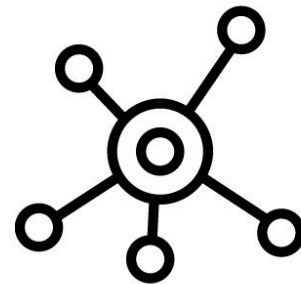
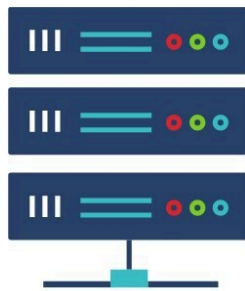


## *Tarea #991 Python scripts*

*Tarea #991*



# Python Sockets



**Abdiel Gabriel Hau Tun**

20-02-24

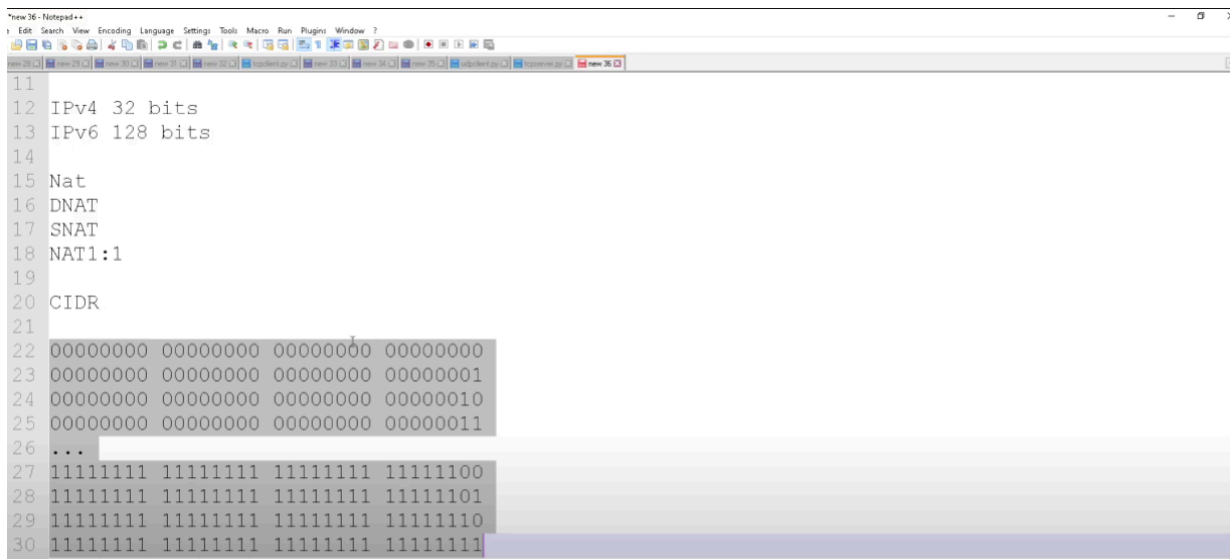
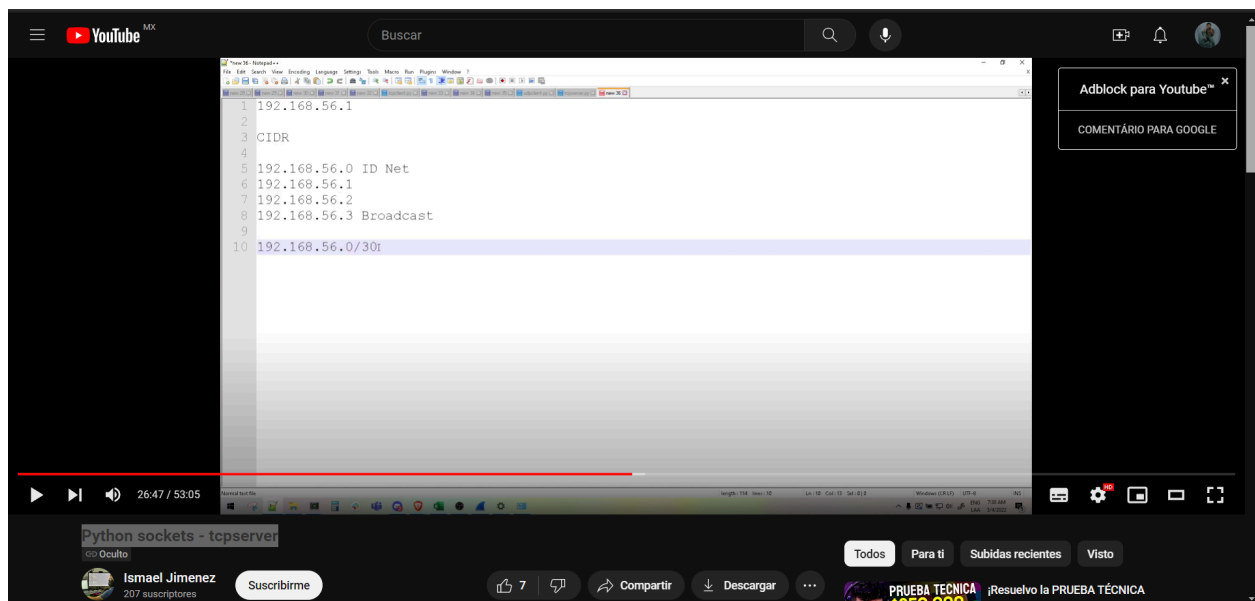
Seguridad de Datos

