REPORT TECNICO PORT SNIFFER

Report sniffer 3
Obiettivo 3
Utilizzo libreria socket 3
Utilizzo libreria scapy 4

Report sniffer

Obiettivo

Sviluppare un software, semplice da utilizzare, in grado di catturare il socket di rete. Questo obiettivo è stato raggiunto in due modi differenti.

Utilizzo libreria socket

Nel primo caso (**sniff.py**) è stato sviluppato un software in grado di mettere il computer in ascolto su una porta fornita in input dall'utente e di catturare il socket del client collegato a quella porta.

```
1    import socket
```

Per raggiungere questo obiettivo abbiamo importato la liberia **socket** che permette di creare e gestire connessioni di rete tramite i socket.

```
3    SRV_ADDRESS="0.0.0.0"
4    SRV_PORT=int(input("Inserisci la porta su cui vuoi ascoltare: "))
```

Tramite le variabili **SRV_ADDRESS** e **SRV_PORT** scegliamo l'indirizzo IP e la porta su cui ascoltare. Selezionando come indirizzo IP l'indirizzo "0.0.0.0", ci metteremo in ascolto su qualsiasi interfaccia. La porta invece viene inserita dall'utente.

```
6    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
7    s.bind((SRV_ADDRESS,SRV_PORT))
```

L'istruzione **socket.socket** ci permette di creare un socket con le seguenti caratteristiche:

- **AF_INET**: specifichiamo l'utilizzo di un indirizzo IPv4
- **SOCK_STREAM**: selezioniamo il protocollo TCP in questo caso

Successivamente "leghiamo" questo socket alle informazioni inserite precedentemente tramite la funzione **.bind**

```
8    s.listen(1)
```

La funzione **.listen** permette di metterci in ascolto sul socket selezionato. Il valore 1 indica il numero di elementi in coda.

```
10   connection, address= s.accept()
```

Con il comando **.accept()** accettiamo la connessione in entrata. Esso restituirà due valori:

- Un nuovo socket che può essere utilizzato per comunicare con il client, assegnato alla variabile **connection**
- Un indirizzo (tipicamente una tupla che contiene l'indirizzo IP e il numero di porta) che identifica il client, assegnato alla variabile **address**.

```
12      connection.close()
```

Una volta stampati i dati richiesti (**address**), chiudiamo la connessione con il comando soprastante.

```
└─$ python sniff.py
Inserisci la porta su cui vuoi ascoltare: 8080
Server inizializzato! In ascolto sulla porta  8080
In attesa di connesione...
Il client si è connesso con il seguente indirizzo:  ('10.0.2.15', 59724)
```

Utilizzo libreria scapy

Nel secondo caso (**sniff2.py**), è stato sviluppato un software in grado di catturare tutto il traffico in ingresso e in uscita e di mostrare i pacchetti con le relative informazioni.

```
1      from scapy.all import *
```

Tramite il comando **import** abbiamo importato una libreria specifica per il raggiungimento del nostro obiettivo. La libreria **scapy** è infatti una potente libreria Python utilizzata per l'analisi dei pacchetti di rete.
Con l'istruzione soprastante importiamo tutte le funzioni di tale libreria.

```
3      def lista (packet):
4          print (packet.summary())
```

Creiamo una funzione in grado di mostrare i pacchetti catturati. In questo caso, grazie alla funzione **.summary()** indichiamo al programma di mostrarci le informazioni principali dei pacchetti senza scendere troppo nei dettagli, cosa che farebbe ad esempio il comando **.show()**. Lo scopo è di rendere il tutto più leggibile.

```
6      while True:
7          print("Inizio della cattura dei pacchetti...")
8          sniff(prn=lista, store=0, timeout=10)
9          break
10     print ("Rilevamento terminato")
```

Quello soprastante è il programma vero e proprio.

- La funzione **while TRUE** ci permette di entrare immediatamente nel ciclo.
- La funzione **sniff** rileverà i pacchetti che verranno stampati grazie alla funzione scritta precedentemente e qui richiamata.
- L'indicatore **store=0** indica al programma di non memorizzare ciò che viene catturato.
- La funzione **timeout=10** ci permette di uscire dal ciclo e terminare il programma dopo 10 secondi di rilevamento (tempo standard scelto).

```
└─$ sudo python sniff2.py
[sudo] password for kali:
Inizio della cattura dei pacchetti...
Ether / IP / TCP 10.0.2.15:51460 > 34.107.221.82:http A
Ether / IP / TCP 34.107.221.82:http > 10.0.2.15:51460 A / Padding
Ether / IP / TCP 10.0.2.15:51388 > 80.67.82.80:http A
Ether / IP / TCP 10.0.2.15:33546 > 80.67.82.80:http A
Ether / IP / TCP 80.67.82.80:http > 10.0.2.15:51388 A / Padding
Ether / IP / TCP 80.67.82.80:http > 10.0.2.15:33546 A / Padding
Ether / IP / TCP 10.0.2.15:47610 > 2.23.155.242:http A
Ether / IP / TCP 2.23.155.242:http > 10.0.2.15:47610 A / Padding
Ether / IP / TCP 10.0.2.15:54258 > 142.250.180.131:http A
Ether / IP / TCP 10.0.2.15:54268 > 142.250.180.131:http A
Ether / IP / TCP 142.250.180.131:http > 10.0.2.15:54258 A / Padding
Ether / IP / TCP 142.250.180.131:http > 10.0.2.15:54268 A / Padding
Ether / IP / TCP 10.0.2.15:54276 > 142.250.180.131:http A
Ether / IP / TCP 142.250.180.131:http > 10.0.2.15:54276 A / Padding
Ether / IP / TCP 10.0.2.15:51626 > 2.23.155.235:http A
Ether / IP / TCP 2.23.155.235:http > 10.0.2.15:51626 A / Padding
Rilevamento terminato
```