Universidad
Carlos III de Madrid

PROYECTO FINAL: DETECTOR DE ESCANER DE PUERTOS

Trabajo realizado por:

- José Serrano Martínez 100366667@alumnos.uc3m.es

Índice

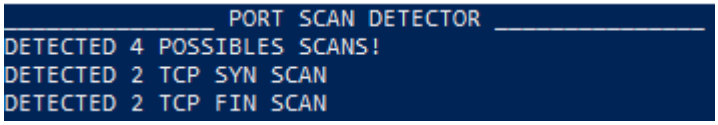
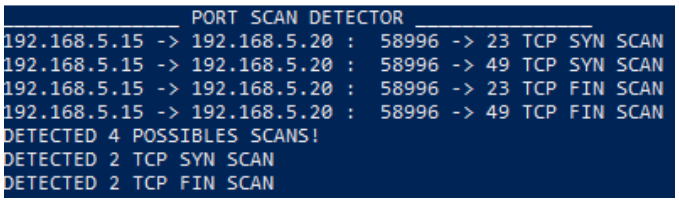
1. Introducción.

En este documento va a encontrar los pasos realizados para llevar a cabo la implementación de la segunda opción de la práctica final de Ingeniería de la seguridad. El lenguaje que se ha empleado para realizar esta implementación es Python y la librería seleccionada para realizar el tratamiento de los paquetes es “dpkt”. También se han incorporado otras librerías para mejorar los resultados obtenidos, como la librería “socket” o la librería “subprocess”.

El programa implementado se ejecutará por defecto con el siguiente comando:

python .\readScans.py .\SCAN_FINSYN.pcap

Al comando anterior podemos añadirle la opción “-vv” para obtener un mayor detalle en la información dada por el programa. El uso de esta opción modificará el output devuelto por el programa al siguiente:

python .\readScans.py SCAN_FINSYN.pcap	**python .\readScans.py SCAN_FINSYN.pcap -vv**
	

Pero ¿por qué debemos de detectar los escaneos de puertos realizados sobre nuestro ordenador o conjunto de ordenadores en nuestra red empresarial?

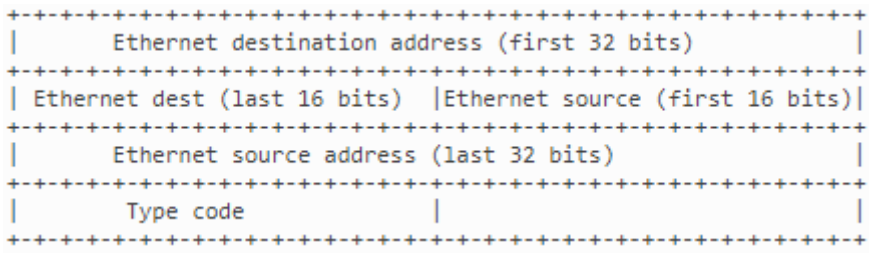
Generalmente, si realizamos una comparación con la vida real, un ladrón antes de entrar a una casa comprueba si tiene alguna puerta o ventana abierta. Al tratarse de una actividad no ilegal, puede llegar a generar una denegación de un servicio si se realiza de forma continuada, pero es poco común, un escaneo de puertos le proporciona la información sobre que “puertas o ventanas”, puertos del ordenador, se encuentran abiertos. Además, esta información suele poder almacenarse para un futuro ataque y no ser necesario explotar esta vulnerabilidad inmediatamente. Pero cada vez que existen estos puertos abiertos en el ordenador, existe la posibilidad de pérdida de datos, aparición de software malicioso o, en caso de encontrarnos en una red de ordenadores, poner en riesgo esta red.

Por eso es aconsejable tener aquellos puertos que no usemos siempre cerrados, para evitar que un posible atacante pueda explotar una vulnerabilidad producida por la existencia de un puerto abierto. Pero es común la existencia de puertos abiertos en muchos ordenadores personales por la falta de conocimiento sobre cómo cerrar estos puertos o que se encuentren abiertos por defecto y el usuario no sea consciente de ello. Por ello es obligatoria y necesaria la existencia de estos detectores de escaneos de puertos para notificar al usuario cuándo su información personal se pueda ver comprometida.

A continuación, conoceremos el formato de cada uno de los paquetes que trataremos en el detector de escaneo de puertos.

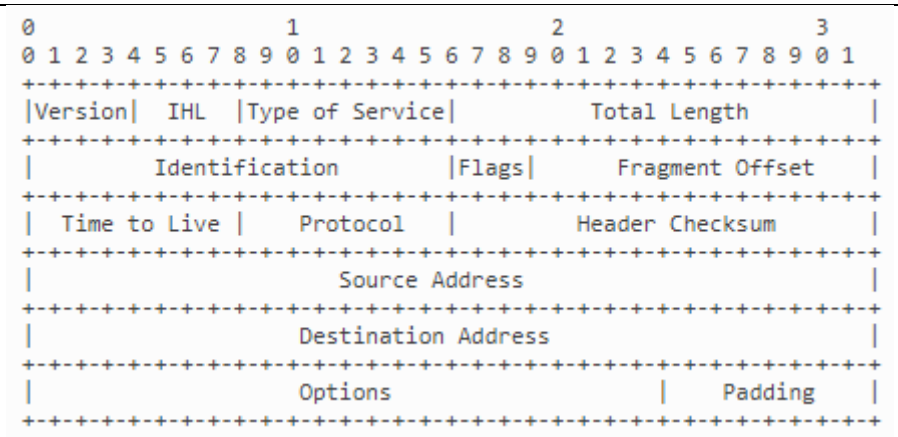
Comenzaremos conociendo el paquete de red ETHERNET. En este paquete encontramos información sobre la dirección ip origen desde la que se envía el paquete, la dirección ip destino, así como el tipo de protocolo IP o ARP que emplea el paquete. De este paquete empelaremos la información relacionada con el protocolo empleado, así como, las ips origen y destino del paquete.

ETHERNET



Este paquete está relacionado con el protocolo IP, de este paquete obtendremos información sobre el tipo de protocolo TCP o UDP empleado por el paquete, el cual se encuentra en el campo protocol de nuestro paquete IP. Sin embargo, podemos obtener más información contenida en este paquete como: la longitud del datagrama, contenida en total length, o información acerca de su tiempo de vida contenida en el campo Time to Live, también conocido como TTL, en el que si el campo alcanza el valor 0 el datagrama debe de ser descartado.

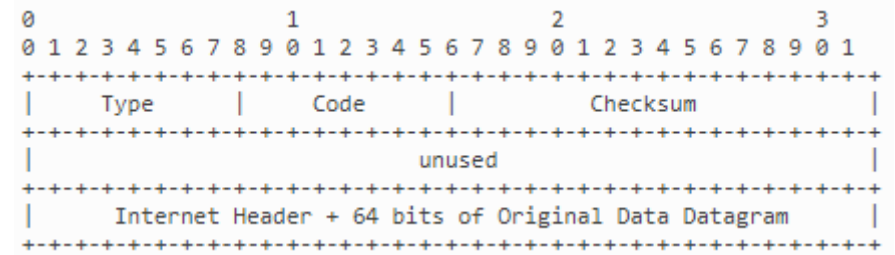
IP



El protocolo ICMP o protocolo de control de mensajes es parte del conjunto de protocolos IP. Es empleado para enviar mensajes de error acerca del estado de un host: el host no se encuentra o un servicio no está disponible. A diferencia de otros protocolos como el TCP y el UDP, este protocolo no es usado para intercambiar información entre sistemas ni tampoco aplicaciones de usuario. Tendremos que tratar con este paquete cuando el usuario realice acciones de ping desde una máquina a otra. Nos basaremos en la información que nos transmite el dato “type”, donde dependiendo del valor sabremos que tipo de respuesta nos está dando. Los valores más comunes en este campo de datos son:

- 0: respuesta echo.
- 3: destino no disponible, no es posible establecer conexión.
- 8: petición echo.

