

ACTIVIDADES MODULARES NETWORK

1ª Actividad escaneo de red

El contenido de esta actividad se ha creado utilizando Kali Linux, una distribución de Linux especializada en seguridad informática, pruebas de penetración y auditorías. Las herramientas y comandos descritos a lo largo de este documento están basados en la funcionalidad proporcionada por Kali Linux.

Usando nmap:

- Explicación:

Aclaraciones:

Durante la realización de este proyecto, utilicé la herramienta Nmap para llevar a cabo un escaneo en mi red local. Este escaneo tuvo como objetivo identificar dispositivos y analizar la configuración de servicios y puertos.

También implementamos Wireshark para realizar análisis de tráfico de red.

Wireshark nos permitió examinar en detalle los paquetes de datos que circulan en nuestra red, facilitando la comprensión de patrones de comunicación, protocolos utilizados y cualquier otra información relevante.

Pasos:

Para poder ejecutar este script necesitamos acceder a modo super usuario(sudo su).

Escribiendo en la terminal ifconfig podemos ver la configuración de red que tenemos. En el apartado eth0 podemos distinguir nuestra ip, la máscara de red y el broadcast.

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.138 netmask 255.255.255.0 broadcast 192.168.1.2
```

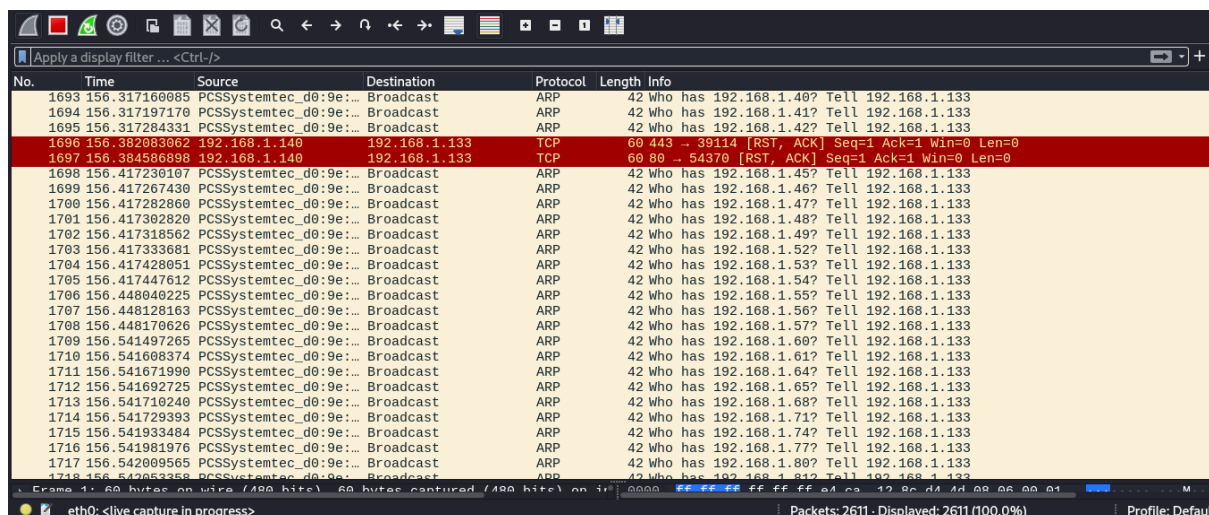
Con esto descubrimos que la dirección de nuestra red es 192.168.1.0 con un máscara /24 = 255.255.255.0, lo que significa que tenemos una ip de clase C.

Para hacer el escaneo de nuestra red local con nmap utilizaremos este comando: nmap -sn 'dirección ip', Nmap realiza el envío de solicitudes de eco ICMP (ping) a todas las direcciones IP dentro del rango especificado. A través de este proceso, identifica las direcciones IP que responden positivamente a las solicitudes de ping. El resultado final de este escaneo se presenta como una lista que refleja las

direcciones IP activas en la red local, proporcionando así una visión clara de la conectividad y disponibilidad de dispositivos en la red.

