Pratica S6/L3 - Analisi di uno script Python per l'invio di pacchetti UDP

1. Introduzione

Lo scopo di questa esercitazione è analizzare in modo dettagliato il funzionamento di uno script Python che utilizza i socket per inviare pacchetti UDP verso un host remoto.
L'analisi viene condotta **riga per riga**, al fine di comprendere sia gli aspetti sintattici del linguaggio Python sia il comportamento dello stack di rete sottostante.

Lo script oggetto di studio implementa l'invio ripetuto di datagrammi UDP verso un indirizzo IP e una porta specificata dall'utente.

Il codice sorgente oggetto dell'analisi è riportato integralmente in **Appendice A**.

2. Analisi

Importazione dei moduli

```
1   import socket
2   import random
```

Il modulo ***socket*** è una libreria standard di Python che fornisce un'interfaccia a basso livello per la comunicazione di rete, permettendo di creare socket TCP e UDP e di inviare o ricevere dati.

Il modulo ***random*** fornisce funzioni per la generazione di valori pseudo-casuali. In questo script viene utilizzato per creare un payload di dimensione fissa composto da byte casuali.

Definizione della funzione principale

```
4   def udp_flood():
```

Questa istruzione definisce una funzione chiamata ***udp_flood***.
Il nome della funzione è significativo: il termine flood indica un invio massivo e ripetuto di pacchetti, tipico di alcune tecniche di stress o denial-of-service.

Acquisizione degli input dall'utente

Questa riga richiede all'utente di inserire un indirizzo IP (o un hostname) e lo memorizza nella variabile ***target_ip***.
Non viene effettuata alcuna validazione dell'input: un valore non valido potrebbe causare errori durante la fase di invio dei pacchetti.

L'utente inserisce una porta di destinazione, che viene convertita in un valore intero tramite la funzione ***int()***.
Se l'input non è numerico, il programma genera un'eccezione ***ValueError***. Inoltre, non viene verificato che la porta rientri nel range valido (0–65535).

Questa istruzione determina il numero di pacchetti UDP che verranno inviati.
Un valore molto elevato può comportare un consumo significativo di risorse locali e di rete.

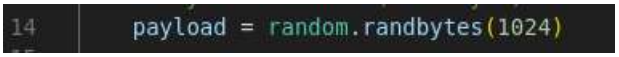
Creazione del socket UDP

Qui viene creato un socket di rete:

- ***AF_INET*** indica l'utilizzo del protocollo IPv4.
- ***SOCK_DGRAM*** specifica che il socket utilizza il protocollo UDP.

UDP è un protocollo **connectionless**, che non prevede handshake, controllo di flusso, né garanzia di consegna dei pacchetti.

Creazione del payload



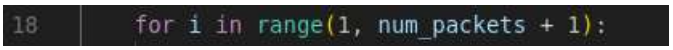
Questa riga genera un payload di 1024 byte (1 KB) contenente dati casuali.
Il payload è un oggetto di tipo bytes e viene creato una sola volta, quindi lo stesso contenuto viene inviato per tutti i pacchetti.

La funzione ***randbytes()*** è disponibile nelle versioni più recenti di Python (3.9 e successive).

Messaggio di avvio

Viene stampato un messaggio informativo che indica l'inizio dell'invio dei pacchetti, utilizzando una f-string per includere IP e porta di destinazione.

Ciclo di invio dei pacchetti



Questo ciclo viene eseguito tante volte quanto indicato dall'utente. L'indice ***i*** serve esclusivamente a fini di output.

Il metodo ***sendto()*** invia il payload UDP all'indirizzo IP e alla porta specificati.
Ogni iterazione genera un nuovo pacchetto UDP con lo stesso contenuto.

Viene stampato un messaggio per ogni pacchetto inviato.
Questa operazione rallenta significativamente l'esecuzione del programma, poiché l'I/O su console è molto più lento rispetto all'invio di pacchetti di rete.

Chiusura del socket

```
22      print("\nInvio completato.")
23      sock.close()
```

Al termine del ciclo, il programma stampa un messaggio finale e chiude il socket, liberando le risorse di sistema associate.
In contesti più robusti sarebbe preferibile l'uso di un costrutto try/finally o di un context manager per garantire la chiusura del socket anche in caso di errore.

Entry point del programma

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

Questo blocco assicura che la funzione udp_flood() venga eseguita solo quando il file viene avviato direttamente e non quando viene importato come modulo in un altro script.

3. Conclusione

Lo script analizzato dimostra l'utilizzo basilare dei socket UDP in Python e mette in evidenza il funzionamento di un invio ripetuto di datagrammi verso una destinazione specifica.
Dal punto di vista della cybersecurity, è un esempio utile per comprendere come traffico UDP massivo possa essere generato a livello applicativo e quali effetti possa avere sul sistema mittente e sulla rete.

Appendice A – Codice sorgente analizzato

```python
import socket
import random

def udp_flood():
    # Input utente
    target_ip = input("Inserisci IP target: ")
    target_port = int(input("Inserisci porta UDP target: "))
    num_packets = int(input("Numero di pacchetti da inviare: "))

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

    # Payload da 1 KB (1024 byte)
    payload = random.randbytes(1024)

    print(f"\nInizio UDP flood verso {target_ip}:{target_port}\n")

    for i in range(1, num_packets + 1):
        sock.sendto(payload, (target_ip, target_port))
        print(f"Pacchetto {i} inviato")

    print("\nInvio completato.")
    sock.close()

if __name__ == "__main__":
    udp_flood()

```