

Redes de Computadoras

Trabajo Práctico N° 4

Capa de Transporte - Sockets TCP y UDP - Escaneo y reenvío de Puertos.

Fabris, Adriano Santino
Padilla Lumelli, Gonzalo Agustín
Robledo, Mariano José

Introducción

En este trabajo práctico se realizó la comunicación mediante sockets TCP y UDP, se analizó el intercambio de segmentos en el establecimiento y finalización de conexiones TCP, y en el intercambio de datos mediante TCP y UDP y se comprendió el uso y utilidad de herramientas de escaneo de puertos con programas como nmap y nikto.

Ejecución del program

Para la ejecutar los archivos con terminación en .py, ya sea en windows o linux, se debe de situar en la carpeta que contiene a los archivos y escribir el siguiente comando "python3 'nombre_del_archivo'.py", donde 'nombre_del_archivo' se reemplaza por el nombre del archivo sin comillas.

Actividad 1: Sockets UDP

En esta actividad se programa un chat de mensajería por protocolo UDP, al ejecutar el programa se nos pedirá el nombre con el cual vamos a interactuar con los demás participantes. Luego, ingresamos al chat donde podremos chatear con todos los usuarios que estén escuchando. Al enviar un mensaje se le enviará a todas las ip de la LAN inclusive a nosotros mismo.

Al ingresar por primera vez, se enviará un mensaje automáticamente anunciando el nombre del usuario que se ha conectado.

Los mensajes se mostrarán con el nombre del usuario que lo escribió y su ip, junto con el mensaje dado.

Al escribir el mensaje "exit" se desconectara de la comunicación y se mostrará a los usuarios que estén escuchando que dicho usuario se ha desconectado

El puerto que se ocupa para la comunicación es el 60.000.

El programa consta de un hilo para ejecutar la lectura de los mensajes mientras que en la ejecución normal del programa se encuentra la función de enviar mensajes.

La ejecución del mismo se vería así:

```
mariano@mariano-Ubuntu:~/git/Redes/tp4$ python3 UDP.py
Dar tu nombre: Mariano
->El usuario Mariano se ha unido a la conversación
El usuario Adriano se ha unido a la conversación
hola
->Mariano(192.168.1.2)dice: hola
Adriano(192.168.1.7)dice: Hola
chau
->Mariano(192.168.1.2)dice: chau
exit
Saliendo de la conversación
El usuario Mariano (192.168.1.2) ha abandonado la conversación
mariano@mariano-Ubuntu:~/git/Redes/tp4$
```

Actividad 2: Sockets TCP

En esta actividad se programa un chat de mensajería por protocolo TCP, lo primero que nos pide es nuestro nombre de usuario. Luego el programa está dividido en dos partes, puede funcionar como un servidor a la espera de peticiones o como cliente que intenta conectarse a un servidor.

En el primer caso actúa como servidor, se bloquea el proceso esperando una petición de conexión, cuando ésta llega se establece la conexión y se activa el chat de mensajería. Cabe mencionar que el proceso que actúa como servidor no puede abandonar la conversación, solo el que actúa como cliente puede terminar la conexión de mensajería. Una vez que el cliente finalice la conexión el proceso de servidor se puede elegir si esperar una nueva conexión o finalizar el proceso.

El segundo caso actúa como cliente, al escribir la ip de servidor, este enviará un mensaje de sincronización y se establecerá la comunicación entre ambos. Al finalizar la comunicación con el comando "exit", se le pedirá si quiere tratar de establecer conexión con otro servidor o finalizar el proceso de cliente.

En ambos casos el Puerto que se utiliza es el 60.000.

