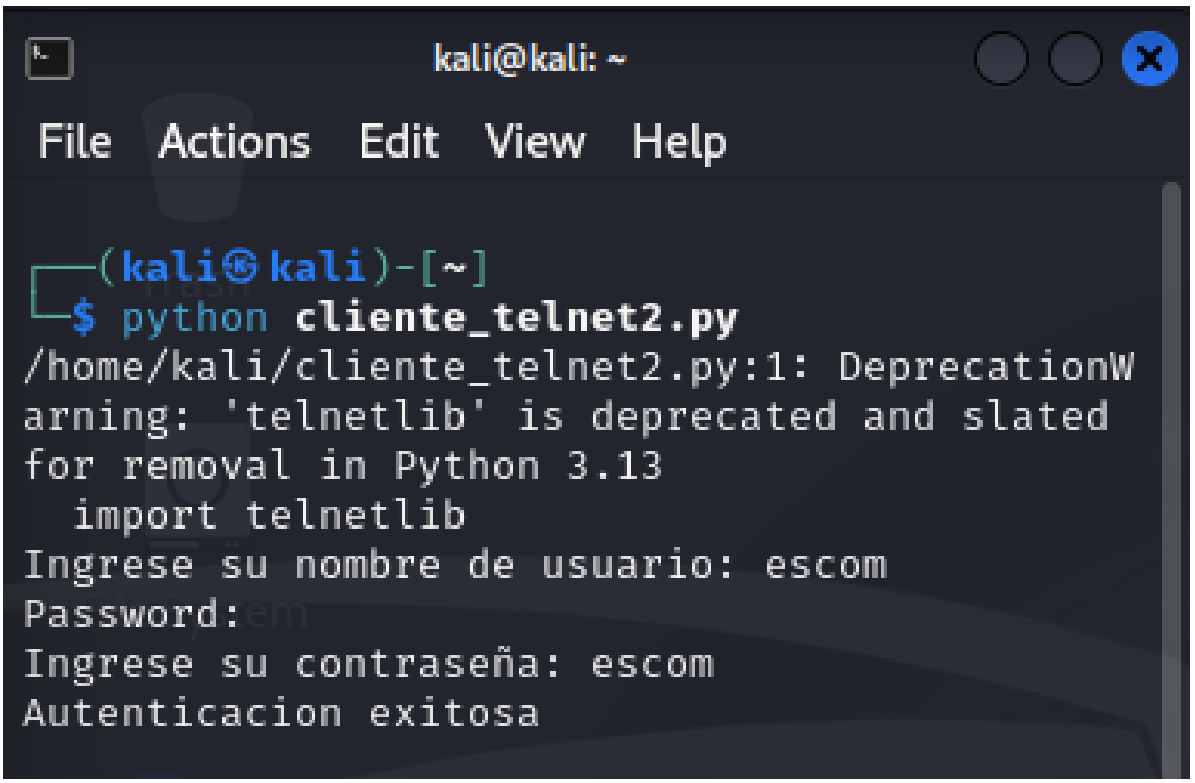
ACTIVIDAD:

Realización del mismo procedimiento de autenticación, pero usando la biblioteca telnetlib.