**Docente: Ismael Jimenez**

## Python sockets - tcpserver

Un servidor TCP en Python utilizando sockets es una aplicación que escucha y acepta conexiones entrantes de clientes a través del protocolo TCP (Transmission Control Protocol). El servidor establece un punto final en una dirección IP y un número de puerto específicos, esperando conexiones de clientes que deseen comunicarse con él.

Cuando un cliente se conecta al servidor, se establece una conexión TCP bidireccional entre el cliente y el servidor, lo que permite la transferencia de datos en ambas direcciones de manera confiable y ordenada.



El script `pcap_server.py` es un programa Python que crea un servidor TCP y escucha en todas las interfaces de la computadora en el puerto 9999. Utiliza subprocesos para multiprocesamiento y múltiples conexiones. El servidor recibe mensajes, imprime una confirmación y envía una respuesta de "ack". El script `pcap_client.py` se utiliza para probar el servidor y se puede modificar para conectarse a la IP y al puerto del servidor. El vídeo demuestra la funcionalidad del servidor y del cliente, así como el uso de Wireshark para el análisis del tráfico. En conclusión, el video muestra una implementación básica de un servidor y un cliente TCP, con énfasis en comprender el comportamiento del servidor y capturar el tráfico de la red.

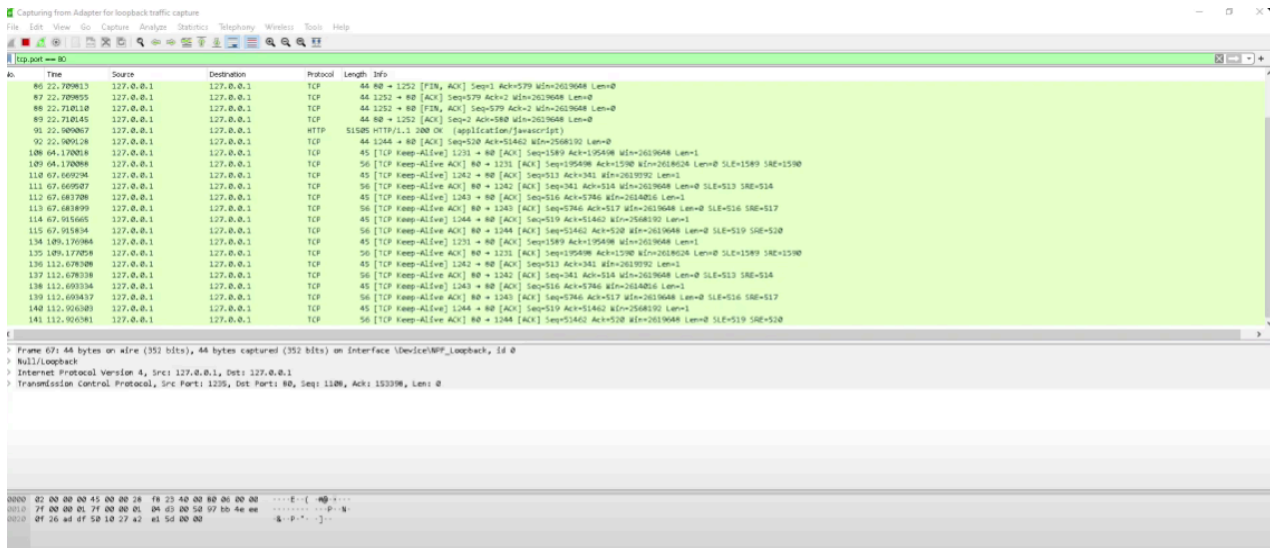
## Wireshark - Python TCP Proxy

Un proxy TCP en Python es una aplicación que actúa como intermediario entre un cliente y un servidor TCP, interceptando y reenviando el tráfico entre ellos. Wireshark, por otro lado, es una herramienta de análisis de protocolos de red que permite capturar y analizar el tráfico en una red.

Cuando se combina Wireshark con un proxy TCP en Python, se puede realizar un análisis detallado del tráfico TCP que fluye a través del proxy. El proxy TCP en Python podría estar diseñado para realizar diversas funciones, como el registro de solicitudes y respuestas, la modificación de los datos en tránsito, la aplicación de políticas de seguridad, entre otras

The screenshot shows a YouTube video player with the title "Wireshark - Python TCP Proxy" and the channel name "Ismael Jimenez". The video content displays a Python script for a TCP proxy server and a Wireshark packet capture. The script, `pcap_server.py`, is shown in a code editor window. It imports `sys`, `socket`, and `threading`. The `proxy_handler` function takes `client_socket`, `remote_host`, `remote_port`, and `receive_first` as arguments. It connects to the remote host, receives data, and sends it to the local client. The script also includes a `while True` loop for continuous operation. The Wireshark packet capture shows a TCP connection between the proxy and the remote host, with the proxy acting as the intermediate node.