Como **servidor** se vería de la siguiente forma:

```
mariano@mariano-Ubuntu:~/git/Redes/tp4$ python3 TCP.py
Dar tu nombre: Mariano
1) enviar mensaje 2) esperar mensaje
2
Esperando conexion
-->Inicio de converscion
Hola
-->Adriano dice: Como andas
exit
No se puede salir porque estas en una conexion
-->Adriano dice: chau
Ha abandonado la conversación
-->
Quieres Continuar? 1=si, 2=no 2
-->Has finalizado el servidor.
mariano@mariano-Ubuntu:~/git/Redes/tp4$
```

Como **cliente** se vería de la siguiente forma:

```
mariano@mariano-Ubuntu:~/git/Redes/tp4$ python3 TCP.py
Dar tu nombre: Mariano
1) enviar mensaje 2) esperar mensaje
1
Dar una IP: 192.168.1.7
Se ha conectado
Inicio de converscion
-->Hola
-->Adriano dice:Hola como andamos
exit
Quiere volver a intenerar o establecer otra conexion1- Si 2- No
2
mariano@mariano-Ubuntu:~/git/Redes/tp4$
```

Actividad 3: Análisis de tráfico

Al realizar una comunicación por TCP o UDP y analizar el tráfico con wireshark, al filtrar por el puerto específico, podemos observar la información de los paquetes, viendo así los mensajes que se envían.

Una comunicación UDP:

udp.port==60000							
No.	Time	Source	Destination	Protocol	Length	Info	
7	3.785284004	192.168.1.2	255.255.255.255	UDP	57	60000 → 60000	Len=13
12	9.143994904	192.168.1.7	255.255.255.255	UDP	62	60000 → 60000	Len=13
13	10.665446042	192.168.1.7	255.255.255.255	UDP	62	60000 → 60000	Len=12
29	13.689704070	192.168.1.2	255.255.255.255	UDP	56	60000 → 60000	Len=12
31	15.769744729	192.168.1.2	255.255.255.255	UDP	59	60000 → 60000	Len=15
34	18.370273812	192.168.1.2	255.255.255.255	UDP	61	60000 → 60000	Len=17
35	20.753177985	192.168.1.7	255.255.255.255	UDP	62	60000 → 60000	Len=13
87	24.794507064	192.168.1.2	255.255.255.255	UDP	56	60000 → 60000	Len=12
101	27.717886489	192.168.1.7	255.255.255.255	UDP	62	60000 → 60000	Len=12

00 04 00 01 00 06 40 8d 5c ff ea d9 00 00 08 00@. \.....
45 00 00 28 24 a4 40 00 40 11 54 77 c0 a8 01 02	E..(\$.@. @.Tw....
ff ff ff ff ea 60 ea 60 00 14 04 16 4d 61 72 69`.. Mari
61 6e 6f 3a 68 6f 6c 61	ano:hola

Una comunicación TCP:

tcp.port==60000							
No.	Time	Source	Destination	Protocol	Length	Info	
113	14.565882017	192.168.1.7	192.168.1.2	TCP	76	40588 → 60000	[SYN, Seq=0 Win=64240
114	14.565898869	192.168.1.2	192.168.1.7	TCP	76	60000 → 40588	[SYN, ACK] Seq=0 Ack=3
115	14.566574535	192.168.1.7	192.168.1.2	TCP	68	40588 → 60000	[ACK] Seq=1 Ack=1 Win=
123	18.350497067	192.168.1.7	192.168.1.2	TCP	88	40588 → 60000	[PSH, ACK] Seq=1 Ack=1
124	18.350519226	192.168.1.2	192.168.1.7	TCP	68	60000 → 40588	[ACK] Seq=1 Ack=21 Win
137	23.096102233	192.168.1.7	192.168.1.2	TCP	90	40588 → 60000	[PSH, ACK] Seq=21 Ack=
138	23.096123314	192.168.1.2	192.168.1.7	TCP	68	60000 → 40588	[ACK] Seq=1 Ack=43 Win
148	28.803504345	192.168.1.2	192.168.1.7	TCP	85	60000 → 40588	[PSH, ACK] Seq=1 Ack=4
149	28.810050297	192.168.1.7	192.168.1.2	TCP	68	40588 → 60000	[ACK] Seq=43 Ack=18 Wi
156	31.668104625	192.168.1.2	192.168.1.7	TCP	85	60000 → 40588	[PSH, ACK] Seq=18 Ack=
157	31.677044464	192.168.1.7	192.168.1.2	TCP	68	40588 → 60000	[ACK] Seq=43 Ack=35 Wi
160	33.426955092	192.168.1.2	192.168.1.7	TCP	82	60000 → 40588	[PSH, ACK] Seq=35 Ack=
161	33.428918673	192.168.1.7	192.168.1.2	TCP	68	40588 → 60000	[ACK] Seq=43 Ack=49 Wi
193	36.299242590	192.168.1.7	192.168.1.2	TCP	80	40588 → 60000	[PSH, ACK] Seq=43 Ack=
194	36.299263701	192.168.1.2	192.168.1.7	TCP	68	60000 → 40588	[ACK] Seq=49 Ack=55 Wi
195	37.851542930	192.168.1.7	192.168.1.2	TCP	80	40588 → 60000	[PSH, ACK] Seq=55 Ack=
196	37.851563604	192.168.1.2	192.168.1.7	TCP	68	60000 → 40588	[ACK] Seq=49 Ack=67 Wi
197	37.851687178	192.168.1.2	192.168.1.7	TCP	68	60000 → 40588	[FIN, ACK] Seq=49 Ack=
198	37.852431212	192.168.1.7	192.168.1.2	TCP	68	40588 → 60000	[FIN, ACK] Seq=67 Ack=
199	37.852439486	192.168.1.2	192.168.1.7	TCP	68	60000 → 40588	[ACK] Seq=50 Ack=68 Wi