```
rodrix@maquinon ~/Documents/Informáticos nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-12 17:25 CET
Nmap scan report for 192.168.1.1
Host is up (0.0011s latency).
Nmap scan report for 192.168.1.128
Host is up (0.0036s latency).
Nmap scan report for 192.168.1.131
Host is up (0.015s latency).
Nmap scan report for 192.168.1.132
Host is up (0.0064s latency).
Nmap scan report for 192.168.1.133
Host is up (0.0067s latency).
Nmap scan report for 192.168.1.134
Host is up (0.0066s latency).
Nmap scan report for 192.168.1.135
Host is up (0.041s latency).
Nmap scan report for 192.168.1.139
Host is up (0.0073s latency).
Nmap scan report for 192.168.1.140
Host is up (0.071s latency).
Nmap done: 256 IP addresses (9 hosts up) scanned in 3.11 seconds
```

Con wireshark podemos analizar el tráfico de red que estamos generando cuando utilizamos este comando



Si agregamos un par de modificaciones al script podemos conseguir que nos genere un archivo html con estilo en donde esten todas las ips activas de nuestra red local:

```
#!/bin/bash

# Especifica el rango de IP de tu red (ajusta según tu red)
red="192.168.128.0/24"
```

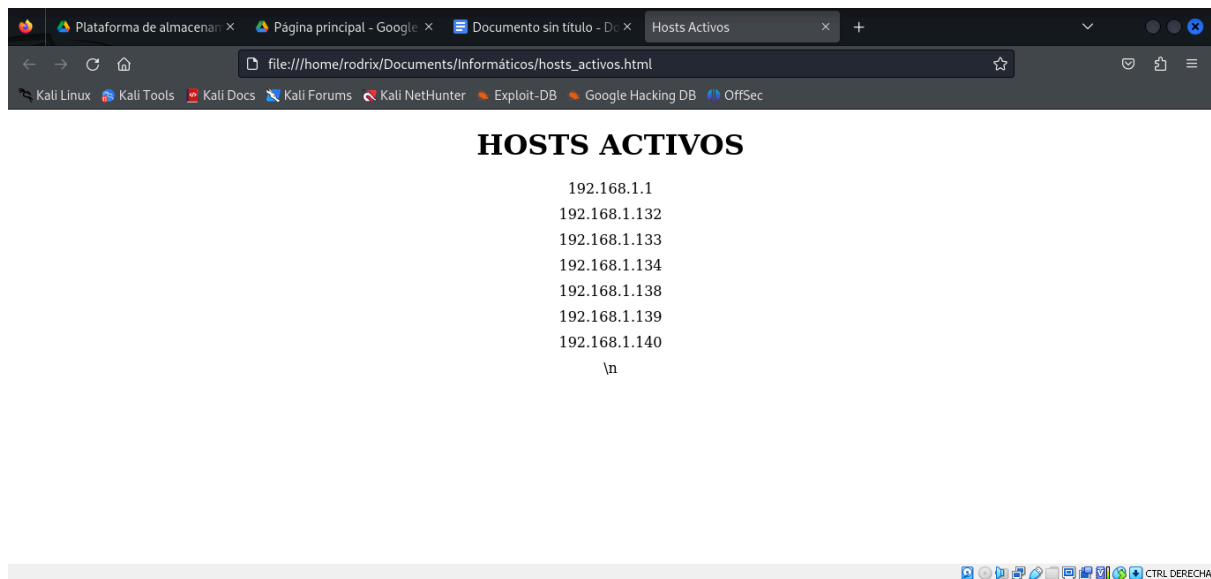
```
# Realiza un escaneo de todos los hosts activos en la red y guarda los
resultados en un archivo
nmap -sn $red -oG resultados.txt

# Crea un archivo HTML con la estructura básica y un título
echo -e "<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n<meta
charset=\"UTF-8\">\n<meta name=\"viewport\"
content=\"width=device-width, initial-scale=1.0\">\n<title>Hosts
Activos</title>\n<style>\nbody { text-align: center; }\np { margin:
10px; }\nh1 { margin-top: 20px; }\n</style>\n</head>\n<body>\n<h1>HOSTS
ACTIVOS</h1>" > script_host_activos.html

# Extrae las direcciones IP activas del archivo de resultados y las
agrega al archivo HTML
awk '/Up$/ {print "<p>" $2 "</p>"}' resultados.txt >>
script_host_activos.html

# Cierra la estructura del archivo HTML
echo "</body>\n</html>" >> script_host_activos.html

# Abre el archivo HTML en el navegador predeterminado
xdg-open script_host_activos.html
```



Usando bash

Además de utilizar herramientas especializadas como nmap para realizar escaneos de red, también podemos aprovechar el poder del lenguaje de scripting Bash para crear scripts personalizados que lleven a cabo escaneos de red básicos.

