Trabajo Práctico

Segundo cuatrimestre de 2023

Introducción

En este Trabajo Práctico se explorarán algunas herramientas comunes en el análisis de las redes. El objetivo es implementar la herramienta **Traceroute** (vista en clase) usando la librería de Python **Scapy**[1]**.** Además, se deberá implementar un **Port Scanner**. Por último, experimentamos con las herramientas desarrolladas, comparándolas con las otras implementaciones existentes.

Primera parte: **Traceroute**

Traceroute es una herramienta que permite mostrar la ruta entre un host origen y uno remoto, mostrando el RTT de enviar un paquete a cada uno de los sucesivos hosts de la ruta.

Traceroute es implementada aprovechando el campo **TTL (time-to-live)** del encabezado de los paquetes IP. Al enviar un paquete, el host origen setea un valor inicial de TTL. Un router, antes de reenviar un paquete en la red, decrementa y analiza el campo TTL de ese paquete. Si el TTL llega a 0, el router descarta el paquete, y potencialmente envía un mensaje de error (ttl-zero-during-transit) al host cuya dirección aparece en el campo **SRC** del paquete. Para implementar **traceroute** entonces, basta con enviar al host destino una serie de paquetes IP cuyo TTL se va incrementando. Si los hosts intermedios responden mensajes de error, tendremos las direcciones IP de los hosts intermedios.

Para comenzar con nuestra implementación de **traceroute** , podemos tomar como base el siguiente fragmento de código para enviar un mensaje *ping* a un host:

```
from scapy.all import *
direccionIpDst = "www.utdt.com"
packet = IP(dst= "www.utdt.com", ttl = 64)/ICMP(type=8, code=0)
resp = sr1(packet, timeout = 10)
```

Ejercicios:

1. Implementar Traceroute. El programa debe estar implementado en Python 3, utilizando Scapy. El mismo debe correrse desde la terminal de comandos, recibiendo como único parámetro el host destino.

[1] https://scapy.readthedocs.io/en/latest/

Segunda parte: **Port Scanning**

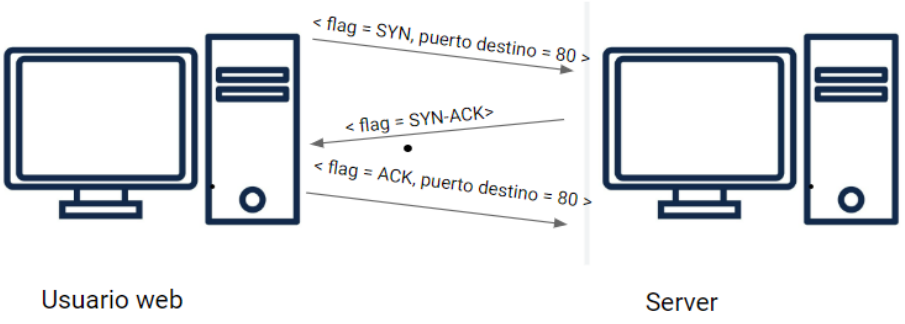
Port Scanning es un conjunto de técnicas para analizar el estado de los puertos de un host. Las mismas se basan en el **3-way-handshake,** protocolo necesario para establecer conexiones TCP.

Definimos el estado de un puerto como el tipo de conexiones que permite. En particular, los estados que consideraremos son los siguientes:

- open: un puerto está *abierto* cuando una aplicación se encuentra aceptando conexiones en ese puerto.
- close: un puerto está *cerrado* cuando, al intentar conectarse con él, se recibe una respuesta negativa.
- filtered: un puerto está *filtrado* cuando al intentar conectarse con él, no obtenemos ninguna respuesta.

Si quisiéramos observar el estado de cierto puerto de un servidor podríamos intentar realizar un **3-way-handshake** y observar cómo responde el servidor a nuestro intento de conexión.

Consideren el siguiente gráfico, que muestra de forma resumida un 3-way handshake entre un usuario y un web server. Es importante notar que evitamos campos no relevantes de los segmentos TCP.