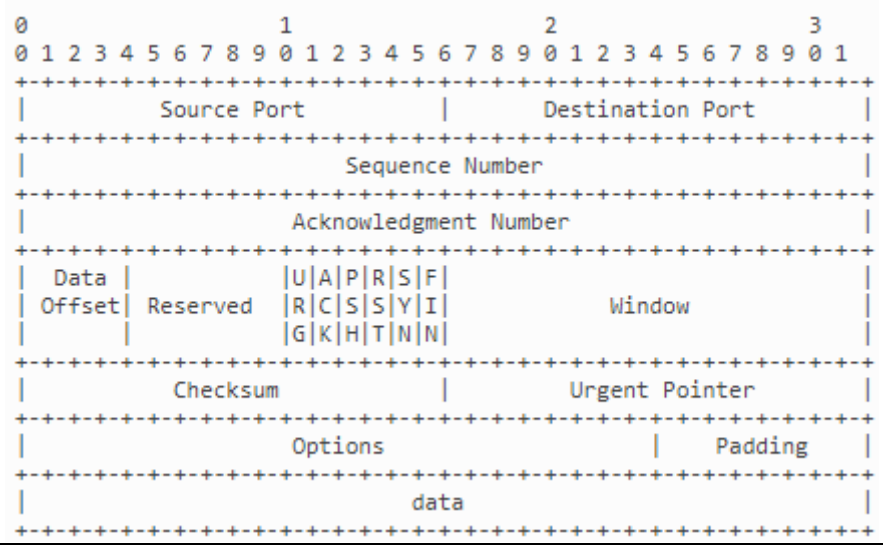
ICMP



Hablaremos ahora del protocolo TCP, un protocolo de la capa de transporte de Internet, fiable y orientado a la conexión. Se dice que está orientado a la conexión porque antes de que un proceso de la capa de aplicación comience a enviar datos a otro proceso, los dos procesos deben de haber establecido una comunicación preliminar entre ellos. Se le denomina fiable porque cualquier pérdida de datos se detecta y se resuelve.

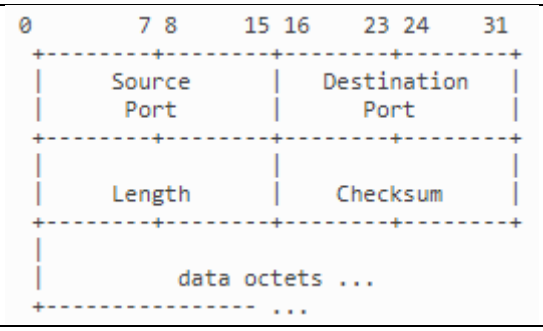
Esta estructura contiene información sobre, el puerto de origen y puerto destino, el número de secuencia, número de reconocimiento, así como las banderas que se encuentran activadas y el tamaño de la ventana. Toda esta información es de gran ayuda para determinar si se está haciendo algún tipo de escaneo en TCP, ya que, como se verá más adelante, los escaneos suelen contar con un número bajo de paquetes enviados, así como determinado tipo de banderas activadas.

TCP



Finalmente, tendremos que tratar, dependiendo del tipo de escaneo realizado, con el paquete UDP. En este paquete no encontramos información sobre las banderas a diferencia del paquete TCP. En este paquete nos centraremos en la información proporcionada por los campos de datos “Source Port”, puerto de origen y “Destination Port”, puerto destino.

UDP

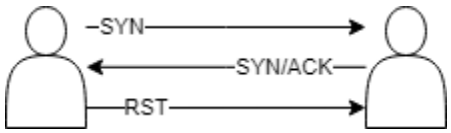


2. Técnicas de escaneo seleccionadas:

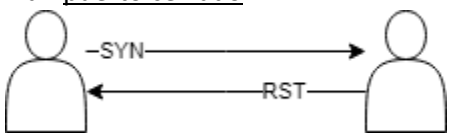
2.1. TCP SYN:

Esta técnica de escaneo de puertos es la opción más popular para detectar los puertos abiertos. Es capaz de realizar miles de escaneos de puertos por segundo en una red que no se vea obstaculizada por la existencia de Firewalls. Este tipo de escaneo es discreto y sigiloso al no completar nunca la conexión TCP.

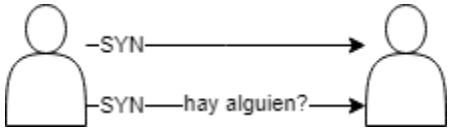
Para detectar esta conexión nos centraremos en obtener el valor de la bandera SYN, y comprobar que el resto de las banderas no están activadas. Ejemplo del funcionamiento del escaneo TCP SYN:
En caso de que el puerto se encuentre abierto:



En caso de que nos encontremos con un puerto cerrado:



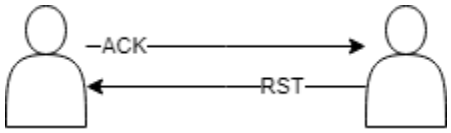
Finalmente, en caso de no recibir ninguna respuesta por parte del puerto atacado (no es común recibir este tipo de situaciones en un escaneo):



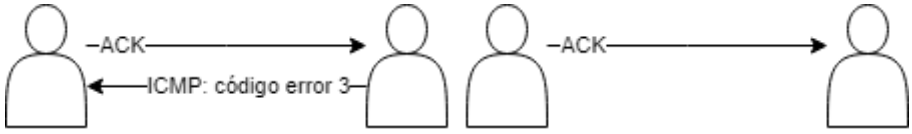
Como podemos observar este tipo de escaneo tiene unos paquetes aproximados de máximo 2 paquetes enviados, por lo que en el código filtraremos este tipo de escaneo basándonos en el número de paquetes origen-destino mismos puertos, con un número de secuencia superior al paquete enviado con anterioridad.

2.2. TCP ACK:

Este tipo de escaneo es diferente al resto de escaneos realizados ya que no determina si un puerto está abierto. Se centra en realizar un mapeo de las reglas de los firewalls en cada uno de los distintos puertos. Este tipo de escaneo envía un paquete TCP con únicamente el bit de la bandera ACK puesto. Sin importar si el puerto está abierto o cerrado, el puerto objetivo está obligado a responder con un paquete reset. Por otro lado, aquellos firewalls que bloquean la conexión normalmente no responden, o en caso de responder, devuelven un paquete ICMP indicando que no se ha podido establecer a conexión. A continuación, encontraremos un ejemplo de este tipo de escaneo:
Este tipo de escaneo determina que el puerto no está filtrado por el firewall:



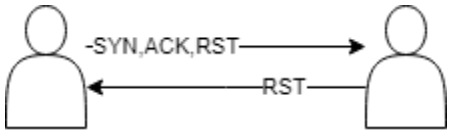
En caso de no recibir respuesta o recibir una respuesta ICMP se determina que este puerto está controlado por el firewall:



2.3. TCP XMAS:

Este tipo de escaneo busca explotar una pequeña rendija o escapatoria en el documento TCP RFC que permite diferenciar entre puertos abiertos y cerrados. En la página 65 del documento RFC 793 se indica: “si el puerto destino se encuentra cerrado, y el paquete que llega no contiene la bandera reset activada, se debe de enviar un paquete reset con la bandera activada”. Este tipo de escaneo es útil contra aquellos sistemas que cumplan

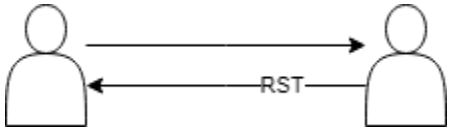
el documento RFC 793, con lo que todos los paquetes que no contengan la bandera SYN, ACK o RST activada provoca una devolución por parte del puerto escaneado de un paquete con la bandera RST si el puerto está cerrado. Este tipo de escaneo envía un paquete con las banderas: FIN, PSH y URG activadas, es por ello por lo que se le denomina XMAS, por que decora el paquete como un árbol de navidad, en inglés, Christmas (XMAS) Tree.
Un ejemplo a partir del cual se denomina que el puerto está cerrado:



Sin embargo, un problema de este tipo de escaneos es que no es capaz de determinar si un puerto del que no recibe respuesta se encuentra abierto o filtrado, por lo que para encontrar los puertos cerrados este tipo de escaneo es útil, pero es incapaz de determinar los puertos que se encuentran abiertos.

