ANÁLISIS DE SERVICIOS DE RED

Práctica 2

Informe de prácticas

Primera Parte

a. Usando la página de manual de *nmap*, documente las funciones y opciones básicas de *nmap*.

nmap es una herramienta para línea de comandos que sirve para efectuar rastreo de puertos. Se usa para evaluar la seguridad de sistemas informáticos, así como para descubrir servicios o servidores en una red informática, para ello envía paquetes definidos a otros equipos y analiza sus respuestas. También se puede utilizar para detección de sistemas conectados a una red, así como conocer servicios y sistemas operativos ejecutándose en la misma.

Si procedemos a lanzar *nmap* sin argumentos, se nos muestra un manual con las opciones más comunes tales como:

- -**sL** (Sondeo de lista). El sondeo de lista es un tipo de descubrimiento de sistemas que tan solo lista cada equipo de la/s red/es especificada/s, sin enviar paquetes de ningún tipo a los objetivos.
- -**sP** (Sondeo *ping*). Esta opción le indica a *nmap* que únicamente realice descubrimiento de sistemas mediante un sondeo *ping*, y que luego emita un listado de los equipos que respondieron al mismo.
- -**n** (No realizar resolución de nombres). Le indica a *nmap* que nunca debe realizar resolución DNS.
- -**sS** (sondeo TCP SYN). El sondeo SYN es relativamente sigiloso y poco molesto, ya que no llega a completar las conexiones TCP. Se envía un paquete SYN y se espera una respuesta. Si se recibe un paquete SYN/ACK indica que el puerto está abierto, mientras que si se recibe un RST indica que no hay nada escuchando en el puerto. Si no se recibe respuesta entonces el puerto se marca como filtrado.
- -**sT** (sondeo TCP connect()). El sondeo TCP Connect() es el sondeo TCP por omisión cuando no se puede utilizar el sondeo SYN. Esto sucede, por ejemplo, cuando el usuario no tiene privilegios para enviar paquetes en crudo o cuando se están sondeando redes IPv6. Nmap le pide al sistema operativo subyacente que establezcan una conexión con el sistema objetivo en el puerto indicado utilizando la llamada del sistema `connect()`, a diferencia de otros tipos de sondeo, que escriben los paquetes a bajo nivel.

- -**sU** (sondeos UDP). Los sondeos UDP funcionan mediante el envío de una cabecera UDP para cada puerto objetivo. Si se obtiene un error ICMP que indica que el puerto no es alcanzable, entonces se marca el puerto

como cerrado. Si se recibe cualquier error ICMP no alcanzable se marca el puerto como filtrado. En algunas ocasiones se recibirá una respuesta al paquete UDP, lo que prueba que el puerto está abierto. Si no se ha recibido ninguna respuesta después de algunas retransmisiones entonces se clasifica el puerto como abierto|filtrado.

- -**p** <rango de puertos>. Esta opción especifica los puertos que desea sondear y toma precedencia sobre los valores por omisión.
- -**p**-. Esta opción activa el escaneo de los 65535 puertos TCP/UDP. Útil para realizar un escaneo completo, ya que por defecto sólo se escanean los 1000 puertos TCP/UDP conocidos.
- -**sV**. Esta opción permite activar la detección de versiones.
- -**O**. Activa la detección de sistema operativo.
- -**v**. Con esta opción se imprime más información sobre el sondeo que se está realizando incrementado el nivel de detalle.

b. Active un proceso de monitorización *tcp* en *mallet* para poder seguir los diferentes métodos de *scanning*.

Para monitorizar todos los paquetes *tcp* que salen de *mallet* utilizamos *tcpdump*, el cual es una herramienta que permite realizar un análisis de tráfico de la red.

Escribimos el comando ***sudo tcpdump -i eth4 tcp port 80 -v***. Con este comando lo que hacemos es monitorizar los paquetes cuyo origen y destino sean el puerto 80.

- **-i**: estamos capturando el tráfico de una interfaz específica, en este caso se trata de la interfaz *eth4* que hace referencia a la máquina *mallet.*
- **tcp**: indicamos que deseamos capturar sólo paquetes del protocolo tcp.
- **port 80**: indicamos que queremos buscar paquetes a un número de puerto específico, en este caso se trata del puerto 80.
- **-v**: aumentamos la cantidad de información de paquetes que se recibe.

A continuación, abrimos otra terminal y procedemos a realizar los escaneos. Tenemos 2 tipos distintos de escaneo, el **escaneo completo** [2] y el **escaneo SYN** [1].