```
1 import sys
2 import socket
3 import threading
4
5 def proxy_handler(client_socket, remote_host, remote_port, receive_first):
6
7     # connect to the remote host
8     remote_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9     print(remote_host + ":" + str(remote_port))
10    remote_socket.connect((remote_host, remote_port))
11
12    # receive data from the remote end if necessary
13    if receive_first:
14
15        remote_buffer = receive_from(remote_socket)
16        print("Hexdump\n" + str(remote_buffer))
17        hexdump(remote_buffer)
18        print("After Hexdump\n")
19
20        # send it to our response handler
21        remote_buffer = response_handler(remote_buffer)
22
23        # if we have data to send to our local client, send it
24        if len(str(remote_buffer)):
25            print ("[+] Sending " + str(len(str(remote_buffer))) + " bytes to localhost.")
26            client_socket.send(str(remote_buffer).encode())
27
28        # now lets loop and read from local,
29        # send to remote, send to local
30        # rinse, wash, repeat
31        while True:
32
33            # read from local host
34            local_buffer = receive_from(client_socket)
35            print("After receive from: " + str(local_buffer))
36
37            if len(local_buffer):
38
39                print ("[-] Received " + str(len(local_buffer)) + " bytes from localhost.")
```



En este video, se presenta un script de TCP proxy funcionando en Python 3, el cual es capaz de interceptar y modificar el tráfico que viaja a un puerto específico. Para ilustrar su funcionamiento, el narrador utiliza Wireshark para monitorear el tráfico en el loopback y filtra por el puerto 80.

Después de ejecutar el script, acceden a un sitio web a través del navegador en modo incógnito, utilizando la URL 127.0.0.1. El script intercepta la solicitud y la reenvía al sitio real, capturando toda la información en el tráfico entre el navegador y el sitio web.

El video explica que el script puede modificar el tráfico capturado, proporcionando dos funciones como ejemplo: `perform_request_handler` y `response_handler`. Estas funciones permiten alterar la solicitud del usuario y la respuesta del sitio web, respectivamente.

La actividad requerida para los espectadores es ejecutar el script TCP proxy, navegar a un sitio web específico a través de la URL 127.0.0.1, capturar pantallas de las ventanas relevantes, y subirlas a un repositorio como evidencia de la actividad completada.

El video menciona que aplicaciones como Squid proxy y Burp Suite tienen funcionalidades similares, pero en este caso, el script permite definir el comportamiento directamente en el código.