```
#!/bin/bash

# Escaneo de red utilizando ping en un rango de direcciones IP

if [ "$#" -ne 1 ]; then
    echo "Uso: $0 <rango_de_IP>"
    echo "Ejemplo: $0 192.168.1.1-254"
    exit 1
fi

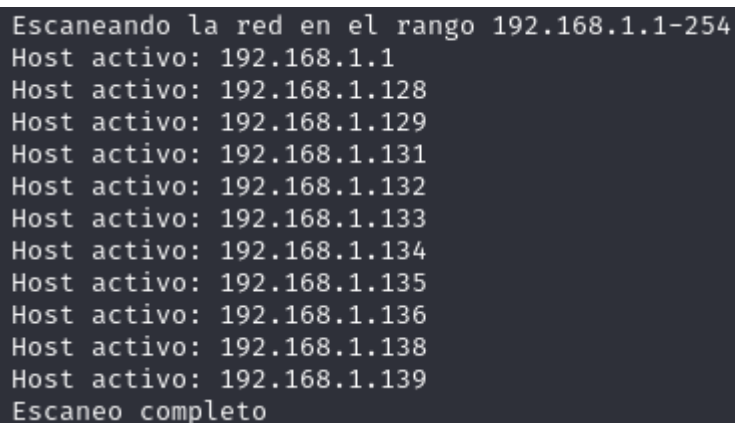
range="$1"

echo "Escaneando la red en el rango $range"

for ip in $(seq -f "192.168.1.%g" 1 254); do
    ping -c 1 -w 1 $ip > /dev/null 2>&1
    if [ $? -eq 0 ]; then
        echo "Host activo: $ip"
    fi
done

echo "Escaneo completo"
```

Este script utiliza un bucle for para iterar a través de direcciones IP en el rango especificado y utiliza el comando ping para determinar si cada host está activo. La salida del script mostrará los hosts activos en la red.

A terminal window with a dark background showing the output of the network scan script. The output is as follows:

```
Escaneando la red en el rango 192.168.1.1-254
Host activo: 192.168.1.1
Host activo: 192.168.1.128
Host activo: 192.168.1.129
Host activo: 192.168.1.131
Host activo: 192.168.1.132
Host activo: 192.168.1.133
Host activo: 192.168.1.134
Host activo: 192.168.1.135
Host activo: 192.168.1.136
Host activo: 192.168.1.138
Host activo: 192.168.1.139
Escaneo completo
```

El método de escaneo de red con ping en un script Bash tiende a ser más lento que herramientas especializadas como nmap por varias razones. El enfoque secuencial del script, la espera entre solicitudes, y la limitación de respuestas de algunos dispositivos contribuyen a la lentitud. En contraste, herramientas como nmap

realizan escaneos concurrentes y utilizan técnicas avanzadas para obtener información detallada de la red, lo que las hace más eficientes, especialmente en entornos más grandes o complejos. Aunque el script Bash es útil para tareas básicas, nmap es preferible para escaneos rápidos y detallados.

2ª Actividad calculadora de direcciones ip en binario

Para esta actividad, la cual consiste en crear un script que calcule todos los posibles hosts disponibles de una red con una máscara específica, me he ayudado de chatgpt.

Con una serie de prompts conseguí que me generase un script parecido al resultado que quería sin embargo tenía un par de problemas los cuales tuve que solucionar tocando yo el código.

[illegible]

```

        bit12,
        bit13,
        bit14,
        bit15,
        bit16
    )
    ip_addresses.add(ip_address)

return ip_addresses

# Patrón base para la dirección IP en binario con bits variables
ip_pattern = "11000000.{}{}{}{}{}{}.00001{}0{}.00010111"

# Generar todas las posibles combinaciones de direcciones IP en binario
all_ip_addresses = generate_all_binary_combinations(ip_pattern)