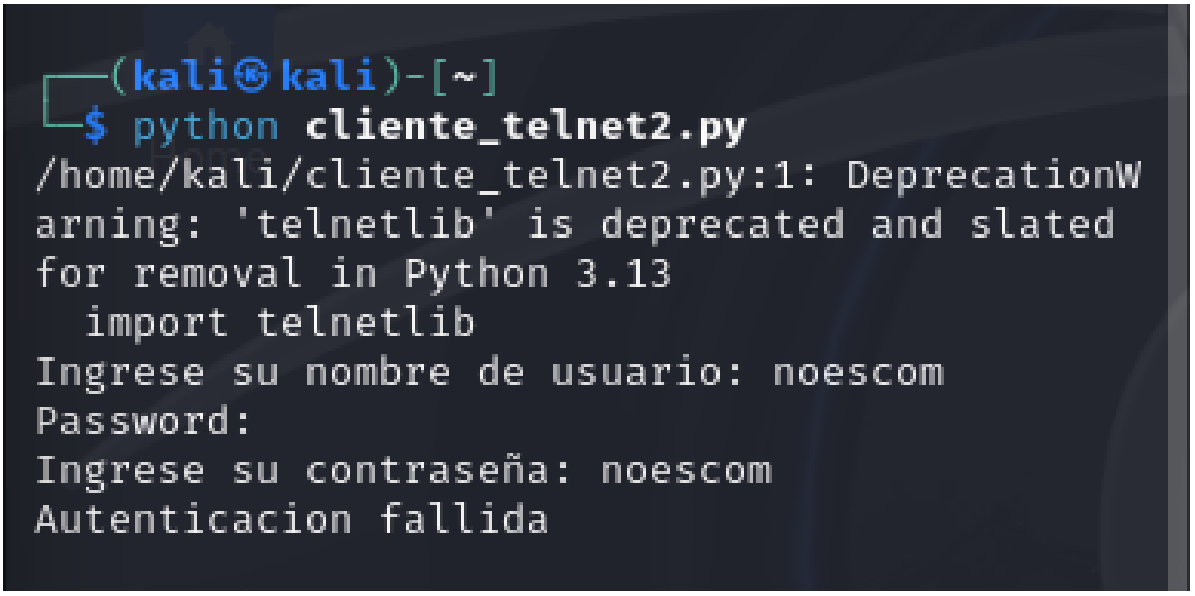
Para esta práctica, se agrega la siguiente lógica del lado del cliente_telnet2.py:

```
import telnetlib

# Configuración del cliente
HOST = '127.0.0.0'  # IP de tu servidor
PORT = 5001         # Puerto configurado

# Crear conexión Telnet
try:
    tn = telnetlib.Telnet(HOST, PORT)

    # Recibir el mensaje de bienvenida
    tn.read_until(b"\n").decode('utf-8')

    # Enviar nombre de usuario
    username = input("Ingrese su nombre de u>
    tn.write(username.encode('utf-8') + b"\n>

    # Recibir la solicitud de contraseña
    print(tn.read_until(b"\n", timeout=1).de>

    # Enviar la contraseña
    password = input("Ingrese su contraseña:>
    tn.write(password.encode('utf-8') + b"\n>

    # Recibir el resultado de la autenticaci>
    print(tn.read_all().decode('utf-8'))

except Exception as e:
    print(f"Error al conectar con el servido>
```

Al ejecutar y realizar las pruebas se puede ver el siguiente mensaje del lado del cliente para el caso de autenticación exitosa:



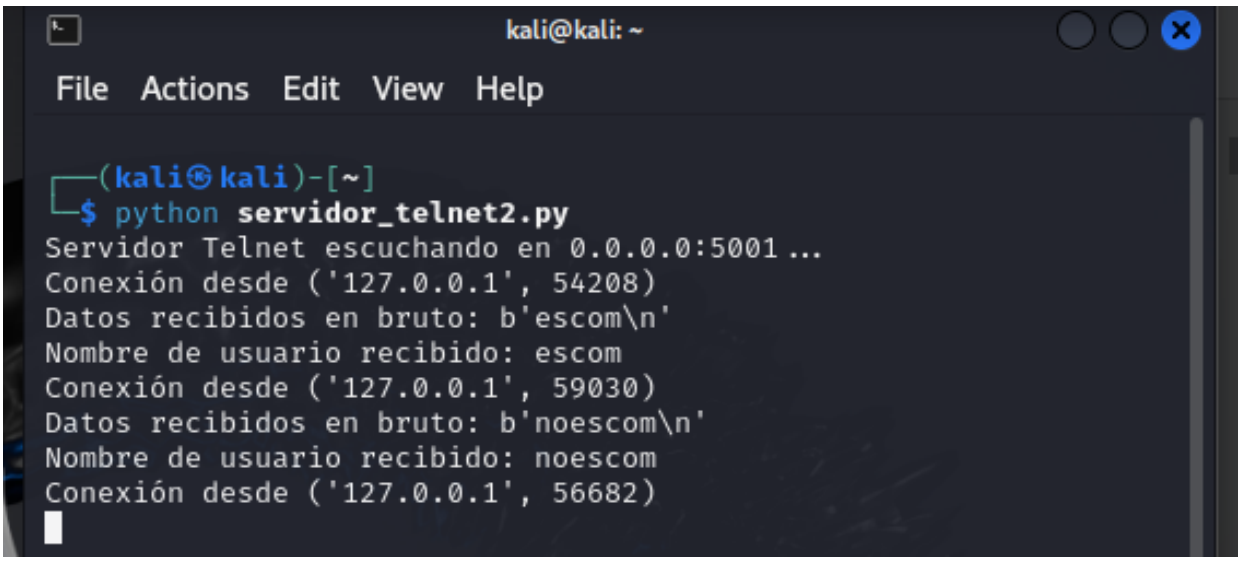
Para el caso de la autenticación fallida se puede ver el siguiente mensaje:



Para el servidor_telnet2.py se va a agregar la siguiente lógica para poder recibir las credenciales y auténticas:

```
import socket

# Configuración del servidor
HOST = '0.0.0.0'  # Escuchar en todas las interfaces disponibles
PORT = 5001       # Puerto arbitrario que no requiere privilegi>
```

```
# Crear el socket del servidor
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREA>
server_socket.bind((HOST, PORT))
server_socket.listen(1)
print(f"Servidor Telnet escuchando en {HOST}:{PORT}...")

while True:
    client_socket, client_address = server_socket.accept()
    print(f"Conexión desde {client_address}")

    # Enviar mensaje de bienvenida
    client_socket.sendall(b"Bienvenido al servidor Telnet\n")


    # Recibir nombre de usuario
    username_data = client_socket.recv(1024)
    print(f"Datos recibidos en bruto: {username_data}")
    username = username_data.decode('utf-8').strip()
    print(f"Nombre de usuario recibido: {username}")


    # Pedir contraseña
    client_socket.sendall(b"Password: ")
    password = client_socket.recv(1024).strip().decode('utf-8')

    # Autenticación simple
    if username == 'escom' and password == 'escom':
        client_socket.sendall(b"Autenticacion exitosa\n")
    else:
        client_socket.sendall(b"Autenticacion fallida\n")

    client_socket.close()
```



CONCLUSIONES}

La creación de un sistema de autenticación básico en Telnet con Python demuestra cómo se pueden construir aplicaciones de red sencillas y efectivas, resaltando la versatilidad de la biblioteca socket para gestionar la comunicación entre cliente y servidor y su utilidad en aplicaciones de administración remota.

BIBLIOGRAFÍA

Stevens, W. Richard. Programación en red con Unix: La API de Sockets. Addison-Wesley Professional, 2003.

Forouzan, Behrouz A. Comunicaciones de Datos y Redes. McGraw-Hill, 2012. Tanenbaum, Andrew S., y David J. Wetherall. Redes de Computadoras. Pearson, 2010.

Comer, Douglas E. Internetworking con TCP/IP Volumen Uno. Prentice Hall, 2006. Kurose, James F., y Keith W. Ross. Redes de Computadoras: Un Enfoque Descendente. Pearson, 2017.

Beazley, David, y Brian K. Jones. Python Cookbook: Recetas para Dominar Python 3. O'Reilly Media, 2013. Begg, A. Sockets TCP/IP en C: Guía Práctica para Programadores. Morgan Kaufmann, 2000