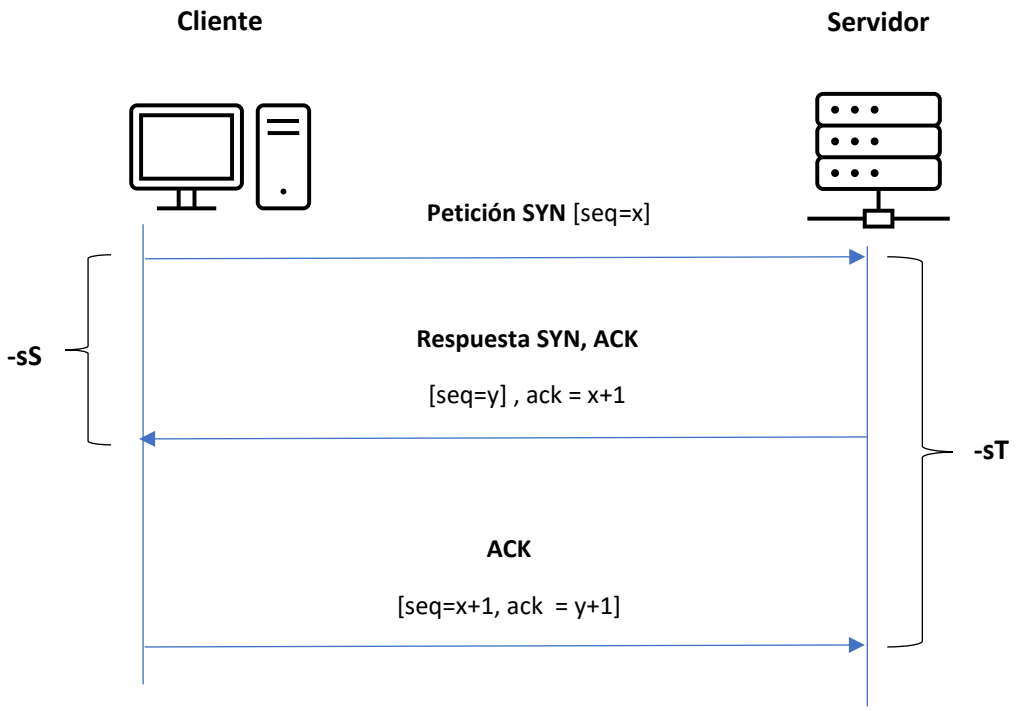
- Comenzamos con el escaneo completo. Para ello utilizamos el siguiente comando: ***nmap 10.0.2.4 -n -p 80 -sT***

```
root@mallet:/home/mallet# sudo tcpdump -i eth4 tcp port 80 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
11:33:37.400575 IP (tos 0x0, ttl 64, id 31175, offset 0, flags [DF], proto TCP (
6), length 60)
    mallet.local.52442 > alice.local.www: Flags [S], cksum 0x5b37 (correct), seq
 1788366836, win 5840, options [mss 1460,sackOK,TS val 538163 ecr 0,nop,wscale 5
], length 0
11:33:37.400953 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 60)
    alice.local.www > mallet.local.52442: Flags [S.], cksum 0xb41b (correct), se
q 1771831887, ack 1788366837, win 5792, options [mss 1460,sackOK,TS val 539463 e
cr 538163,nop,wscale 5], length 0
11:33:37.400960 IP (tos 0x0, ttl 64, id 31176, offset 0, flags [DF], proto TCP (
6), length 52)
    mallet.local.52442 > alice.local.www: Flags [.], cksum 0xf8ce (correct), ack
 1, win 183, options [nop,nop,TS val 538163 ecr 539463], length 0
11:33:37.401316 IP (tos 0x0, ttl 64, id 31177, offset 0, flags [DF], proto TCP (
6), length 52)
    mallet.local.52442 > alice.local.www: Flags [R.], cksum 0xf8c9 (correct), se
q 1, ack 1, win 183, options [nop,nop,TS val 538164 ecr 539463], length 0
```

- Ahora realizamos un escaneo SYN. Para ello utilizamos el siguiente comando: ***nmap 10.0.2.4 -n -p 80 -sS***

```
root@mallet:/home/mallet# sudo tcpdump -i eth4 tcp port 80 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
13:04:37.563026 IP (tos 0x0, ttl 46, id 12192, offset 0, flags [none], proto TCP
 (6), length 44)
    mallet.local.40162 > alice.local.www: Flags [S], cksum 0x3485 (correct), seq
 3684812484, win 3072, options [mss 1460], length 0
13:04:37.563405 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 44)
    alice.local.www > mallet.local.40162: Flags [S.], cksum 0x8e01 (correct), se
q 3838031582, ack 3684812485, win 5840, options [mss 1460], length 0
13:04:37.563411 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 40)
    mallet.local.40162 > alice.local.www: Flags [R], cksum 0x583e (correct), seq
 3684812485, win 0, length 0
```

Esquema Protocolo TCP



El esquema anterior representa cómo funciona el protocolo TCP. Procedamos a analizar los resultados que hemos obtenido en las imágenes.