# Imprimir las direcciones IP en binario
for ip_address in all_ip_addresses:
    print(ip_address)

# Nombre del archivo de salida
output_file_name = "IP_address.txt"

# Escribir las direcciones IP en el archivo de salida
with open(output_file_name, "w") as output_file:
    for ip_address in all_ip_addresses:
        output_file.write(ip_address + "\n")

```

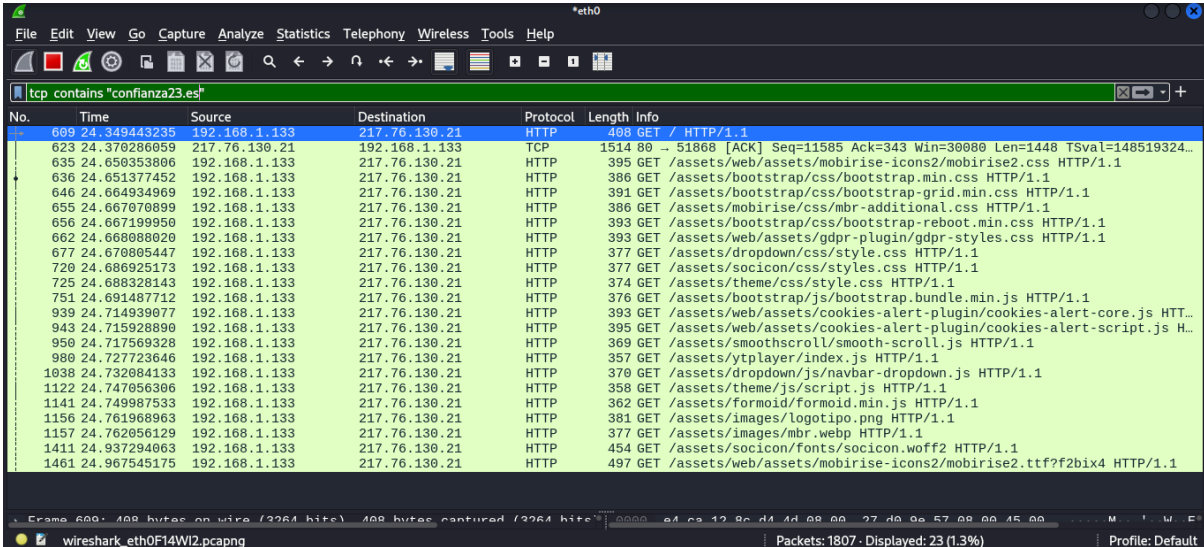
- Define una función llamada `generate_all_binary_combinations` que toma un patrón de dirección IP como argumento.
- Crea un conjunto llamado `ip_addresses` para almacenar direcciones IP únicas.
- Utiliza bucles anidados para generar todas las combinaciones posibles de bits en la dirección IP, reemplazando los bits variables en el patrón.
- Devuelve el conjunto de direcciones IP generadas por la función.
- Define un patrón de dirección IP en binario con bits variables representados por `{}`.
- Llama a la función con el patrón de dirección IP y almacena el resultado en `all_ip_addresses`.
- Define el nombre del archivo de salida como `"resultados.txt"`.

- Abre el archivo en modo de escritura y escribe cada dirección IP única en el archivo, una por línea.
- Imprime un mensaje indicando que las direcciones IP han sido escritas en el archivo.

3ª Actividad sniffer captura de tráfico de la red

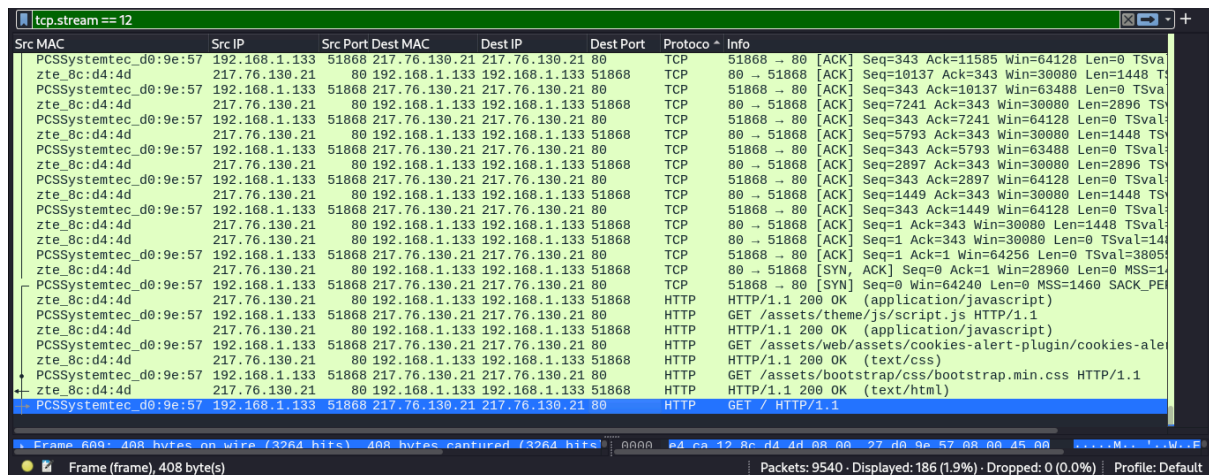
Para realizar esta actividad he utilizado wireshark. Este módulo tiene como objetivo llevar a cabo la captura de tráfico de una red local en los equipos participantes, simulando el comportamiento normal de un usuario durante un período específico. La finalidad principal de esta actividad es capturar y analizar el patrón de tráfico típicamente asociado con las operaciones habituales de los usuarios en la red. La simulación se centra en reproducir las acciones cotidianas de un usuario convencional dentro de un intervalo de tiempo definido. Esto incluirá actividades como navegación web, envío y recepción de correos electrónicos, acceso a servicios en línea y otras tareas comunes realizadas en el entorno de red. La intención es establecer un "perfil normal" del tráfico de red durante estas operaciones regulares.

Primero iniciamos en wireshark, luego hacemos una búsqueda en google, en este caso [www.confianza 23.es](http://www.confianza23.es). Filtramos en wireshark tcp contains “conifanza23.es” para ver el tráfico generado por la búsqueda;



No.	Time	Source	Destination	Protocol	Length	Info
609	24.349443235	192.168.1.133	217.76.130.21	HTTP	408	GET / HTTP/1.1
623	24.370286059	217.76.130.21	192.168.1.133	TCP	1514	80 → 51808 [ACK] Seq=11585 Ack=343 Win=30080 Len=1448 TSval=148519324...
635	24.650353806	192.168.1.133	217.76.130.21	HTTP	395	GET /assets/web/assets/mobirise-icons2/mobirise2.css HTTP/1.1
636	24.651377452	192.168.1.133	217.76.130.21	HTTP	386	GET /assets/bootstrap/css/bootstrap.min.css HTTP/1.1
646	24.664934969	192.168.1.133	217.76.130.21	HTTP	391	GET /assets/bootstrap/css/bootstrap-grid.min.css HTTP/1.1
655	24.667070899	192.168.1.133	217.76.130.21	HTTP	386	GET /assets/mobirise/css/mbr-additional.css HTTP/1.1