2.4. TCP NULL:

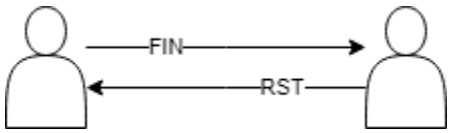
Este tipo de escaneo es similar al anterior, presenta las mismas peculiaridades, lados positivos y lados negativos. Es capaz de determinar si un puerto se encuentra cerrado basado en la respuesta que le da el puerto. Como se ha indicado anteriormente si cumplen con el documento RFC 793, en caso de estar cerrados y no haber recibido una petición sin la bandera RST activada, deben de responder con un paquete en el que la bandera RST se encuentra activada.
Este tipo de escaneo se diferencia del anterior en las banderas que se encuentran activadas en paquete TCP enviado. En este tipo de escaneo no se envía ninguna bandera activada.
Para determinar si un puerto se encuentra cerrado debe de ocurrir la siguiente comunicación:



Al igual que ocurre con el escaneo XMAS, este tipo de escaneo es capaz de determinar si un puerto está cerrado, sin embargo, es incapaz de diferenciar entre los puertos abiertos y filtrados cuando no existe ninguna respuesta por parte del puerto escaneado.

2.5. TCP FIN:

Este tipo de escaneo es similar a los dos escaneos anteriores en cuanto que tiene las mismas ventajas y desventajas. Es capaz de determinar si un puerto se encuentra cerrado basado en la respuesta que le da el puerto. Como se ha indicado anteriormente si cumplen con el documento RFC 793, en caso de estar cerrados y no haber recibido una petición sin la bandera RST activada, deben de responder con un paquete en el que la bandera RST se encuentra activada.
Este tipo de escaneo se diferencia de los anteriores en las banderas que tiene activadas y desactivadas, en este caso, la única bandera que se encuentra activada es la bandera de FIN.
Para determinar si un puerto está cerrado debe de ocurrir la siguiente comunicación:



Al igual que ocurre con los anteriores es incapaz de determinar si un puerto que no devuelve una respuesta se encuentra abierto o filtrado.

2.6. UDP:

Generalmente, muchos gestores de seguridad obvian este tipo de escaneo al ser poco común y más lento en comparación con los tipos de escaneos basados en paquetes TCP. No por ello este tipo de escaneo es menos peligroso que los anteriores ya que un atacante puede obtener mucha información de aquellos puertos basados en el protocolo UDP.

3. Ejecución del programa:

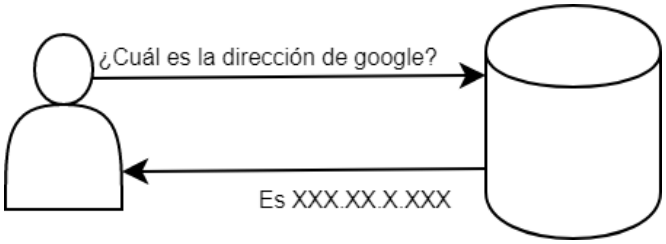
Para generar los distintos tipos de tráfico dentro de una red, he empleado la herramienta nmap para la generación de tráfico y su captura la realizo con la herramienta tcpdump. Comenzaremos explicando las herramientas empleadas, nmap y tcpdump, y finalmente expondremos como hemos generado el tráfico en nuestra red.

3.1. NMAP:

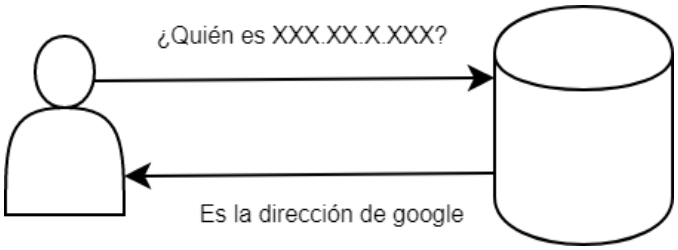
Nmap, Network Mapper, es una herramienta de código abierto para la exploración de redes y auditorías de seguridad. Fue diseñada para escanear rápidamente grandes redes, pero funciona perfectamente contra redes de pequeño tamaño. Esta herramienta presenta un gran número de capacidades como el descubrimiento de hosts, así como el descubrimiento de puertos abiertos. Para el descubrimiento de hosts encontramos las siguientes opciones:

- -sL (escaneo de lista o List scan): este comando enumera cada uno de los hosts de la red especificada, sin enviar ningún paquete de datos a los hosts de destino. De manera predeterminada nmap todavía emplea "reverse DNS". El funcionamiento que sigue este "Reverse DNS" es el siguiente: si tú quieres ir a google, se pregunta al DNS donde se encuentra alojado google. Sin embargo, reverse DNS hace exactamente lo contrario. Consiste en buscar hosts a través de su IP.

  Mientras que DNS funciona de la siguiente manera:



  Reverse DNS funciona de la siguiente manera:



- -sn (no se realiza un escaneo de puertos): esta opción le indica a nmap que no realice un escaneo de puertos.

A continuación, hablaremos de cómo se realiza el descubrimiento de los puertos abiertos. Para ello NMAP categoriza los puertos de la siguiente manera:

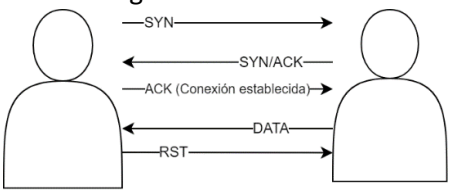
- Abierto: una aplicación acepta conexiones TCP, datagramas UDP o asociaciones SCTP en un puerto específico.
- Cerrado: se puede acceder al puerto, pero no hay ninguna aplicación escuchando.

- Filtrado: no puede determinar que el puerto se encuentre abierto porque el filtrado de paquetes impide que sus sondas lleguen al puerto.
- Sin filtrar: no se puede acceder a un puerto, pero no se puede determinar si se encuentra abierto o cerrado.
- Abierto | Filtrado: cuando no puede determinar si un puerto está abierto o filtrado.
- Cerrado | Filtrado: cuando no puede determinar si un puerto está cerrado o filtrado.

Para realizar los escaneos de puertos encontramos distintas posibilidades, pudiendo realizar una mezcla entre ellos. Los disponibles en nmap son los siguientes:

- TCP SYN Scan: es el tipo de escaneo más común como hemos indicado anteriormente. Para indicarle a NMAP que queremos realizar este tipo de escaneo emplearemos "-sS".

- TCP Connect Scan: cuando el escaneo anterior (SYN) no es una opción, se emplea este tipo de escaneo. A diferencia del escaneo SYN, este tipo de escaneo completa la conexión con aquellos puertos que se encuentran abiertos. Frente al escaneo anterior, nos encontramos con un escaneo más lento y requiere de más paquetes para obtener la misma información, además un IDS (sistema de detección de intrusos) decente, sería capaz de detectarlo con facilidad. Para realizar este tipo de escaneo emplearemos "-sT" junto con nuestro comando de NMAP. Un ejemplo del funcionamiento es el siguiente:



- UDP Scan: pese a que la mayoría de los servicios en internet emplean el protocolo TCP, el protocolo UDP está muy extendido. Como el escaneo UDP es generalmente más lento y más difícil que el escaneo mediante TCP, se suele ignorar dejando la puerta abierta a poder explotar los servicios UDP. Para hacer uso de este tipo de escaneo incorporaremos "-sU" a nuestro comando en NMAP.

- TCP FIN: para hacer uso de este escaneo añadiremos "-sF" a nuestro comando en NMAP.

- TCP NULL: si queremos ejecutar este escaneo incorporaremos "-sN" a nuestro comando en NMAP.

- TCP XMAS: para ejecutar este escaneo añadiremos "-sX" a nuestro comando en NMAP.

- TCP ACK Scan: este tipo de escaneo como se ha dicho anteriormente no determina si un puerto se encuentra abierto o abierto|filtrado, es empleado para realizar un mapeo de las reglas del firewall. Para realizar un escaneo TCP ACK añadiremos "-sA" a nuestro comando de NMAP.