1. **Sondeo completo (-sT)**
   Según el esquema, en el sondeo completo tenemos 3 etapas: una de petición SYN, otra de respuesta SYN/ACK y otra de ACK.
   - En la petición que manda el cliente (*mallet*) se puede ver cómo se envía el paquete SYN [S], mediante el cual se comienza la conexión.

- Se recibe la respuesta del Servidor (*alice*), el cual manda el paquete SYN/ACK *flag [S.]*, para indicar que el puerto está abierto.
- Ahora, el cliente responde al Servidor con un paquete ACK *flag [.]*, completando así la negociación en 3 pasos (SYN, SYN/ACK, ACK) y la fase de establecimiento de conexión.
- Finalmente, el cliente manda el paquete Reset *flag [R],* para reiniciar la conexión.

2. Sondeo SYN (-sS)

Por su parte, en el sondeo SYN tenemos 2 etapas: una de petición SYN y otra de respuesta SYN/ACK.

- En la petición que manda el cliente (*mallet*) se puede ver cómo se envía el paquete SYN [S], mediante el cual se comienza la conexión.
- Se recibe la respuesta del Servidor (*alice*), el cual manda el paquete SYN/ACK *flag [S.]*, para indicar que el puerto está abierto.
- Finalmente, el cliente manda el paquete Reset *flag [R],* para reiniciar la conexión.

c. Usando *nmap*, realice y documente un barrido de puertos UDP en *alice* y compare con los datos que se obtienen a través de *tcpdump*.

Para realizar un barrido de puertos en *alice*, primero comprobamos que ambas máquinas pueden comunicarse.

```
root@mallet:/home/mallet# ping 10.0.2.4 -c 10
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=5.04 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.489 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.794 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.807 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=1.62 ms
64 bytes from 10.0.2.4: icmp_seq=6 ttl=64 time=1.42 ms
64 bytes from 10.0.2.4: icmp_seq=7 ttl=64 time=0.585 ms
64 bytes from 10.0.2.4: icmp_seq=8 ttl=64 time=1.45 ms
64 bytes from 10.0.2.4: icmp_seq=9 ttl=64 time=0.453 ms
64 bytes from 10.0.2.4: icmp_seq=10 ttl=64 time=0.999 ms

--- 10.0.2.4 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9004ms
```

ping de mallet a alice exitoso

Primera Parte

Comenzamos con el barrido de los puertos tcp. Para ello utilizamos el siguiente comando: **sudo nmap 10.0.2.4 -n -p-**; lo cual realiza un escaneo tcp de tipo SYN[1] de los 65535 puertos.

```
root@mallet:/home/mallet# sudo nmap 10.0.2.4 -n -p-

Starting Nmap 5.00 ( http://nmap.org ) at 2022-09-30 11:16 CEST
Interesting ports on 10.0.2.4:
Not shown: 65522 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
443/tcp   open  https
754/tcp   open  krb_prop
2049/tcp  open  nfs
6000/tcp  open  X11
36769/tcp open  unknown
40588/tcp open  unknown
53032/tcp open  unknown
MAC Address: 08:00:27:48:F3:46 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 4.24 seconds
```

Para realizar el barrido de los puertos udp, utilizamos el siguiente comando: **sudo nmap 10.0.2.4 -n -p- -sU**; lo cual realiza un escaneo UDP[3] de los 65535 puertos. Sin embargo, el tiempo que se requiere para analizar todos los puertos *UDP* es bastante amplio, debido a que el cliente manda varias peticiones a todos los servicios para obtener una respuesta, es decir, comprobar su estado. Debido al elevado número de puertos el proceso resulta lento. Por este motivo, nos vamos a centrar en un puerto en específico, por ejemplo, el puerto 80.
Finalmente utilizamos la siguiente sentencia:

sudo nmap 10.0.2.4 -n -p 80 -sU

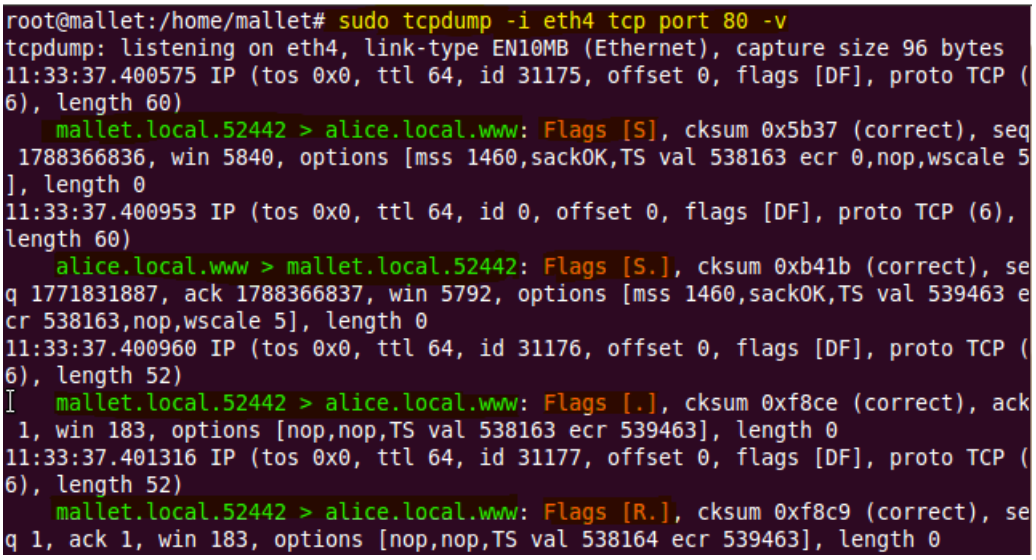
```
root@mallet:/home/mallet# sudo nmap 10.0.2.4 -n -p 80 -sU

Starting Nmap 5.00 ( http://nmap.org ) at 2022-09-30 11:24 CEST
Interesting ports on 10.0.2.4:
PORT   STATE  SERVICE
80/udp closed http
MAC Address: 08:00:27:48:F3:46 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
```

Como se puede ver en la imagen, el puerto 80/udp se encuentra **cerrado**. Esto se puede deber a que cuando el cliente realizó una petición a este servicio, no obtuvo una respuesta o el tiempo que transcurrió fue elevado, por lo que se cancela la petición y se marca el puerto como cerrado.

Segunda Parte

En esta segunda parte, analizaremos cuáles son las diferencias que hay entre escaneos TCP y escaneos UDP utilizando la analizador de tráfico de red *tcpdump*. Para este caso, nos centraremos en analizar un puerto en específico, por ejemplo, el puerto 80.

Comenzamos con el escaneo tipo TCP, para ello utilizamos una segunda terminal en la que analizaremos el tráfico de red mediante: **sudo tcpdump -i eth4 tcp port 80 -v ,** y seguido utilizamos el siguiente comando en otra terminal para realizar el rastreo de puertos mediante : **sudo nmap 10.0.2.4 -n -p 80 -sT**.



Como se puede ver en la imagen, obtenemos el mismo resultado que en el apartado *b) Sondeo Completo*, donde se explica cada paso de manera detallada.

Por otra parte, realizamos el escaneo UDP. Realizaremos el análisis sobre un puerto distinto, por ejemplo, el puerto 67 el cual se corresponde con el Servidor de protocolo Bootstrap. Abrimos una nueva terminal donde rastrearemos los puertos mediante: **sudo nmap 10.0.2.4 -n -p 67 -sU**.

```
root@mallet:/home/mallet# sudo tcpdump -i eth4 udp port 67 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
14:25:42.948827 IP (tos 0x0, ttl 38, id 24380, offset 0, flags [none], proto UDP
 (17), length 28)
    mallet.local.33231 > alice.local.bootps:  [|bootp]
14:27:30.101477 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17),
 length 328)
    mallet.local.bootpc > 10.0.2.3.bootps: BOOTP/DHCP, Request from 08:00:27:62:
45:e6 (oui Unknown), length 300, xid 0x925cfe3a, Flags [none]
          Client-IP mallet.local
          Client-Ethernet-Address 08:00:27:62:45:e6 (oui Unknown) [|bootp]
14:27:30.115989 IP (tos 0x0, ttl 255, id 42, offset 0, flags [none], proto UDP (
17), length 576)
    10.0.2.3.bootps > mallet.local.bootpc: BOOTP/DHCP, Reply, length 548, xid 0x
925cfe3a, Flags [none]
          Client-IP mallet.local
          Your-IP mallet.local
          Client-Ethernet-Address 08:00:27:62:45:e6 (oui Unknown) [|bootp]
```

Como se puede ver en la imagen, se utiliza el protocolo de configuración de hosts Bootstrap (BOOTP), que es el que se utilizó antes de que se desarrollara el protocolo de configuración dinámica de hosts DHCP.

d. Documente toda la información sobre el servicio web que se ejecuta en *alice* usando una simple conexión *telnet*.

telnet es un servicio que permite establecer conexiones remotas con otros ordenadores, servidores y dispositivos. De forma predeterminada se utiliza el puerto de conexión 23. Sin embargo, telnet tiene un gran problema y es que la información desde un terminal a otro viaja sin ningún tipo de cifrado, solamente en texto plano.