656	24.667199950	192.168.1.133	217.76.130.21	HTTP	393	GET /assets/bootstrap/css/bootstrap-reboot.min.css HTTP/1.1
662	24.668088020	192.168.1.133	217.76.130.21	HTTP	393	GET /assets/web/assets/gdpr-plugin/gdpr-styles.css HTTP/1.1
677	24.670805447	192.168.1.133	217.76.130.21	HTTP	377	GET /assets/dropdown/css/style.css HTTP/1.1
720	24.686925173	192.168.1.133	217.76.130.21	HTTP	377	GET /assets/socicon/css/stylesheet.css HTTP/1.1
725	24.688328143	192.168.1.133	217.76.130.21	HTTP	374	GET /assets/theme/css/style.css HTTP/1.1
751	24.691487712	192.168.1.133	217.76.130.21	HTTP	376	GET /assets/bootstrap/js/bootstrap.bundle.min.js HTTP/1.1
939	24.714939077	192.168.1.133	217.76.130.21	HTTP	393	GET /assets/web/assets/cookies-alert-plugin/cookies-alert-core.js HTTP/1.1
943	24.715928890	192.168.1.133	217.76.130.21	HTTP	395	GET /assets/web/assets/cookies-alert-plugin/cookies-alert-script.js HTTP/1.1
959	24.717569328	192.168.1.133	217.76.130.21	HTTP	369	GET /assets/smoothscroll/smooth-scroll.js HTTP/1.1
980	24.727723646	192.168.1.133	217.76.130.21	HTTP	357	GET /assets/ytplayer/index.js HTTP/1.1
1038	24.732084133	192.168.1.133	217.76.130.21	HTTP	370	GET /assets/dropdown/js/navbar-dropdown.js HTTP/1.1
1122	24.747056306	192.168.1.133	217.76.130.21	HTTP	358	GET /assets/theme/js/script.js HTTP/1.1
1141	24.749987533	192.168.1.133	217.76.130.21	HTTP	382	GET /assets/formoid/formoid.min.js HTTP/1.1
1156	24.761968963	192.168.1.133	217.76.130.21	HTTP	361	GET /assets/images/logotipo.png HTTP/1.1
1157	24.762056129	192.168.1.133	217.76.130.21	HTTP	377	GET /assets/images/mbr.webp HTTP/1.1
1411	24.937294063	192.168.1.133	217.76.130.21	HTTP	454	GET /assets/socicon/fonts/socicon.woff2 HTTP/1.1
1461	24.967545175	192.168.1.133	217.76.130.21	HTTP	497	GET /assets/web/assets/mobirise-icons2/mobirise2.ttf?fb2bix4 HTTP/1.1

Y finalmente hacemos la modificaciones especificadas en la actividad;



The screenshot shows a Wireshark capture of network traffic. The top pane displays a list of packets, and the bottom pane shows the details of the selected packet (Frame 408). The traffic is between PCSSystemtec (192.168.1.133) and zte_8c (217.76.130.21). The traffic includes TCP connections and HTTP requests/responses.

Src MAC	Src IP	Src Port	Dest MAC	Dest IP	Dest Port	Protocol	Info
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [ACK] Seq=343 Ack=11585 Win=64128 Len=0 TSval=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	TCP	80 → 51868 [ACK] Seq=10137 Ack=343 Win=30080 Len=0 TSval=...
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [ACK] Seq=343 Ack=10137 Win=63488 Len=0 TSval=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	TCP	80 → 51868 [ACK] Seq=7241 Ack=343 Win=30080 Len=2896 TSval=...
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [ACK] Seq=343 Ack=7241 Win=64128 Len=0 TSval=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	TCP	80 → 51868 [ACK] Seq=5793 Ack=343 Win=30080 Len=1448 TSval=...
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [ACK] Seq=343 Ack=5793 Win=63488 Len=0 TSval=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	TCP	80 → 51868 [ACK] Seq=2897 Ack=343 Win=30080 Len=2896 TSval=...