- TCP Window Scan: tiene la misma base que el escaneo anterior, TCP ACK, sin embargo, este tipo de escaneo busca explotar una vulnerabilidad de algunos sistemas para diferenciar los puertos abiertos de los cerrados. Esto lo realiza comprobando el campo Window en el paquete con la bandera Reset devuelto por el puerto. En algunos sistemas los puertos abiertos usan un tamaño positivo de ventana mientras que los puertos cerrados tienen un tamaño de ventana cero. El filtrado de puertos que realiza el este escaneo es el siguiente:

Respuesta	Estado asignado
Respuesta TCP RST con un valor distinto de 0 para el campo de la ventana	Abierto
Respuesta TCP RST con un valor igual a cero para el campo ventana	Cerrado
No se recibe ninguna respuesta	Filtrado

Se recibe un ICMP indicando que no es posible conectarse con el destino	Filtrado

- TCP Maimon Scan: recibe este nombre por su descubridor, Uriel Maimon. Es similar a los escaneos TCP NULL, FIN y XMAS, pero en este caso las banderas del paquete enviado son FIN/ACK. De acuerdo con el documento RFC 793(TCP), un paqeute RST debe de ser devuelto como respuesta al paquete enviado si el puerto está abierto o cerrado. Si embargo, Uriel se percató que muchos sistemas realizaban un drop del paquete si el puerto estaba abierto. NMAP hace uso de esta técnica para determinar si un puerto está abierto o cerrado. La interpretación que devuelve NMAP sobre el estado de los puertos se basa en los siguientes criterios:

Respuesta	**Estado asignado**
No se recibe respuesta después de la transmisión	Abierto|Filtrado
Se recibe un paquete RST	Cerrado
ICMP tipo 3, no es posible establecer la conexión	Fitlrado

Además, una característica interesante de NMAP es la capacidad de editar las banderas que se encuentran activadas en un tipo de escaneo empleado. Por ejemplo, al escaneo SYN podemos indicar que también se encuentre activada la bandera ACK o la bandera FIN mediante el uso de la opción "--scanflags *banderas a activar*". Como, por ejemplo, empleando el siguiente comando:

```
nmap -sS --scanflags SYNFIN *ip*
```

Además, también encontramos el comando "-TX", que sirve para definir el tiempo entre los paquetes. Se encuentra entre 0 y 5, siendo 0 el más lento y 5 el más rápido. Por defecto se ejecutan los escaneos con un valor "-T3". El uso de esta configuración ayuda a evitar ser detectados al aumentar el tiempo entre los envíos de paquetes.

3.2. TCPDUMP:

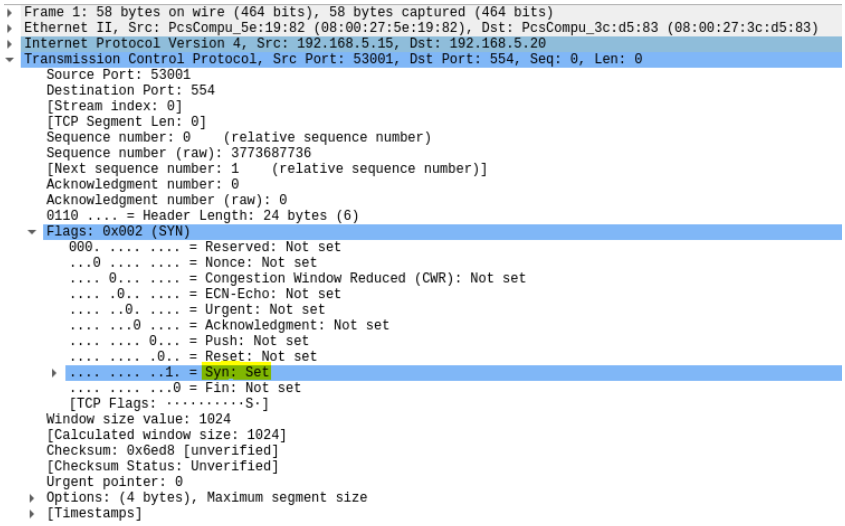
TCPDUMP es una herramienta que se encarga de analizar u observar los paquetes que se envían o reciben a través de una red en una interfaz específica. Para ello encontramos diversas opciones con las que capturar y mostrar esta información:

- "-i": capturamos los paquetes de la interfaz indicada; "-i eth1".
- "-c": finaliza la captura una vez que hemos recibido X paquetes; "-c X".
- "-w": guardar la información capturada en el fichero pasado por parámetro; "-w capture.pcap".
- "-r": lee la información de un fichero.pcap que contenga capturas de paquetes.
- "-XX": captura los datos de cada paquete incluido su encabezado de nivel de enlace en formato hexadecimal y ASCII.
- "-A": muestra el contenido de cada paquete en formato ASCII.

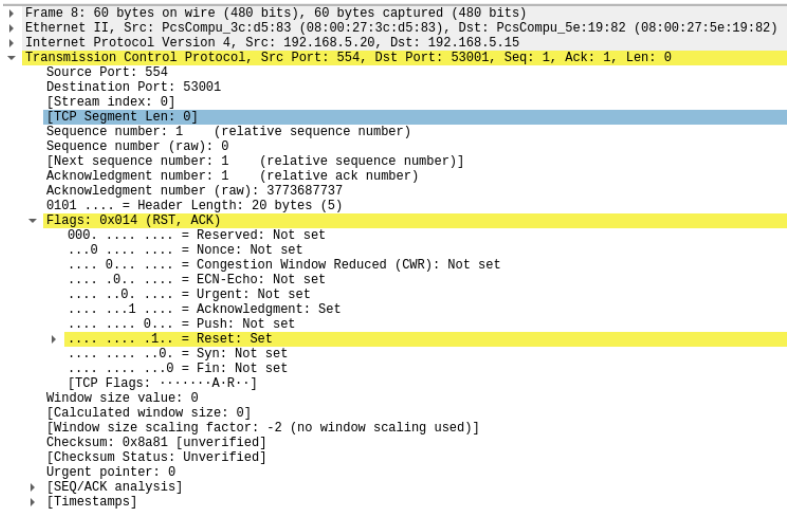
3.3. GENERACIÓN DE TRÁFICO Y CAPTURA:

a) Generación de tráfico para el tipo de escaneo TCP SYN:

Primero comenzaremos preparando la captura de tráfico en la interfaz número 1 de nuestra máquina Kali. Para ello empelaremos el comando en tcp dump: `tcpdump -i eth1 -w SYN_scan.pcap`, con el que guardaremos todo el tráfico capturado en el fichero "SYN_scan.pcap", el cual procederemos a abrir a través de wireshark para comprobar su contenido. Una vez estamos capturando todo el tráfico de red que hay en la interfaz 1, procedemos a crear este tráfico mediante la herramienta nmap, para ello ejecutaremos el siguiente comando: `nmap -Pn -sS 192.168.5.20/24`, una vez ha terminado este tipo de escaneo, finalizamos la captura de tráfico realizada por tcpdump y procedemos a comprobar el contenido de nuestro fichero. Para ello emplearemos la herramienta wireshark, y nos aseguraremos de que en los paquetes tcp, la única bandera que está activa es la bandera SYN:



Como se puede observar para un paquete origen: 192.168.5.15 (máquina atacante), destino: 192.168.5.20 y puerto origen: 53001 y destino: 554, se envía un paquete TCP en el que la bandera SYN se encuentra activada y el resto desactivadas. Si buscamos la respuesta dada por el puerto atacado:



Podemos observar que le envía un paquete con las banderas ACK y RESET activadas, con lo que nuestro atacante puede determinar que este puerto se encuentra cerrado.

