TP N°4 - Redes

Integrantes: Ramirez Victor , Olivares Agustin , Ruiz Joaquín.

Actividad 1- CHAT UDP

Este programa implementa un sistema de chat simple que permite a los usuarios en la misma red local comunicarse entre sí.

1. El programa establece los parámetros necesarios para la comunicación, como la dirección de broadcast (en este caso `192.168.0.255`) y el puerto (`60000`) que se utilizarán para enviar y recibir los mensajes.
2. Al iniciar el programa, se solicita al usuario que ingrese un nombre. Este nombre se usará para identificar sus mensajes dentro del chat.
3. La función de envío permite al usuario escribir mensajes que serán enviados a todos los demás usuarios en la red. Si el usuario escribe "exit", el programa termina y se desconecta del chat.
4. La recibir función escucha continuamente los mensajes entrantes y los muestra en pantalla. Además, notifica cuando un usuario se une o abandona el chat.
5. Para manejar el envío y la recepción de mensajes simultáneamente, el programa utiliza hilos. Esto permite que ambas funciones operen al mismo tiempo sin interferencias.

El programa facilita la comunicación entre múltiples usuarios en la misma red local mediante mensajes enviados y recibidos usando el protocolo UDP y la dirección de broadcast, asegurando que todos los participantes vean los mensajes.

Instrucciones de funcionamiento:

- Descargue el archivo del código fuente y descomprimirlo si es necesario
- Abra una terminal
- Navegue hasta el directorio donde se encuentra el archivo del programa (ChatUDP.py).
- Ejecute el programa con el comando ‘python ChatUDP.py’
- Para salir del chat, escriba "exit" y presione Enter.

Nota: Asegúrese de que su red local permita el tráfico de broadcast.

Actividad 2- CHAT TCP

Este programa consta de dos partes, el lado del cliente y el del servidor. permite a un usuario conectarse a un servidor de chat en la red local para enviar y recibir mensajes. Es importante tener en cuenta que es necesario correr primero el programa ‘ServidorTCP.py’ antes de ejecutar algún cliente.

1. Configuración del Cliente:
   - El usuario ingresa su nombre y el programa se conecta a un servidor específico usando una dirección IP (`25.41.61.1`) y un puerto (`60000`).
   - La función ‘recive_message’ escucha continuamente los mensajes entrantes del servidor. Si el servidor solicita el nombre de usuario, lo envía. Si recibe un mensaje de desconexión, cierra la conexión y termina el hilo.
   - La función ‘write_message’ permite al usuario enviar mensajes al servidor. Si el usuario escribe "exit", la conexión se cierra y el hilo termina.
   - El programa utiliza hilos para manejar simultáneamente el envío y la recepción de mensajes. El hilo de escritura es declarado como ‘demonio’ para poder cerrarlo al ejecutar el comando “exit”

2. Configuración del Servidor:
   - El servidor se inicia en una dirección IP (`25.41.61.1`) y un puerto (`60000`). Permanece a la espera de conexiones de los clientes.
   - La función ‘broadcast’ envía un mensaje a todos los clientes conectados excepto al cliente que envió el mensaje.
   - La función ‘message_handler’ gestiona los mensajes entrantes de los clientes. Si un cliente envía un mensaje de salida, el cliente se desconecta y se notifica a los demás usuarios.
   - La funcion ‘recive_connection’ permite al servidor aceptar nuevas conexiones de clientes, solicitar sus nombres de usuario y crear un hilo para manejar sus mensajes.
   - El servidor utiliza hilos para manejar múltiples conexiones de clientes simultáneamente.

Instrucciones de Ejecución:

- Descargue los archivos del código fuente y descomprimirlos si es necesario.
- Abra una terminal.
- Navegue hasta el directorio donde se encuentra el archivo del programa (‘ServidorTCP.py`).
- Ejecute el programa con el comando: ’python ServidorTCP.py’
- Desde otra PC ejecute el comando: ‘python ClienteTCP.py’
- Ingrese un nombre de usuario cuando se le solicite.

Nota: Antes de ejecutar el código, revisar que la IP del programa cliente sea la IP seteada en el servidor.