00 00 00 01 00 06 20 1e 88 70 7b 50 00 00 08 00p{P....
45 00 00 48 90 1e 40 00 40 06 27 38 c0 a8 01 07	E..H..@. @.'8....
c0 a8 01 02 9e 8c ea 60 e5 dd a9 ca ae e9 8c 7e~
80 18 01 f6 2e 0b 00 00 01 01 08 0a bb 4d 2f faM/.
fb 9d d5 8a 41 64 72 69 61 6e 6f 3a 68 6f 6c 61Adri ano:hola
20 4d 61 72 69 61 6e 6f	Mariano

Observamos como en los primeros tres mensajes se establece la conexión y como se cierra la conexión en las últimas tres líneas.

Actividad 4: Escaneo de puertos

4.2:

Al realizar esta actividad con mis dos computadoras una de escritorio y otra notebook ambas con ubuntu, al buscar algún puerto abierto de la lista (21, 22, 23, 25, 135 al 139, 443, 445, 3389) ninguno se encontró abierto.

```
PORT    STATE  SERVICE
21/tcp  closed ftp
22/tcp  closed ssh
23/tcp  closed telnet
24/tcp  closed priv-mail
25/tcp  closed smtp
135/tcp closed msrpc
136/tcp closed profile
137/tcp closed netbios-ns
138/tcp closed netbios-dgm
139/tcp closed netbios-ssn
443/tcp closed https
444/tcp closed snpp
445/tcp closed microsoft-ds
3389/tcp closed ms-wbt-server
```

A continuación se lista una serie de riesgo al tener dichos puertos abiertos, sacado de chatGPT

Puerto	Servicio	Riesgo Principal
21	FTP	Transfiere archivos sin cifrado. Riesgo de robo de credenciales y archivos.
22	SSH	Acceso remoto seguro, pero objetivo común de fuerza bruta si no está protegido.
23	Telnet	Similar a SSH, pero sin cifrado . Muy inseguro; expone contraseñas en texto plano.
25	SMTP	Para enviar correos. Puede usarse para spam o relay abierto si no se configura bien.
135-139	RPC / NetBIOS (Windows)	Facilita ataques como enumeración de red, infección por malware tipo Wanna Cry .
443	HTTPS	Seguro si bien configurado. Riesgos si tiene SSL débil o certificados inválidos.

445	SMB (Windows)	Muy vulnerable a ataques como Eternal Blue . Facilita propagación de ransomware.
3389	RDP	Acceso remoto a escritorios Windows. Muy usado en ataques de fuerza bruta o ransomware.