Antes de comprobar el fichero que hemos creado anteriormente, pasaremos a probar un fichero de menor tamaño, en el que realizamos una captura sobre los puertos 53 y 21. En este fichero encontramos la siguiente información:

1 0.000000	PcsCompu_5e:19:82	Broadcast	ARP	42 Who has 192.168.5.20? Tell 192.168.5.15
2 0.000561	PcsCompu_3c:d5:83	PcsCompu_5e:19:82	ARP	60 192.168.5.20 is at 08:00:27:3c:d5:83
3 13.082280	192.168.5.15	192.168.5.20	TCP	58 39472 → 53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4 13.082356	192.168.5.15	192.168.5.20	TCP	58 39472 → 21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5 13.083204	192.168.5.20	192.168.5.15	TCP	60 53 → 39472 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
6 13.083235	192.168.5.15	192.168.5.20	TCP	54 39472 → 53 [RST] Seq=1 Win=0 Len=0
7 13.083279	192.168.5.20	192.168.5.15	TCP	60 21 → 39472 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
8 13.083288	192.168.5.15	192.168.5.20	TCP	54 39472 → 21 [RST] Seq=1 Win=0 Len=0

Como se puede observar en las líneas 3 y 4 de nuestro fichero, se está enviando un paquete TCP con la bandera SYN activada, por lo que sin importar la respuesta del servidor, podemos determinar que se trata de un escaneo puertos del tipo TCP Syn Scan o TCP Half-Connect. El resultado devuelto por nuestro programa es el siguiente:

```
192.168.5.15 -> 192.168.5.20 :  39472 -> 53              TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  39472 -> 21              TCP SYN SCAN
DETECTED 2 POSSIBLES SCANS!
```

Si comparamos el output recibido para este fichero, con el contenido del fichero observado a través de la herramienta wireshark, podemos ver que efectivamente se está realizando un escaneo TCP SYN, basándonos en el orden que siguen los paquetes, primero se realiza un envío de un paquete TCP con la bandera SYN activada, se recibe la respuesta por parte del puerto escaneado (que en este caso devuelve un SYN/ACK, lo que indica al atacante que está abierto), y finaliza la conexión enviando un paquete TCP RST.

Para el primer fichero creado, "SYN_scan.pcap", se ha obtenido el siguiente output (se presenta una captura de todo el output recibido ya que su extensión no permite mostrarlo entero):

```
192.168.5.15 -> 192.168.5.20 :  53001 -> 10010           TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 3003            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 481             TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 1098            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 3260            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 1132            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 1108            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 2301            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 1002            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 8181            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 15000           TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 7625            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 50002           TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 5902            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 1057            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  53001 -> 3800            TCP SYN SCAN
```

b) Generación de tráfico para el tipo de escaneo TCP ACK:

Comenzaremos igual que el apartado anterior, preparando la captura de tráfico a través de la herramienta tcpdump, pero esta vez cambiaremos el nombre del fichero a "ACK_scan.pcap". A continuación, procederemos a generar todo el tráfico en nuestra red con el uso de la herramienta nmap a través del comando: `nmap -Pn -sA 192.168.5.20/24`. Una vez ha finalizado el escaneo, finalizamos la captura de tráfico y todo el tráfico capturado lo encontraremos en nuestro fichero ACK_scan.pcap, el cual, procederemos a abrir con la herramienta WireShark para comprobar su contenido y asegurarnos que el tipo de escaneo realizado se corresponde con el escaneo TCP ACK. En este tipo de escaneo, únicamente se debe de tener activada la bandera de ACK, como observaremos a continuación:

```
▾ Transmission Control Protocol, Src Port: 44745, Dst Port: 1723, Seq: 1, Ack: 1, Len: 0
    Source Port: 44745
    Destination Port: 1723
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 0
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 1273442569
    0101 .... = Header Length: 20 bytes (5)
  ▾ Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
    Window size value: 1024
    [Calculated window size: 1024]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xedeb [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [Timestamps]
```

Además, como se ha indicado anteriormente, este tipo de escaneo envía únicamente un solo paquete al puerto que va a ser escaneado, por lo que, en nuestro código, para detectar este tipo de escaneos únicamente se deben de cumplir dos condiciones, que los paquetes puerto origen-puerto destino agrupados para este paquete no sean superior a uno.

Al igual que se ha realizado con el apartado anterior, expondremos un pequeño caso para poder observar toda la información. En la siguiente imagen podremos comprobar el contenido de nuestro "fichero.pcap" para los puertos 21, 53 y 49:

```
3 13.067483    192.168.5.15    192.168.5.20    TCP    54 62147 → 21 [ACK] Seq=1 Ack=1 Win=1024 Len=0
4 13.067549    192.168.5.15    192.168.5.20    TCP    54 62147 → 53 [ACK] Seq=1 Ack=1 Win=1024 Len=0
5 13.067573    192.168.5.15    192.168.5.20    TCP    54 62147 → 49 [ACK] Seq=1 Ack=1 Win=1024 Len=0
6 13.068069    192.168.5.20    192.168.5.15    TCP    60 21 → 62147 [RST] Seq=1 Win=0 Len=0
7 13.068082    192.168.5.20    192.168.5.15    TCP    60 53 → 62147 [RST] Seq=1 Win=0 Len=0
8 13.068087    192.168.5.20    192.168.5.15    TCP    60 49 → 62147 [RST] Seq=1 Win=0 Len=0
```

Se puede apreciar el envío de un único paquete a cada puerto, además este paquete posee la bandera ACK activada. Los resultados obtenidos por nuestro programa son los siguientes:

```
192.168.5.15 -> 192.168.5.20 :  62147 -> 21          TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  62147 -> 53          TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  62147 -> 49          TCP ACK SCAN
DETECTED 3 POSSIBLES SCANS!
```

Para el primer fichero que hemos creado, ACK_scan.pcap, obtenemos el siguiente output (lo hemos reducido al ser demasiado extenso):

```
192.168.5.15 -> 192.168.5.20 :  44745 -> 5901        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 4004        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 85          TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 2004        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 32776       TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 6567        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 7200        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 25734       TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 10243       TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 65389       TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 9103        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 49          TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 873         TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 8222        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 1105        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 1065        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 2103        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 1100        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 406         TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 19101       TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 444         TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 7103        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 17988       TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 2111        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 2048        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 1068        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 625         TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 5280        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 5678        TCP ACK SCAN
192.168.5.15 -> 192.168.5.20 :  44745 -> 49155       TCP ACK SCAN
```

c) Generación de tráfico para el tipo de escaneo TCP XMAS:

Comenzaremos igual que en los apartados anteriores, preparando la captura de tráfico a través de la herramienta tcpdump y el fichero donde almacenaremos los datos capturados denominado “XMAS_scan.pcap”. Una vez estamos capturando el tráfico procedemos a generarlo a través de la herramienta NMAP ejecutando el siguiente comando: `nmap -Pn -sX 192.168.5.20`, donde mediante “-sX” le indicamos que queremos que nos realice un escaneo del tipo XMAS, y mediante el comando “-Pn” le indicamos que no es necesario que realice un ping a la máquina que vamos a escanear. Una vez hemos finalizado la captura de tráfico, procedemos a realizar un análisis de los datos capturados a través de la herramienta wireshark:

```
3 13.063906    192.168.5.15    192.168.5.20    TCP    54 40392 → 25 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
4 13.063930    192.168.5.15    192.168.5.20    TCP    54 40392 → 111 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
5 13.063939    192.168.5.15    192.168.5.20    TCP    54 40392 → 1723 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
6 13.063947    192.168.5.15    192.168.5.20    TCP    54 40392 → 1025 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
7 13.063955    192.168.5.15    192.168.5.20    TCP    54 40392 → 993 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
8 13.063965    192.168.5.15    192.168.5.20    TCP    54 40392 → 1720 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
9 13.063973    192.168.5.15    192.168.5.20    TCP    54 40392 → 139 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
0 13.063982    192.168.5.15    192.168.5.20    TCP    54 40392 → 587 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
1 13.063990    192.168.5.15    192.168.5.20    TCP    54 40392 → 5900 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
2 13.063999    192.168.5.15    192.168.5.20    TCP    54 40392 → 445 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
3 13.064250    192.168.5.20    192.168.5.15    TCP    60 1723 → 40392 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
4 13.064257    192.168.5.20    192.168.5.15    TCP    60 1025 → 40392 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
5 13.064258    192.168.5.20    192.168.5.15    TCP    60 993 → 40392 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
6 13.064260    192.168.5.20    192.168.5.15    TCP    60 1720 → 40392 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
7 13.064262    192.168.5.20    192.168.5.15    TCP    60 587 → 40392 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
8 13.066617    192.168.5.15    192.168.5.20    TCP    54 40392 → 22 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
```

