S6L3

Report Tecnico: Simulatore di UDP Flood in Python

1. Codice Sorgente

```
# -*- coding: utf-8 -*-

import socket
import random
import time
import ipaddress # Libreria per validare gli indirizzi IP

class UDPSimulator:

  def __init__(self, target_ip, target_port):

      try:
         ipaddress.ip_address(target_ip)
         self.target_ip = target_ip
      except ValueError:
         print(f"[!] Errore: '{target_ip}' non è un indirizzo IP valido.")
         raise

      if not (1 <= target_port <= 65535):
         raise ValueError("[!] Errore: La porta deve essere un numero tra 1 e 65535.")
      self.target_port = target_port

      print(f"[*] Simulatore inizializzato per target {self.target_ip}:{self.target_port}")

   def run_simulation(self, num_packets):

            print(f"[*] Avvio simulazione: verranno 'inviati' {num_packets} pacchetti.")


         payload = random.randbytes(1024)
      except AttributeError:

               payload = bytes(random.getrandbits(8) for _ in range(1024))

      sent_packets = 0
      for i in range(num_packets):

         print(f"SIMULAZIONE: Invio pacchetto {i + 1}/{num_packets}...", end='\r')



         sent_packets += 1


      print("\n" + "="*60)
      print(f"[*] Simulazione terminata.")
      print(f"[*] Pacchetti 'inviati' in totale: {sent_packets}/{num_packets}")
      print("="*60)

def get_user_input():
    target_ip = input("[?] Inserisci l'IP della macchina target: ")
```

```
    target_port = 0
    while not (1 <= target_port <= 65535):
        try:
            target_port = int(input("[?] Inserisci la porta UDP target (1-65535): "))
        except ValueError:
            print("[!] Errore: Inserisci un numero valido.")

    num_packets = 0
    while num_packets <= 0:
        try:
            num_packets = int(input("[?] Quanti pacchetti da 1 KB vuoi inviare? "))
        except ValueError:
            print("[!] Errore: Inserisci un numero intero valido.")

    return target_ip, target_port, num_packets

if __name__ == '__main__':
    print("--- Simulatore di Attacco UDP Flood (Versione Migliorata) ---")

    try:
          ip, port, packets = get_user_input()

          simulator = UDPSimulator(target_ip=ip, target_port=port)

         simulator.run_simulation(num_packets=packets)

    except (ValueError, KeyboardInterrupt) as e:

          print(f"\n[!] Operazione annullata. {e}")
```

2. Analisi Funzionale del Codice

Sezione 1: Librerie

- **`import socket`:** Importa la libreria per le comunicazioni di rete. Non è usata attivamente, ma è fondamentale per contestualizzare il codice.
- **`import random`:** Necessaria per generare i dati casuali che compongono i pacchetti simulati.
- **`import time`:** Utilizzata per eventuali pause o funzioni legate al tempo.
- **`import ipaddress`:** Una libreria potente e specifica per la validazione degli indirizzi IP (sia IPv4 che IPv6).

Sezione 2: La Classe `UDP`

- `class UDPSimulator:`: Definisce una nuova classe, che agisce come uno "stampo" per creare oggetti "simulatore".
- `def __init__(self, target_ip, target_port):`: È il **costruttore**. Viene eseguito ogni volta che si crea un nuovo oggetto `UDPSimulator`.
    - **`ipaddress.ip_address(target_ip)`**: Tenta di convertire la stringa dell'IP in un oggetto IP. Se la stringa non è un IP valido, solleva un errore `ValueError`.
    - **`self.target_ip = target_ip`:** Se l'IP è valido, lo salva all'interno dell'oggetto.
    - **`if not (1 <= target_port <= 65535):`**: Controlla che il numero di porta sia nell'intervallo consentito.
    - **`raise ValueError(...)`**: Se la porta non è valida, ferma il programma sollevando un errore.
- **`def run_simulation(self, num_packets):`**: È il metodo che esegue la simulazione vera e propria.
    - **`payload = random.randbytes(1024):`** Crea un pacchetto di 1024 byte (1 KB) di dati casuali. Viene creato una sola volta per efficienza.
    - **`for i in range(num_packets):`**: Inizia un ciclo che si ripeterà per il numero di pacchetti specificato dall'utente.
    - **`print(f"...", end='\r')`**: **Azione chiave della simulazione**. Stampa a schermo lo stato dell'invio. L'argomento `end='\r'` fa sì che la riga venga sovrascritta ad ogni ciclo, creando un contatore dinamico.
    - **`sent_packets += 1`**: Incrementa il contatore dei pacchetti "inviati".
    - **`print("\n" + "="*60):`** Alla fine del ciclo, stampa una riga di separazione per pulizia visiva.

Sezione 3: La Funzione `get_user_input`

- **`def get_user_input():`**: Definisce la funzione.
- **`target_ip = input(...)`**: Chiede all'utente di inserire l'IP e salva il testo nella variabile.
- **`while not (1 <= target_port <= 65535):`**: Inizia un ciclo che continua a chiedere la porta finché l'utente non inserisce un numero valido nell'intervallo corretto. Il blocco `try-except` gestisce il caso in cui l'utente inserisca testo invece di un numero.
- **`while num_packets <= 0:`**: Similmente, si assicura che il numero di pacchetti inserito sia un intero positivo.
- **`return target_ip, target_port, num_packets`:** Una volta che tutti i dati sono stati raccolti e validati, la funzione li "restituisce" al programma principale.

Sezione 4: Blocco di Esecuzione Principale

- **`if __name__ == '__main__':`**: Costrutto standard di Python che indica il punto di partenza del programma.
- **`try...except`**: Un blocco per la gestione degli errori. Permette di "catturare" in modo controllato eventuali problemi (come un IP non valido o l'interruzione da parte dell'utente con CTRL+C) e terminare il programma elegantemente.
- **`ip, port, packets = get_user_input():`** Chiama la funzione per ottenere i dati dall'utente.
- **`simulator = UDPSimulator(...):`** Crea un **oggetto** (una "istanza") della classe `UDPSimulator`, passando i dati dell'utente al costruttore `__init__`.
- **`simulator.run_simulation(...)`**: Chiama il metodo `run_simulation` sull'oggetto appena creato per avviare il ciclo di simulazione.
- **`except (ValueError, KeyboardInterrupt) as e:`**: Se durante l'esecuzione si verifica un `ValueError` (da `__init__`) o un `KeyboardInterrupt` (l'utente preme CTRL+C), il programma salta qui, stampa un messaggio di annullamento e si chiude senza crashare.