En la imagen se puede observar como un usuario web quiere conectarse al puerto 80 de un servidor (enviando un segmento con el flag **SYN**) y éste responde de forma positiva (ver el flag de su respuesta, **SYN-ACK**)**.** Podemos inferir entonces que el puerto 80 del servidor se encuentra **abierto**.
El envío del primer mensaje del three-way-handshake se puede realizar con las siguientes líneas de código:

```
from scapy.all import *
dest_port = 80
packet = IP(dst= "www.utdt.com")/TCP(flags="S", dport = dest_port)
resp = sr1(packet, timeout = 10)
resp.show()
```

En la segunda línea se elige el puerto al que se enviará el paquete. En la tercera línea se crea un paquete, modificando el campo "dst" para enviar un mensaje al servidor de la universidad, y se modifica el puerto de destino (*dport).* También se asegura de enviar el paquete con el flag "S", para informarle al destinatario del paquete que se desea iniciar una conexión.

Ejercicios:

2.1. Implementar un ***port scanner*** que reciba como único parámetro una dirección IP de un host y realice un escaneo de todos los puertos menores a 1000 en el host. El scanner debe considerar como abiertos **aquellos puertos en los que el host que responda el mensaje SYN con un SYN-ACK** (es decir, realizar las primeras dos partes del 3-way handshake). El estado de los puertos debe ser guardado en un archivo de texto. El output del programa debe ser el porcentaje de los puertos abiertos y filtrados. El scanner debe estar implementado en Python 3, usando Scapy.

2.2. Extender el ***port scanner*** anterior de forma tal que tome un "*char*" como un segundo parámetro. Si el caracter es un -h se ejecutará el scanner del ejercicio 2.1. Si el caracter es un -f, se ejecutará una variante del scanner que **considerará un puerto abierto cuando se puede establecer una conexión TCP completa** (es decir, las 3 partes del 3-way handshake). Para confirmar esto, deberán agregar **payload** al tercer mensaje del handshake, y esperar recibir conformación (un **AKC**) del payload.

Tercera Parte: Experimentación e Informe

El último punto del TP consiste en realizar un informe sobre las herramientas desarrolladas. Las preguntas y experimentos detallados a continuación consisten en lo ***mínimo indispensable*** a incluir en el informe para aprobar. Tienen libertad de agregar cuantos experimentos quieran. El informe debe tener un máximo de 15 páginas incluyendo la carátula.

Ejercicios:

3.1. Ejecutar su ***traceroute*** trazando las rutas a 5 universidades de diferentes continentes. Responder las siguientes preguntas:

a. ¿Qué porcentaje de hosts intermedios envían mensajes de ttl-zero-during-transit? Investigar y dar posibles razones para este comportamiento.

b. Para cada una de las rutas descubiertas, analizar la diferencia entre los RTT de diferentes saltos (es decir, la diferencia entre los RTT de dos hosts sucesivos en una ruta). ¿Hay algún salto grande entre el RTT de un host de la ruta y el siguiente? ¿A que creen que se puede deber?

3.2 Ejecutar ambos ***Port Scanners*** desarrollados sobre las páginas web de 3 universidades distintas.

a. Realizar un análisis estadístico sobre el porcentaje de puertos cerrados, abiertos y filtrados. Para realizar el análisis sugerimos graficar e intentar explicar los resultados obtenidos.
b. ¿Encontraron patrones en los estados de los puertos en diferentes servidores? ¿A qué se debe?
c. Analizar las diferencias en los resultados de escanear un mismo servidor con los dos diferentes scanners desarrollados.
d. ***nmap***[2] es una herramienta conocida que permite realizar Port Scanning. Comparar los resultados de escanear servidores usando ***nmap*** con los del Port Scanner que desarrollaron.

Entrega y consideraciones finales

Deberán entregar un solo archivo en formato .zip. Dentro de él deberá haber únicamente 2 archivos .py con las herramientas desarrolladas y un archivo pdf con el informe.

El TP debe ser entregado mediante el campus virtual, hasta las 11:59 pm del domingo 29 de octubre de 2023. No se aceptarán TPs entregados después de este horario.

[2] https://nmap.org/