Como podemos observar, los paquetes enviados a los puertos a escanear tienen las banderas FIN, PUSH y URGENT activadas, por lo que sabemos que el escaneo que se está realizando sobre este puerto es un escaneo

del tipo XMAS al encontrar el paquete decorado como un árbol de Navidad. El resultado obtenido por nuestro programa para este fichero es el siguiente:

```
192.168.5.15 -> 192.168.5.20 :  40392 -> 25            TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 111           TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 1723          TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 1025          TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 993           TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 1720          TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 139           TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 587           TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 5900          TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 445           TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 22            TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 53            TCP XMAS SCAN
192.168.5.15 -> 192.168.5.20 :  40392 -> 21            TCP XMAS SCAN
```

En el que indica los puertos sobre los cuales se está realizando un escaneo del tipo TCP XMAS.

d) Generación de tráfico para el tipo de escaneo TCP FIN:

Para llevar a cabo la generación del tráfico para nuestro escaneo del tipo TCP FIN, en primer lugar, iniciaremos la captura de paquetes a través de la herramienta tcpdump, ejecutando el siguiente comando: `tcpdump -i eth1 -w FIN_scan.pcap`, donde nuestro fichero de salida, que analizaremos más tarde, se llama FIN_scan.pcap. Y, en segundo lugar, generaremos el tráfico que queremos capturar a través de la herramienta nmap ejecutando el siguiente comando: `nmap -Pn -sF 192.168.5.20`, en el que, como se ha visto anteriormente, mediante la opción "-Pn" evitamos que realice un ping a la máquina, y con la opción "-sF" le indicamos que tipo de escaneo queremos que realice (FIN scan). Ahora que ya tenemos capturado el tráfico que hemos generado, procedemos a comprobar su contenido mediante la herramienta wireshark:

```
 8 13.168866    192.168.5.15    192.168.5.20    TCP    54 62676 → 111 [FIN] Seq=1 Win=1024 Len=0
 9 13.168975    192.168.5.15    192.168.5.20    TCP    54 62676 → 1025 [FIN] Seq=1 Win=1024 Len=0
10 13.169030    192.168.5.15    192.168.5.20    TCP    54 62676 → 53 [FIN] Seq=1 Win=1024 Len=0
11 13.169243    192.168.5.15    192.168.5.20    TCP    54 62676 → 3389 [FIN] Seq=1 Win=1024 Len=0
12 13.169343    192.168.5.15    192.168.5.20    TCP    54 62676 → 8080 [FIN] Seq=1 Win=1024 Len=0
```

Como podemos observar en la captura de tráfico realizada, los paquetes TCP capturados y enviados a los puertos que van a ser escaneados, presentan únicamente la bandera FIN activada. Como se ha indicado anteriormente, la existencia de un sólo paquete y que, este paquete, únicamente posea la bandera FIN activada, es un claro indicador de que el tipo de escaneo que estamos realizando es FIN. Como veremos a continuación, el resultado devuelto por nuestro programa debería indicarnos que se ha realizado un escaneo de tipo TCP FIN:

```
192.168.5.15 -> 192.168.5.20 :  62676 -> 111           TCP FIN SCAN
192.168.5.15 -> 192.168.5.20 :  62676 -> 1025          TCP FIN SCAN
192.168.5.15 -> 192.168.5.20 :  62676 -> 53            TCP FIN SCAN
192.168.5.15 -> 192.168.5.20 :  62676 -> 3389          TCP FIN SCAN
192.168.5.15 -> 192.168.5.20 :  62676 -> 8080          TCP FIN SCAN
192.168.5.15 -> 192.168.5.20 :  62676 -> 5900          TCP FIN SCAN
192.168.5.15 -> 192.168.5.20 :  62676 -> 113           TCP FIN SCAN
```

e) Generación de tráfico para el tipo de escaneo TCP NULL:

Para realizar el análisis del tipo de escaneo TCP NULL, comenzaremos capturando el tráfico que circula a través de nuestra interfaz número uno. Para ello, emplearemos un comando similar al comando anterior, pero esta vez, el fichero de salida donde almacenaremos el tráfico capturado lo llamaremos "NULL_scan.pcap". Una vez que estamos capturando todo el tráfico que circula por esta interfaz, generaremos el tráfico a través de la herramienta nmap, mediante el comando: `nmap -Pn -sN 192.168.5.20`, donde como hemos visto anteriormente, la opción "-Pn" la empleamos para evitar que realice un ping a la máquina, y la opción "-sN" la empleamos para seleccionar el tipo de escaneo, en este caso, TCP NULL Scan. Ahora que hemos generado, capturado y almacenado el tráfico, podemos comprobar su contenido para asegurarnos que el output

generado por nuestro programa es realmente un escaneo del tipo TCP NULL Scan, para ello emplearemos la herramienta wireshark:

3 13.096113	192.168.5.15	192.168.5.20	TCP	54 38111 → 25 [<None>] Seq=1 Win=1024 Len=0
4 13.096137	192.168.5.15	192.168.5.20	TCP	54 38111 → 256 [<None>] Seq=1 Win=1024 Len=0
5 13.096146	192.168.5.15	192.168.5.20	TCP	54 38111 → 1025 [<None>] Seq=1 Win=1024 Len=0
6 13.096153	192.168.5.15	192.168.5.20	TCP	54 38111 → 53 [<None>] Seq=1 Win=1024 Len=0
7 13.096161	192.168.5.15	192.168.5.20	TCP	54 38111 → 21 [<None>] Seq=1 Win=1024 Len=0
8 13.096170	192.168.5.15	192.168.5.20	TCP	54 38111 → 111 [<None>] Seq=1 Win=1024 Len=0
9 13.096178	192.168.5.15	192.168.5.20	TCP	54 38111 → 135 [<None>] Seq=1 Win=1024 Len=0
0 13.096222	192.168.5.15	192.168.5.20	TCP	54 38111 → 8080 [<None>] Seq=1 Win=1024 Len=0
1 13.096235	192.168.5.15	192.168.5.20	TCP	54 38111 → 199 [<None>] Seq=1 Win=1024 Len=0
2 13.096287	192.168.5.15	192.168.5.20	TCP	54 38111 → 22 [<None>] Seq=1 Win=1024 Len=0

Como se puede observar en la captura anterior, la peculiaridad de este tipo de escaneo es que envía paquetes en los que no se encuentra activada ninguna de las banderas, por lo que nuestro programa, al recibir el fichero donde se encuentra toda la información sobre el tráfico capturado, debería de indicarnos que se está realizando un escaneo del tipo TCP NULL. El output recibido por el programa es el siguiente:

```
192.168.5.15 -> 192.168.5.20 :  38111 -> 25            TCP NULL SCAN
192.168.5.15 -> 192.168.5.20 :  38111 -> 256           TCP NULL SCAN
192.168.5.15 -> 192.168.5.20 :  38111 -> 1025          TCP NULL SCAN
192.168.5.15 -> 192.168.5.20 :  38111 -> 53            TCP NULL SCAN
192.168.5.15 -> 192.168.5.20 :  38111 -> 21            TCP NULL SCAN
192.168.5.15 -> 192.168.5.20 :  38111 -> 111           TCP NULL SCAN
192.168.5.15 -> 192.168.5.20 :  38111 -> 135           TCP NULL SCAN
```

f) Generación de tráfico para el tipo de escaneo UDP:

Finalmente, para realizar el escaneo de puertos del tipo UDP, comenzaremos preparando la captura de tráfico con la herramienta tcpdump, de forma similar a como hemos hecho en los apartados anteriores. Pero esta vez, el fichero donde capturaremos toda la información lo llamaremos “udp_portScan.pcap”. Procederemos ahora a generar el tráfico que circule por nuestra interfaz número uno mediante la herramienta nmap; para ello ejecutaremos el comando: `nmap -Pn -sU 192.168.5.20`, donde la opción “-Pn” indica que no realiza la fase de ping a la máquina que se encuentra en la ip pasada por parámetro y “-sU” le indica que se debe de realizar un escaneo tipo UDP. Una vez hemos finalizado este tipo de escaneo, que llevará más tiempo que un escaneo basado en paquetes TCP, finalizamos la captura del tráfico y procedemos a abrir el fichero donde hemos capturado toda la información mediante la herramienta wireshark:

1 0.000000	PcsCompu_5e:19:82	Broadcast	ARP	42 Who has 192.168.5.20? Tell 192.168.5.15
2 0.000884	PcsCompu_3c:d5:83	PcsCompu_5e:19:82	ARP	60 192.168.5.20 is at 08:00:27:3c:d5:83
3 13.175945	192.168.5.15	192.168.5.20	UDP	42 62863 → 464 Len=0
4 13.176022	192.168.5.15	192.168.5.20	UDP	42 62863 → 20326 Len=0
5 13.176056	192.168.5.15	192.168.5.20	UDP	42 62863 → 21514 Len=0
6 13.176094	192.168.5.15	192.168.5.20	UDP	42 62863 → 25709 Len=0
7 13.176486	192.168.5.15	192.168.5.20	UDP	42 62863 → 2345 Len=0
8 13.176555	192.168.5.15	192.168.5.20	UDP	42 62863 → 55043 Len=0
9 13.176622	192.168.5.15	192.168.5.20	UDP	42 62863 → 58075 Len=0

Como podemos observar, el protocolo de envío de los paquetes que se encuentran en este fichero es UDP. Ahora procedemos a comparar la información dada por nuestro programa, debe de indicar que el tipo de escaneo realizado es UDP:

```
192.168.5.15 -> 192.168.5.20 :  62863 -> 464           UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 20326         UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 21514         UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 25709         UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 2345          UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 55043         UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 58075         UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 17946         UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 18666         UDP SCAN
192.168.5.15 -> 192.168.5.20 :  62863 -> 42557         UDP SCAN
```

4. Técnicas de evasión:

A continuación, expondré algunas técnicas con las que es posible no ser detectado por el programa y llevaremos a cabo la corrección de estas técnicas.

- **Mismo puerto de origen para todos los escaneos:** en la implementación del programa no se tiene en cuenta que los distintos tipos de escaneos fueran todos desde el mismo puerto origen, por defecto NMAP empela distintos puertos cada vez que se realiza un escaneo. Sin embargo, es posible establecer un mismo puerto origen mediante el siguiente comando:

```
nmap -Pn -sA -g 58996 192.168.5.20
```

Con la opción "-g" le indicamos el puerto desde el que queremos que realice el escaneo. Por lo que, si encadenamos varios escaneos, nuestra implementación entenderá que se está realizando una "falsa comunicación". Para ello procederemos a controlar que tipo de paquete se envía en cada momento comprobando las banderas del paquete TCP enviado. Teniendo como base lo que sabemos de cada tipo de escaneo podemos controlar los paquetes que debe de haber en cada momento en la conexión.
Para la solución a este problema realizamos un ajuste en la función encargada de analizar los paquetes para que pueda diferenciar cuando se trata de una comunicación entre dos servicios y un escaneo. Para generar las pruebas y asegurar que se ha solucionado el problema, comenzaremos preparando la captura del tráfico que hemos generado a través de la herramienta tcpdump tal y como hemos hecho hasta ahora. Ejecutaremos el siguiente comando:

```
tcpdump -i eth1 -w SCAN_FINSYN.pcap
```

Este comando nos capturará todo el tráfico que circule a través de la interfaz de red número uno y lo guardará en un fichero denominado "SCAN_FINSYN.pcap". Una vez estamos capturando el tráfico que circula a través de la interfaz número 1, procederemos a generarlo a través de la herramienta NMAP. Para ello ejecutaremos el siguiente comando:

```
nmap -Pn -sS -g 58996 192.168.5.20 -p 23,49
nmap -Pn -sF -g 58996 192.168.5.20 -p 23,49
```

Este comando realizará dos escaneos diferentes sobre los puertos 23,49. El primero de los escaneos será un escaneo del tipo TCP Syn Scan, y el segundo tipo de escaneo se trata de un escaneo del tipo TCP Fin Scan. Si abrimos el fichero en el que hemos capturado toda la información mediante la herramienta wireshark, encontramos lo siguiente:

1 0.000000	PcsCompu_5e:19:82	Broadcast	ARP	42 Who has 192.168.5.20? Tell 192.168.5.15
2 0.000300	PcsCompu_3c:d5:83	PcsCompu_5e:19:82	ARP	60 192.168.5.20 is at 08:00:27:3c:d5:83
3 13.071644	192.168.5.15	192.168.5.20	TCP	58 58996 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4 13.071721	192.168.5.15	192.168.5.20	TCP	58 58996 → 49 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5 13.072534	192.168.5.20	192.168.5.15	TCP	60 23 → 58996 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
6 13.072575	192.168.5.15	192.168.5.20	TCP	54 58996 → 23 [RST] Seq=1 Win=0 Len=0
7 13.072635	192.168.5.20	192.168.5.15	TCP	60 49 → 58996 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
8 13.280289	PcsCompu_5e:19:82	Broadcast	ARP	42 Who has 192.168.5.20? Tell 192.168.5.15
9 13.281141	PcsCompu_3c:d5:83	PcsCompu_5e:19:82	ARP	60 192.168.5.20 is at 08:00:27:3c:d5:83
10 26.395134	192.168.5.15	192.168.5.20	TCP	54 [TCP Retransmission] 58996 → 23 [FIN] Seq=3593390816 Win=1024 Len=0
11 26.395191	192.168.5.15	192.168.5.20	TCP	54 [TCP Retransmission] 58996 → 49 [FIN] Seq=3593390816 Win=1024 Len=0
12 26.395815	192.168.5.20	192.168.5.15	TCP	60 49 → 58996 [RST, ACK] Seq=1 Ack=3593390817 Win=0 Len=0
13 27.499677	192.168.5.15	192.168.5.20	TCP	54 [TCP Retransmission] 58996 → 23 [FIN] Seq=3593456353 Win=1024 Len=0
14 31.396688	PcsCompu_3c:d5:83	PcsCompu_5e:19:82	ARP	60 Who has 192.168.5.15? Tell 192.168.5.20
15 31.396708	PcsCompu_5e:19:82	PcsCompu_3c:d5:83	ARP	42 192.168.5.15 is at 08:00:27:5e:19:82

Como podemos observar, el puerto de origen es el mismo para ambos tipos de escaneo, y cada escaneo viene distinguido por las banderas que tienen activadas los paquetes TCP. En un inicio, no eran detectados ninguno de estos escaneos, al considerar que se trataba de una comunicación entre los puertos. Controlando este tipo de información, conseguimos que se traten y se detecten como posibles escaneos de puertos. El output generado por el programa es el siguiente:

```
192.168.5.15 -> 192.168.5.20 :  58996 -> 23            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  58996 -> 49            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  58996 -> 23            TCP FIN SCAN
192.168.5.15 -> 192.168.5.20 :  58996 -> 49            TCP FIN SCAN
DETECTED 4 POSSIBLES SCANS!
```

- **Añadiendo banderas activadas a cada tipo de escaneo:** una de las formas más comunes de evitar la detección de un escaneo de puertos, es el uso de la opción que nos da NMAP para seleccionar que banderas del paquete TCP queremos que estén activas. Como nuestro programa busca similitudes en las banderas activadas en los paquetes TCP con los tipos de escaneo que hemos implementado, al modificar estas banderas el programa pasa por alto estos paquetes y no nos alerta de un posible escaneo de puertos. Para poder usar esta opción emplearemos el siguiente comando en NMAP:

```
nmap -Pn -sS --scanflags SYNFIN 192.168.5.20 -p 23,53,59
```

