

## Consegna W7D4

Il programma ha lo scopo di simulare un attacco, inviando un numero definito di pacchetti da 1KB verso un indirizzo IP e una porta specifici. Questa esercitazione è utile per comprendere il funzionamento dei protocolli di rete e le vulnerabilità legate ai servizi UDP.

## Svolgimento

Come primo passaggio andiamo a importare i vari moduli che useremo nel corso dello svolgimento dell'esercitazione.

Nel nostro caso andiamo ad importare:

- Sys: utile per la gestione degli argomenti da linea di comando.
- Socket: per la creazione e la gestione delle comunicazioni UDP.
- Random: per generare dati casuali da inserire nei pacchetti che inviamo.

```
esercizio.py > ...  
1  import sys  
2  import socket  
3  import random
```

Andiamo a creare la funzione che ci permette di creare un pacchetto della grandezza di 1KB (1024 byte) contenente dati casuali, utilizzando dopo "random.randbytes()" per creare una sequenza di byte casuali.

```
5  def create_packet(size: int=1024) -> bytes:  
6      return random.randbytes(size)  
7
```

Successivamente andiamo a creare la parte del codice secondo il quale andiamo a verificare che l'utente abbia inserito correttamente i parametri richiesti, mentre nel caso contrario restituisce un messaggio di utilizzo corretto terminando il programma.

```
8  if len(sys.argv) != 4:  
9      print("Usage: python esercizio.py <ip> <porta> <npack>")  
10     sys.exit(1)  
11
```

Andiamo a creare poi la parte di codice che va a leggere e a convertire nei tipi appropriati l'indirizzo IP e la porta ai quali legarsi e il numero di pacchetti da inviare.

```
12  ip    = sys.argv[1]  
13  porta = sys.argv[2]  
14  npack = int(sys.argv[3])  
15
```

In questo momento andiamo a creare il socket UDP e lo va a collegare all'indirizzo IP e la porta specificata.

```
17  s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  
18  s.connect((ip, int(porta)))
```

Andiamo poi a creare la parte del codice che va a rendere ciclico per il numero specificato precedentemente, per ogni interazione genera un pacchetto da 1KB e lo invia tramite socket infine quella parte che va a generare la stampa di un messaggio di conferma per ogni pacchetto inviato.

```
20 for i in range(npack):
21     packet = random.randbytes(1024) #1024 byte = 1 KB packet
22     s.send(packet)
23     print(f"Pacchetto inviato {i+1}/{npack} a {ip}:{porta}")
24
```


Aggiungiamo infine la chiusura del socket con la “stampa” del messaggio finale.

```
25 s.close()
26 print("Invio di tutti i pacchetti completato.")
27
```

Con le prossime due figure andiamo a mostrare come l’invio di pacchetti sia andato a buon fine e la “lettura” da parte di wireshark di uno dei pacchetti.

```
(kali@kali)-[~/progetti/Consegne/w7d4]
$ python esercizio.py 8.8.8.8 53 10
Pacchetto inviato 1/10 a 8.8.8.8:53
Pacchetto inviato 2/10 a 8.8.8.8:53
Pacchetto inviato 3/10 a 8.8.8.8:53
Pacchetto inviato 4/10 a 8.8.8.8:53
Pacchetto inviato 5/10 a 8.8.8.8:53
Pacchetto inviato 6/10 a 8.8.8.8:53
Pacchetto inviato 7/10 a 8.8.8.8:53
Pacchetto inviato 8/10 a 8.8.8.8:53
Pacchetto inviato 9/10 a 8.8.8.8:53
Pacchetto inviato 10/10 a 8.8.8.8:53
Invio di tutti i pacchetti completato.
```

No.	Time	Source	Destination	Protocol	Length	Info
4	2.194214912	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (7) 0x4
5	2.194519934	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (10) 0x
6	2.194835919	10.0.2.15	8.8.8.8	DNS	1068	Server status request res
7	2.195078576	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (9) res
8	2.195274373	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (15) re
9	2.195649008	10.0.2.15	8.8.8.8	DNS	1068	DNS Stateful operations (
10	2.195912103	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (15) 0x
11	2.196049037	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (3) res
12	2.196168450	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (8) res
13	2.196296360	10.0.2.15	8.8.8.8	DNS	1068	Unknown operation (14) 0x