4.3:

Se realizó una búsqueda a la ip de la UNCuyo (179.0.132.0/22) y se encontraron varias ip con el puerto 80 abierto.

```
Nmap scan report for mail.derecho.uncu.edu.ar (179.0.133.194)
Host is up (0.069s latency).
```

```
PORT      STATE SERVICE
80/tcp    open  http
```

```
Nmap scan report for fing.uncu.edu.ar (179.0.134.129)
Host is up (0.038s latency).
```

```
PORT      STATE SERVICE
80/tcp    open  http
```

Se procede a analizar la ip de la FING (179.0.134.129)

```
-----
-
+ Target IP:          179.0.134.129
+ Target Hostname:    179.0.134.129
+ Target Port:        80
+ Start Time:         2025-05-12 15:18:50 (GMT-3)
-----
-
+ Server: No banner retrieved
+ Uncommon header 'x-frame-options' found, with contents: sameorigin
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ File/dir '/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ "robots.txt" contains 1 entry which should be manually viewed.
+ OSVDB-3092: /license.txt: License file found may identify site software.
+ 6544 items checked: 0 error(s) and 4 item(s) reported on remote host
+ End Time:           2025-05-12 15:23:30 (GMT-3) (280 seconds)
-----
-
+ 1 host(s) tested
```

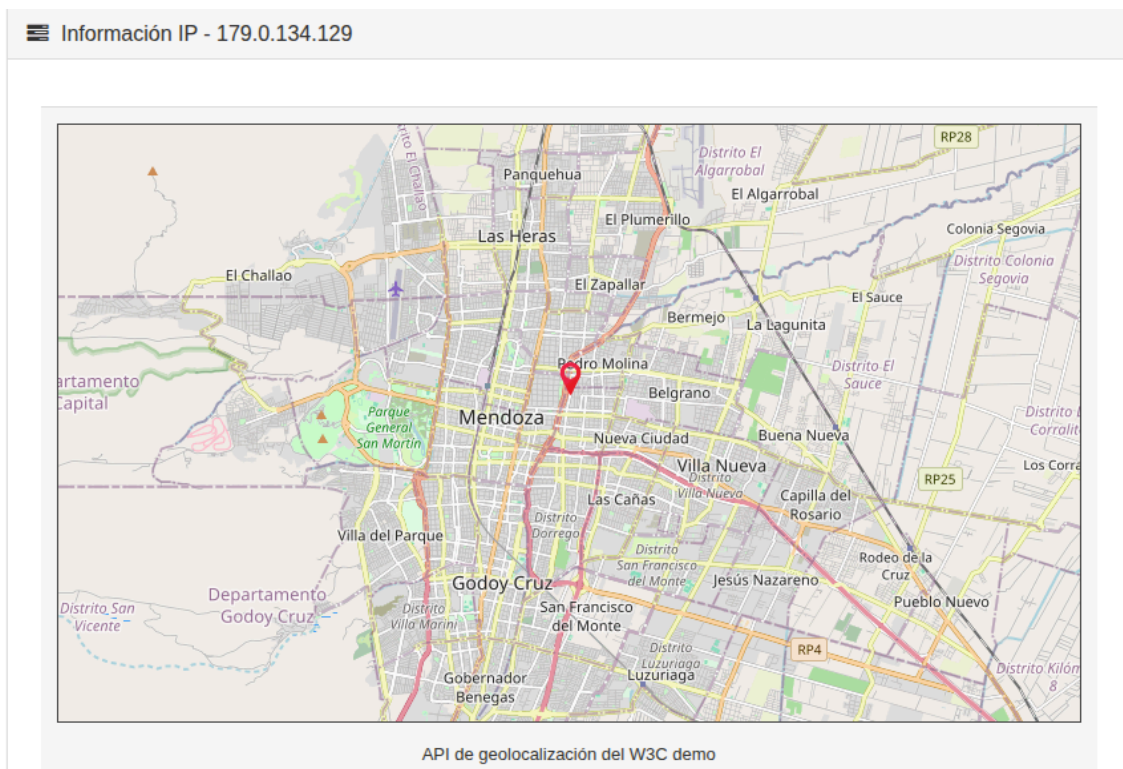

Al pregunta a Gemini sobre los resultados dados, nos da la siguiente información

Análisis de los Resultados de Nikto:

- **Server: No banner retrieved:** Esto significa que el servidor web al que Nikto se conectó no reveló información sobre el software del servidor (por ejemplo, Apache, Nginx, IIS) ni su versión. Si bien esto no es una vulnerabilidad en sí misma, la falta de esta información puede dificultar la identificación de vulnerabilidades específicas del servidor.
- **+ Uncommon header 'x-frame-options' found, with contents: sameorigin:** Este encabezado de respuesta HTTP es una medida de seguridad recomendada. El valor `sameorigin` indica que la página solo se puede incrustar en un `<iframe>` si el sitio que lo contiene tiene el mismo origen (mismo protocolo, dominio y puerto). Esto ayuda a prevenir ataques de Clickjacking. **En este caso, la presencia de este encabezado es una buena práctica de seguridad, no una vulnerabilidad.**
- **+ No CGI Directories found (use '-C all' to force check all possible dirs):** Nikto no encontró directorios de Common Gateway Interface (CGI) conocidos. Los directorios CGI a menudo contienen scripts que pueden ser vulnerables si no están actualizados o configurados correctamente. El mensaje sugiere que podrías usar la opción `-C all` para realizar una búsqueda más exhaustiva en todos los directorios posibles, aunque esto podría generar mucho ruido (falsos positivos) y llevar más tiempo. **La ausencia de directorios CGI comunes no implica necesariamente que no haya scripts vulnerables en otras ubicaciones.**
- **+ File/dir '/' in robots.txt returned a non-forbidden or redirect HTTP code (200):** El archivo `robots.txt` se encuentra en la raíz del sitio web y generalmente contiene instrucciones para los rastreadores web (como los de los motores de búsqueda) sobre qué partes del sitio no deben indexar. Un código de respuesta HTTP 200 significa que el archivo se encontró y se puede acceder a él públicamente. **Esto no es una vulnerabilidad en sí misma, pero el contenido del archivo `robots.txt` podría revelar rutas o directorios que los atacantes podrían encontrar interesantes para investigar.**
- **+ "robots.txt" contains 1 entry which should be manually viewed.:** Nikto detectó una entrada en el archivo `robots.txt` que considera importante revisar manualmente. **Es crucial que examines el contenido de este archivo.** Podría revelar directorios de administración, áreas en desarrollo o cualquier otra información sensible que no debería ser pública.

- **+ OSVDB-3092: /license.txt: License file found may identify site software.:** Nikto encontró un archivo llamado `license.txt`. Si bien tener un archivo de licencia no es una vulnerabilidad, su contenido podría revelar información sobre el software o las bibliotecas que utiliza el sitio web. Esta información podría ser utilizada por un atacante para buscar vulnerabilidades conocidas en esas versiones específicas del software. **Es recomendable revisar el contenido de este archivo para ver qué información revela.**
- **+ 6544 items checked: 0 error(s) and 4 item(s) reported on remote host:** Nikto realizó un total de 6544 pruebas y encontró 4 elementos que considera dignos de mención, sin errores durante el escaneo.
- **+ End Time: 2025-05-12 15:23:30 (GMT-3) (280 seconds):** Indica la hora de finalización del escaneo y la duración del mismo.

Según www.geolocation.com la ip de la FING se encuentra en Mendoza.



4.4:

De las ip de Corea del Norte entre 175.45.176.0 y 175.45.179.255, de 1024 ip's solo se encontraron 5 con el puerto 80 abierto

```
Nmap scan report for 175.45.176.69
Host is up (0.40s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for 175.45.176.85
Host is up (0.39s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for 175.45.176.91
Host is up (0.39s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for 175.45.177.1
Host is up (0.40s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for 175.45.177.11
Host is up (0.40s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1024 IP addresses (14 hosts up) scanned in 93.27 seconds
```