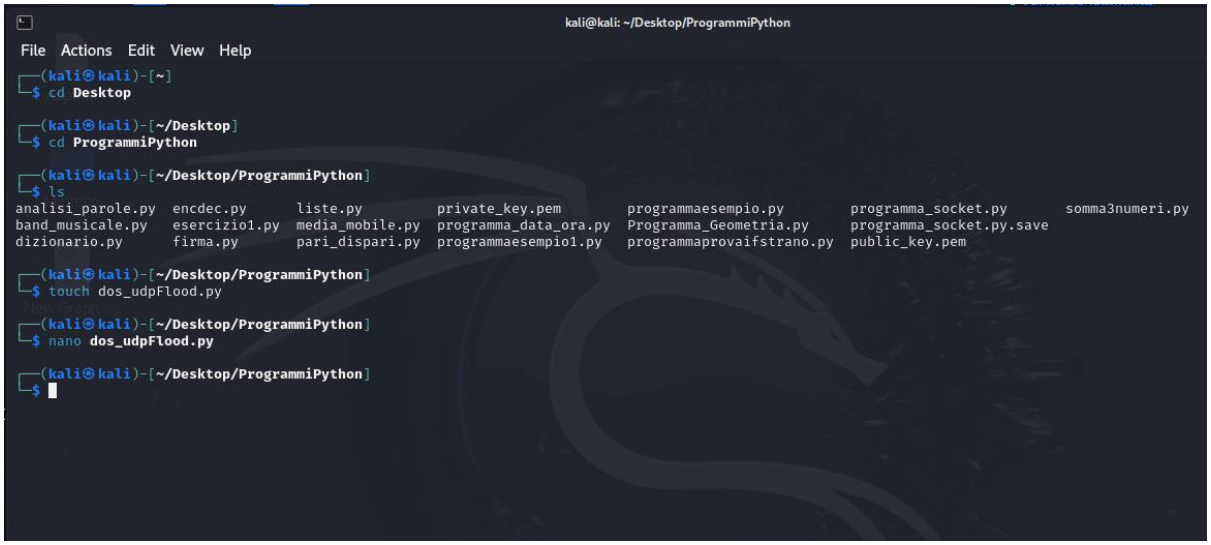
Report Esercizio 15/01/2024

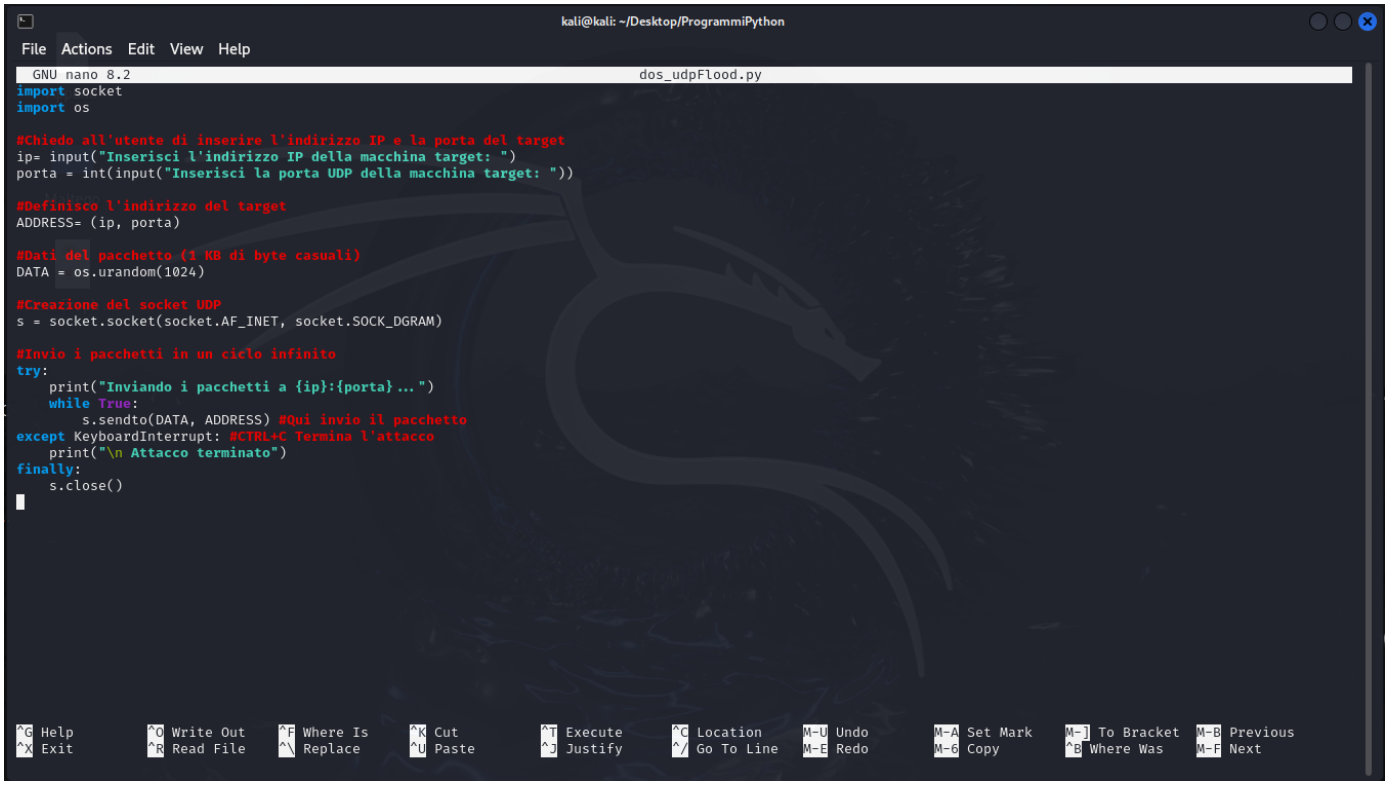
DoS UDP Flood Linguaggio Python Leonardo Catalano