En donde, mediante la opción "--scanflags SYNFIN", le indicamos que las banderas que queremos que estén activadas sean la bandera de SYN y la bandera de FIN. Además, mediante el comando anterior también le indicamos queremos que sean escaneados, en este caso, 23, 53 y 59. Para poder comprobar cómo se puede evitar la detección por parte del programa implementado, comenzaremos capturando el tráfico que circula a través de nuestra interfaz número uno, y generaremos dicho tráfico mediante los comandos:

```
nmap -Pn -sS 192.168.5.20 -p 23,53,59
nmap -Pn -sS --scanflags SYNFIN 192.168.5.20 -p 23,53,59
```

Estos comandos realizarán dos tipos de escaneo; el primero de ellos es un escaneo TCP SYN normal, los paquetes TCP poseen la bandera SYN activada y el segundo de estos comandos es un escaneo TCP SYN en el que se encuentran las banderas SYN y FIN activadas. Si abrimos el fichero donde hemos capturado este tráfico mediante la herramienta wireshark encontramos lo siguiente:

```
1 0.000000      PcsCompu_5e:19:82   Broadcast           ARP      42 Who has 192.168.5.20? Tell 192.168.5.15
2 0.000510      PcsCompu_3c:d5:83   PcsCompu_5e:19:82   ARP      60 192.168.5.20 is at 08:00:27:3c:d5:83
3 13.265811     192.168.5.15        192.168.5.20        TCP      58 39070 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4 13.265889     192.168.5.15        192.168.5.20        TCP      58 39070 → 53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5 13.265924     192.168.5.15        192.168.5.20        TCP      58 39070 → 59 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6 13.266740     192.168.5.20        192.168.5.15        TCP      60 23 → 39070 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
7 13.266774     192.168.5.15        192.168.5.20        TCP      54 39070 → 23 [RST] Seq=1 Win=0 Len=0
8 13.266820     192.168.5.20        192.168.5.15        TCP      60 53 → 39070 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
9 13.266832     192.168.5.15        192.168.5.20        TCP      54 39070 → 53 [RST] Seq=1 Win=0 Len=0
0 13.266863     192.168.5.20        192.168.5.15        TCP      60 59 → 39070 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1 18.265785     PcsCompu_3c:d5:83   PcsCompu_5e:19:82   ARP      60 Who has 192.168.5.15? Tell 192.168.5.20
2 18.265799     PcsCompu_5e:19:82   PcsCompu_3c:d5:83   ARP      42 192.168.5.15 is at 08:00:27:5e:19:82
3 36.475981     PcsCompu_5e:19:82   Broadcast           ARP      42 Who has 192.168.5.20? Tell 192.168.5.15
4 36.476427     PcsCompu_3c:d5:83   PcsCompu_5e:19:82   ARP      60 192.168.5.20 is at 08:00:27:3c:d5:83
5 49.544350     192.168.5.15        192.168.5.20        TCP      58 47363 → 53 [FIN, SYN] Seq=0 Win=1024 Len=0 MSS=1460
6 49.544429     192.168.5.15        192.168.5.20        TCP      58 47363 → 23 [FIN, SYN] Seq=0 Win=1024 Len=0 MSS=1460
7 49.544472     192.168.5.15        192.168.5.20        TCP      58 47363 → 59 [FIN, SYN] Seq=0 Win=1024 Len=0 MSS=1460
8 49.545466     192.168.5.20        192.168.5.15        TCP      60 53 → 47363 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
9 49.545499     192.168.5.15        192.168.5.20        TCP      54 47363 → 53 [RST] Seq=1 Win=0 Len=0
0 49.545542     192.168.5.20        192.168.5.15        TCP      60 23 → 47363 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
1 49.545552     192.168.5.15        192.168.5.20        TCP      54 47363 → 23 [RST] Seq=1 Win=0 Len=0
2 49.545578     192.168.5.20        192.168.5.15        TCP      60 [TCP ACKed unseen segment] 59 → 47363 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
```

En un primer lugar encontramos el escaneo TCP SYN son modificaciones, a continuación, encontramos el escaneo TCP SYN con las banderas SYN y FIN activadas. El output generado el programa es el siguiente:

```
192.168.5.15 -> 192.168.5.20 :  39070 -> 23            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  39070 -> 53            TCP SYN SCAN
192.168.5.15 -> 192.168.5.20 :  39070 -> 59            TCP SYN SCAN
DETECTED 3 POSSIBLES SCANS!
```

Como podemos observar, únicamente detecta el primer tipo de escaneo dejando de lado el segundo escaneo realizado con las banderas SYN y FIN activadas.

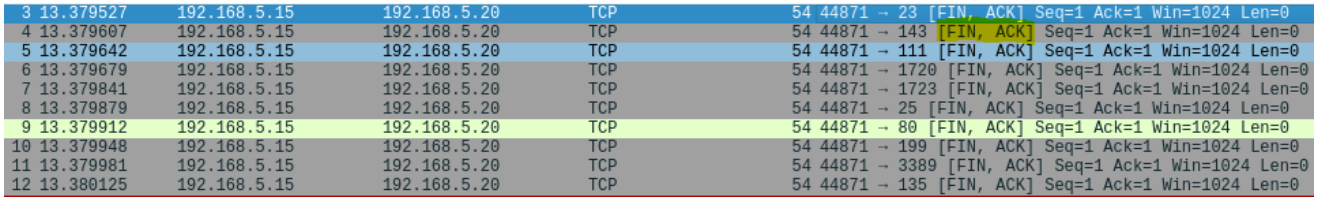
- **Maimon Scan:** ahora trataremos de detectar un tipo de escaneo algo más reciente basado en la activación de algunas banderas. Se trata del tipo de escaneo: "TCP Maimon Scan". Como se ha indicado anteriormente, en

este tipo de escaneo, se envía un paquete TCP con las banderas FIN/SYN activadas, con lo que añadiremos un nuevo caso a nuestro detector para que sea capaz de capturar aquellos escaneos que sean del tipo Maimon.

Comenzaremos preparando la captura de tráfico mediante la herramienta tcpdump. Una vez hemos preparado la captura de tráfico generaremos el tráfico que queremos capturar mediante la herramienta nmap a través del comando:

```
Namp -Pn -sM 192.168.5.20
```

Este comando nos permite realizar un escaneo del tipo Maimon, una vez ha finalizado el escaneo procedemos a comprobar el contenido del fichero a través de la herramienta wireshark:



Como se puede observar en los paquetes TCP enviados, las banderas que se encuentran activadas son ACK y FIN, lo que nos indica que tipo de escaneo se está realizando.

El programa implementado, sin realizar las pertinentes comprobaciones para este tipo de escaneo, no detecta estos paquetes como posible escaneo como nos muestra la siguiente caputra:

```
_____________ PORT SCAN DETECTOR _____________
NO SCANS DETECTED! YOU ARE SAFE!
```

Una vez hemos realizado el ejuste pertinente en el código para que este tipo de escaneo sea detectado obtenemos el siguiente output, en el que como puede observar, se indica que se está realizando un escaneo del tipo Maimon.

```
_____________ PORT SCAN DETECTOR _____________
DETECTED 1000 POSSIBLES SCANS!
DETECTED 1000 TCP MAIMON SCAN
```

Si queremos obtener más detalles sobre el escaneo, añadiendo la opción “-vv” al comando nos devuelve el siguiente output:

```
_____________ PORT SCAN DETECTOR _____________
192.168.5.15 -> 192.168.5.20 :  44871 -> 23      TCP MAIMON SCAN
192.168.5.15 -> 192.168.5.20 :  44871 -> 143     TCP MAIMON SCAN
192.168.5.15 -> 192.168.5.20 :  44871 -> 111     TCP MAIMON SCAN
192.168.5.15 -> 192.168.5.20 :  44871 -> 1720    TCP MAIMON SCAN
192.168.5.15 -> 192.168.5.20 :  44871 -> 1723    TCP MAIMON SCAN
192.168.5.15 -> 192.168.5.20 :  44871 -> 25      TCP MAIMON SCAN
192.168.5.15 -> 192.168.5.20 :  44871 -> 80      TCP MAIMON SCAN
192.168.5.15 -> 192.168.5.20 :  44871 -> 199     TCP MAIMON SCAN
```