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [ACK] Seq=343 Ack=2897 Win=64128 Len=0 TSval=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	TCP	80 → 51868 [ACK] Seq=1449 Ack=343 Win=30080 Len=1448 TSval=...
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [ACK] Seq=343 Ack=1449 Win=64128 Len=0 TSval=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	TCP	80 → 51868 [ACK] Seq=1 Ack=343 Win=30080 Len=1448 TSval=...
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [ACK] Seq=1 Ack=343 Win=30080 Len=0 TSval=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	TCP	80 → 51868 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460...
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	TCP	51868 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=...
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	HTTP	HTTP/1.1 200 OK (application/javascript)
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	HTTP	GET /assets/theme/js/script.js HTTP/1.1
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	HTTP	HTTP/1.1 200 OK (application/javascript)
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	HTTP	GET /assets/web/assets/cookies-alert-plugin/cookies-alert.js HTTP/1.1
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	HTTP	HTTP/1.1 200 OK (text/css)
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	HTTP	GET /assets/bootstrap/css/bootstrap.min.css HTTP/1.1
zte_8c:d4:4d	217.76.130.21	80	PCSSystemtec_d0:9e:57	192.168.1.133	51868	HTTP	HTTP/1.1 200 OK (text/html)
PCSSystemtec_d0:9e:57	192.168.1.133	51868	zte_8c:d4:4d	217.76.130.21	80	HTTP	GET / HTTP/1.1

La finalidad principal es diferenciar este tráfico normal del tipo de tráfico que podría indicar la realización de operaciones relacionadas con el hacking ético o la telemetría de datos.

4ª Actividad Network Scanner, escaneador de red

El objetivo de la actividad es realizar un escaneo de red de una determinada subred (en este caso, 192.168.1.0/24) y generar un informe que incluya la lista de equipos activos a nivel de IP mediante ping ICMP, así como un listado completo de todos los puertos TCP y UDP en esos equipos con su estado (abierto, cerrado, filtrado).

Enfoque:

Escaneo de Direcciones IP:

- Utilicé la herramienta nmap para realizar un escaneo de direcciones IP de la subred especificada. El resultado incluye una lista de equipos activos que responden al ping ICMP.

Escaneo de Puertos TCP y UDP:

- Nuevamente, empleé nmap para escanear todos los puertos TCP y UDP de los equipos identificados como activos. Esto proporciona un listado completo de los puertos y su estado.

Generación de Resultados y Archivo ASCII:

- He utilizado un script en bash para procesar los resultados y mostrarlos en la terminal.
- El resultado se presenta en la terminal mostrando los equipos activos y la información de los puertos.
- Además, redirigí la salida del script a un archivo ASCII llamado resultado.txt para tener una versión persistente y fácil de compartir del informe.