L’esercizio richiede di scrivere un programma che simuli un UDP flood, ossia un invio massimo di richieste UDP verso una macchina target, che è in ascolto su una porta UDP casuale.
Requisiti del programma:
1. Input dell'IP Target:
    -Il programma deve richiedere all'utente di inserire l'IP della macchina target.
2. Input della Porta Target:
    -Il programma deve richiedere all'utente di inserire la porta UDP della macchina target.
3. Costruzione del Pacchetto:
    -La grandezza dei pacchetti da inviare deve essere di 1 KB per pacchetto.
    -Suggerimento: per costruire il pacchetto da 1 KB, potete utilizzare il modulo random per la generazione di byte casuali.
4. Numero di Pacchetti da Inviare:
    -Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare.

Comandi cmd per creare il file.py:

Aperta la shell, con il comando cd ci andiamo a spostare nella directory interessata, in questo caso visto che avevo creato una cartella in precedenza sul Desktop con nome ProgrammiPython il comando sarà : cd Desktop/ProgrammiPython (la / in mezzo serve a concatenare gli spostamenti).
Con il comando ls andremo a vedere il contenuto della cartella.
Con il comando touch nomefile andremo a creare il file ricordandoci di mettere l’estensione.py alla fine del nome.
Con il comando nano nomefile andremo ad aprire un semplice editor di testo dove andremo a creare/modificare il nostro programma.





Descrizione del programma dos_udpFlood.py

Per prima cosa importo due librerie (moduli):

-Socket --> che permette la creazione di socket (coppia indirizzo ip+ porta), per permettere la comunicazione in rete.

-Os --> esso fornisce un'interfaccia tra Python e il sistema operativo sottostante (in questo caso Kali) consentendo di :

-Accedere alle funzionalità del sistema operativo (es. manipolare file, directory, processi).

-Ottenere informazioni sul sistema (come variabili d'ambiente, percorso del file system).

-Eseguire operazioni specifiche del sistema operativo (eseguire comandi di shell,

gestire i processi).
E' una libreria molto importante che ci consentirà di effettuare la creazione randomica di n byte casuali.

Successivamente chiediamo all'utente di inserire i 2 input, il 1* l'indirizzo ip della macchina target, e il 2* la porta UDP della macchina target.
Gli input in python si fanno con questa sequenza --> nome_variabile = input ("Stringa che compare prima dell'inserimento che si va a specificare all'utente cosa deve inserire : ")
Nel nostro caso--> ip= input("Inserisci l'indirizzo IP della macchina target: ")
porta = int(input("Inserisci la porta UDP della macchina target: "))

Poi andiamo ad definire la variabile che conterrà l'indirizzo ip e la porta e la chiamiamo ADDRESS.
Andiamo a creare la variabile che conterrà i dati del pacchetto e la chiamiamo DATA, per crearla utilizziamo os.urandom(n) dove andremo a :
- Genera una sequenza di n byte casuali.
- Ogni byte può avere un valore compreso tra 0 e 255.
- Usa una fonte di numeri casuali criptograficamente sicuri (ottimo per simulare pacchetti realistici).

Andiamo a creare il socket UDP:
"s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)"

socket.AF_INET: specifica che stiamo lavorando con ipv4

socket.SOCK_DGRAM: specifica che stiamo usando il protocollo UDP (non TCP).

Infine andiamo ad inviare i pacchetti in un ciclo infinito:
"try:

print("Inviando i pacchetti a {ip}:{porta}...")

while True:

s.sendto(DATA, ADDRESS)

except KeyboardInterrupt:

print("\n Attacco terminato")

finally:

s.close() "

-Creo un Try except per permettere al ciclo infinito di essere interrotto dall'utente.
- Utilizziamo un ciclo infinito "While True" per inviare pacchetti continuamente fino allo

stop dell’utente.
-s.sendto(DATA, ADDRESS): invia il pacchetto al target specificato in ADDRESS (dove c’è coppia indirizzo ip + porta).
-except KeyboardInterrupt: Interrompo il programma e quindi l’invio di pacchetti con la combinazione Ctrl+ C.
- finally:
        s.close()
quando il programma viene interrotto anche il socket viene chiuso con il metodo s.close().

In sintesi il programma effettua le seguenti operazioni:
1. Chiede all'utente i dettagli del target (IP e porta UDP).
2. Crea un pacchetto UDP da 1 KB di dati.
3. Crea un socket UDP per inviare i pacchetti.
4. Invia pacchetti al target in un ciclo infinito.
5. Permette di interrompere l'invio premendo Ctrl + C.
6. Chiude il socket quando il programma termina.

Per salvare sotto nano si utilizza la combinazione di comandi Ctlr + o + invio

Per tornare alla shell si utilizza la combinazione Ctlr + x