Actividad 4.4

Se escanearon 100 direcciones IP aleatorias utilizando tanto escaneo TCP (-sS) como UDP (-sU) en los 5 puertos más comunes (--top-ports 5). Se encontró un host con el puerto http open:

```
Nmap scan report for 111.172.236.17
Host is up (0.38s latency).

PORT    STATE         SERVICE
21/tcp  filtered      ftp
22/tcp  filtered      ssh
23/tcp  filtered      telnet
80/tcp  open          http
443/tcp open          https
123/udp filtered      ntp
137/udp open|filtered netbios-ns
138/udp open|filtered netbios-dgm
161/udp open|filtered snmp
631/udp filtered      ipp

Nmap scan report for 116.37.182.70
Host is up (0.34s latency).
```

El comando WHOIS nos dió información sobre la dirección IP 111.172.236.17 , está asignada a CHINANET HUBEI PROVINCE NETWORK, una red de China Telecom.

```
inetnum:        111.172.0.0 - 111.175.255.255
netname:        CHINANET-HB
descr:          CHINANET HUBEI PROVINCE NETWORK
descr:          China Telecom
descr:          No.31,jingrong street
descr:          Beijing 100032
country:        CN
admin-c:        CHA1-AP
tech-c:         CHA1-AP
abuse-c:        AC1573-AP
status:         ALLOCATED PORTABLE
remarks:        service provider
remarks:        ------------------------------------------------------
remarks:        To report network abuse, please contact mnt-irt
remarks:        For troubleshooting, please contact tech-c and admin-c
remarks:        Report invalid contact via www.apnic.net/invalidcontact
remarks:        ------------------------------------------------------
mnt-by:         APNIC-HM
mnt-lower:      MAINT-CHINANET-HB
mnt-irt:        IRT-CHINANET-CN
last-modified:  2021-06-15T08:05:50Z
source:         APNIC
```

Actividad 4.5

Para realizar el escaneo a todas las IP de Corea del Norte primero se tuvo que buscar el rango de IPs, el cual comienza en 175.45.176.0 y termina en 175.45.179.255. Luego ejecutamos el siguiente comando:

sudo nmap -sS -sU --top-ports 5 -Pn 175.45.176.0/22 > resultados_escaneo.txt

Parámetros:

- -sS: Realiza un escaneo SYN.
- -sU: Realiza un escaneo de puertos UDP.
- --top-ports 5: Escanea los 5 puertos más comunes, tanto para TCP como para UDP.
- -Pn: Desactiva el ping previo a los hosts, asumiendo que están activos.
- 175.45.176.0/22: Especifica el rango de direcciones IP a escanear. En este caso, el rango cubre desde 175.45.176.0 hasta 175.45.179.255, abarcando un total de 1024 direcciones IP el cual pertenece a las IP de Corea del Norte.
- > resultados_escaneo.txt: Redirige la salida del comando al archivo resultados_escaneo.txt, guardando allí los resultados del escaneo.

Las siguiente IP tiene el puerto SSH abierto:

```
9737 Nmap scan report for 175.45.178.137
9738 Host is up (0.44s latency).
9739
9740 PORT    STATE         SERVICE
9741 21/tcp  filtered      ftp
9742 22/tcp  open          ssh
9743 23/tcp  filtered      telnet
9744 80/tcp  filtered      http
9745 443/tcp filtered      https
9746 123/udp open|filtered ntp
9747 137/udp open|filtered netbios-ns
9748 138/udp open|filtered netbios-dgm
9749 161/udp open|filtered snmp
9750 631/udp open|filtered ipp
```

La siguiente IP tiene el puerto http y https abierto:

```
1067 Nmap scan report for 175.45.176.71
1068 Host is up (0.47s latency).
1069
1070 PORT    STATE         SERVICE
1071 21/tcp  filtered      ftp
1072 22/tcp  filtered      ssh
1073 23/tcp  filtered      telnet
1074 80/tcp  open          http
1075 443/tcp open          https
1076 123/udp open|filtered ntp
1077 137/udp open|filtered netbios-ns
1078 138/udp open|filtered netbios-dgm
1079 161/udp open|filtered snmp
1080 631/udp open|filtered ipp
```

Actividad 5

Se implementaron dos scripts para la transferencia de archivos entre un cliente y un servidor mediante sockets TCP. El servidor escucha las conexiones entrantes y envía archivos solicitados por el cliente. El cliente se conecta al servidor, recibe el archivo y lo guarda localmente. Ambos códigos manejan la transmisión de datos en bloques de 1MB para asegurar una transferencia eficiente y continua.

Funcionamiento del Servidor:

- Define la dirección IP del servidor y el puerto.
- Crea un socket de servidor, lo vincula a la dirección y puerto especificados, y lo pone en modo de escucha.
- Acepta conexiones entrantes en un bucle continuo.
- Solicita al usuario el nombre del archivo que desea transferir.
- Llama a la función send_file para enviar el archivo al cliente conectado.
- Cierra la conexión del socket.

Funcionamiento del Cliente:

- Define la dirección IP del servidor y el puerto.
- Crea un socket de cliente y se conecta al servidor.
- Solicita al usuario el nombre del archivo donde se guardará la información recibida.
- Llama a la función receive_file para recibir y guardar el archivo.
- Cierra la conexión del socket.