Wireshark - Follow UDP Stream (udp.stream eq 0) - any

Find: Case sensitive Find Next

Filter Out This Stream Print Save as... Back Close

## Facoltativa

Lo scopo della parte facoltativa è quello di simulare un comportamento più realistico durante l'invio dei pacchetti UDP, introducendo un ritardo casuale tra 0 e 0.1 secondi tra ogni invio. Questo accorgimento rende la simulazione meno prevedibile e più vicina al traffico generato da utenti reali.

Andiamo ad importare innanzi tutto il modulo `time` utile per gestire le pause e `random` per generare il ritardo, simulando ad esempio più utenti.

Aggiungiamo poi la funzione "`random.uniform()`" per avere in questo modo un valore casuale che va da 0 a 0.1 come ritardo.

```
esercizio.py x
esercizio.py > ...
1  import sys
2  import socket
3  import random
4  import time                                     # AGGIUNTA PER PARTE FACOLTATIVA
5
6  def create_packet(size: int=1024) -> bytes:
7      return random.randbytes(size)
8
9  if len(sys.argv) != 4:
10     print("Usage: python esercizio.py <ip> <porta> <npack>")
11     sys.exit(1)
12
13  ip = sys.argv[1]
14  porta = sys.argv[2]
15  npack = int(sys.argv[3])
16
17
18  s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
19  s.connect((ip, int(porta)))
20
21  for i in range(npack):
22      packet = random.randbytes(1024) #1024 byte = 1 KB packet
23      s.send(packet)
24      print(f"Pacchetto inviato {i+1}/{npack} a {ip}:{porta}")
25      ritardo = random.uniform(0, 0.1)                                     # AGGIUNTA PER PARTE FACOLTATIVA
26      time.sleep(ritardo)                                                   # AGGIUNTA PER PARTE FACOLTATIVA
27      print(f"Pacchetto inviato {i+1}/{npack} a {ip}:{porta} (ritardo: {ritardo:.3f}s)" # AGGIUNTA PER PARTE FACOLTATIVA
28
29
30
31
32  s.close()
33  print("Invio di tutti i pacchetti completato.")
```

```
(kali㉿kali)-[~/progetti/Consegne/w7d4]
$ python3 esercizio.py 8.8.8.8 53 10
Pacchetto inviato 1/10 a 8.8.8.8:53
Pacchetto inviato 1/10 a 8.8.8.8:53 (ritardo: 0.009s)
Pacchetto inviato 2/10 a 8.8.8.8:53
Pacchetto inviato 2/10 a 8.8.8.8:53 (ritardo: 0.059s)
Pacchetto inviato 3/10 a 8.8.8.8:53
Pacchetto inviato 3/10 a 8.8.8.8:53 (ritardo: 0.020s)
Pacchetto inviato 4/10 a 8.8.8.8:53
Pacchetto inviato 4/10 a 8.8.8.8:53 (ritardo: 0.085s)
Pacchetto inviato 5/10 a 8.8.8.8:53
Pacchetto inviato 5/10 a 8.8.8.8:53 (ritardo: 0.050s)
Pacchetto inviato 6/10 a 8.8.8.8:53
Pacchetto inviato 6/10 a 8.8.8.8:53 (ritardo: 0.020s)
Pacchetto inviato 7/10 a 8.8.8.8:53
Pacchetto inviato 7/10 a 8.8.8.8:53 (ritardo: 0.022s)
Pacchetto inviato 8/10 a 8.8.8.8:53
Pacchetto inviato 8/10 a 8.8.8.8:53 (ritardo: 0.059s)
Pacchetto inviato 9/10 a 8.8.8.8:53
Pacchetto inviato 9/10 a 8.8.8.8:53 (ritardo: 0.100s)
Pacchetto inviato 10/10 a 8.8.8.8:53
Pacchetto inviato 10/10 a 8.8.8.8:53 (ritardo: 0.088s)
Invio di tutti i pacchetti completato.
```