```
#!/bin/bash

equipos_activos=$(nmap -sn 192.168.1.0/24 | grep "Nmap scan report" | awk '{print $5}')

echo "Equipos Activos:"

echo "$equipos_activos"

resultado_puertos=$(nmap -p- -sS -sU --open 192.168.1.0/24)

echo -e "\nListado de Todos los Puertos TCP y UDP:"

echo "$resultado_puertos" | grep -E '[0-9]+' | awk '{print "Puerto \"$1\": \"$2\"}'

echo -e "\nGuardando la información en el archivo 'resultado.txt'"

{

echo "Equipos Activos:"

echo "$equipos_activos"

echo -e "\nListado de Todos los Puertos TCP y UDP:"

echo "$resultado_puertos" | grep -E '[0-9]+' | awk '{print "Puerto \"$1\": \"$2\"}'

} > resultado.txt
```

- El script se ejecuta en una terminal de Linux mediante el comando `./nombre_del_script.sh`.
- Muestra información en la terminal y guarda un archivo `resultado.txt` con la misma información.

Material/Apuntes/Anotaciones de la Actividad evaluable Integral (Sistemas informáticos)

Videos para reforzar las lecciones de clase:

Qué es la IP pública. Curso de redes desde 0 | Cap 1 | <https://www.youtube.com/watch?v=gVUE2IDwWA0&list=PLSvxAUzJ-XSfY0KpwV8SHBlyLVcrZkENc&index=2>

Direcciones IP y mascararas de RED

<https://www.youtube.com/watch?v=p9onfNNxGyg>

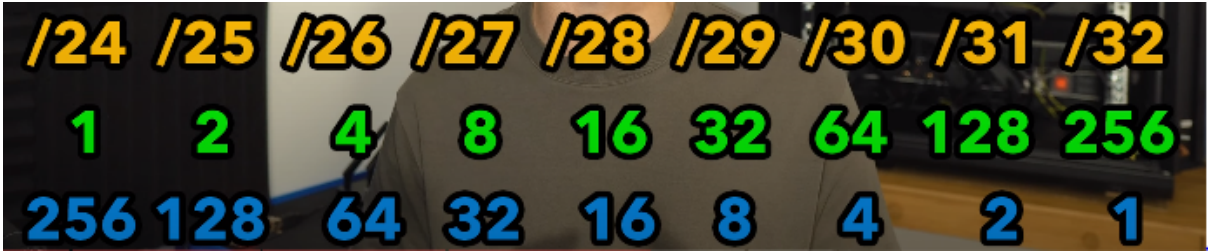
Máscara de red (Qué es y como calcular las redes y hosts posibles)

https://www.youtube.com/watch?v=S7F_7Z3qnzA&t=7s

Direccionamiento IPv4 y Subredes (Explicado)

<https://www.youtube.com/watch?v=SHbBso63X38>

Subneting

A person is holding a large sign that displays a subnetting chart. The chart is organized into three rows and ten columns. The first row contains CIDR notations from /24 to /32. The second row contains the number of subnets for each CIDR, and the third row contains the number of hosts per subnet. The numbers are color-coded: yellow for CIDR, green for number of subnets, and blue for number of hosts.

/24	/25	/26	/27	/28	/29	/30	/31	/32	
1	2	4	8	16	32	64	128	256	
256	128	64	32	16	8	4	2	1	

- máscara

- nº de subredes

- nº de hosts por cada subred