Ahora bien, nos están pidiendo la información sobre el servicio web que se ejecuta en *alice*, por lo que debemos comprobar el puerto 80, ya que este puerto es el que se usa para la navegación web de forma no segura (HTTP).

```
root@mallet:/home/mallet# nmap 10.0.2.4 -p 80

Starting Nmap 5.00 ( http://nmap.org ) at 2022-09-26 22:19 CEST
Interesting ports on 10.0.2.4:
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:48:F3:46 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
```

Ejecutamos el comando: **nmap 10.0.2.4 -p 80** para comprobar cuál es el estado del puerto. Como se muestra en la imagen, el puerto se encuentra **abierto**. A continuación, procedemos a ejecutar el siguiente comando desde *mallet*: **telnet 10.0.2.4 80**; donde especificamos la IP y el puerto.

Una vez introducida esta sentencia, estaremos conectados al servidor web y podremos comenzar a pasarle comandos. Como ahora lo que nos interesa es solamente la información que se encuentra en el encabezado podemos utilizar perfectamente el siguiente comando: **HEAD**.

```
root@mallet:/home/mallet# telnet 10.0.2.4 80
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
HEAD
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.14 (Ubuntu) Server at 127.0.0.1 Port 80</address>
</body></html>
Connection closed by foreign host.
```

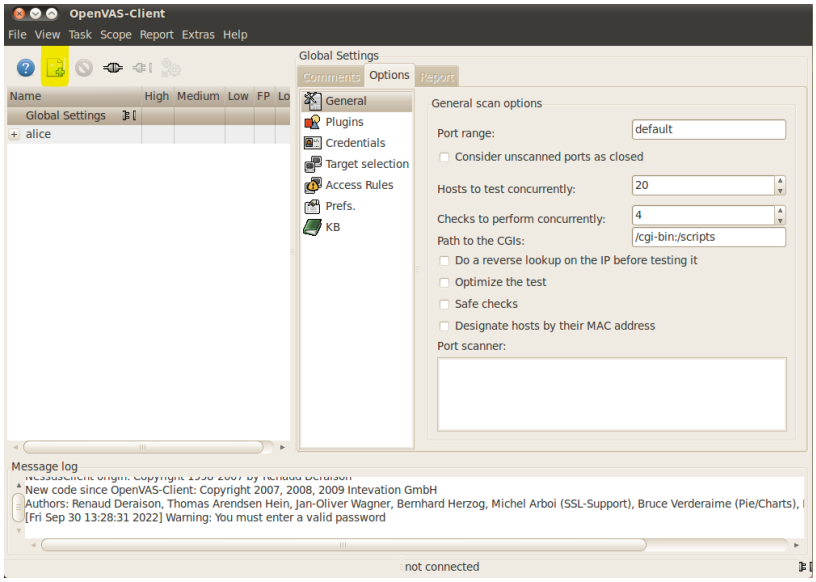
Como se puede ver en la imagen, al introducir el comando **HEAD**, uno de los campos obtenidos es *<address>*, en el cual se especifica el servidor web utilizado y en este caso tenemos el siguiente: **Apache/2.2.14**.

e. Elabore un informe de vulnerabilidades de *alice* usando el analizador de barrido *openvas*. Documente cómo debe iniciarse este analizador y los resultados que encuentre.

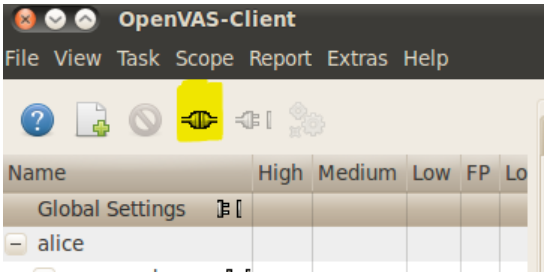
Lo primero que debemos hacer es iniciar el servicio de *openvas* en *mallet*. Para ello utilizamos el siguiente comando: **sudo /etc/init.d/openvas-server start**. Una vez que se haya ejecutado podemos ver el estado en el que se encuentra usando: **status**.

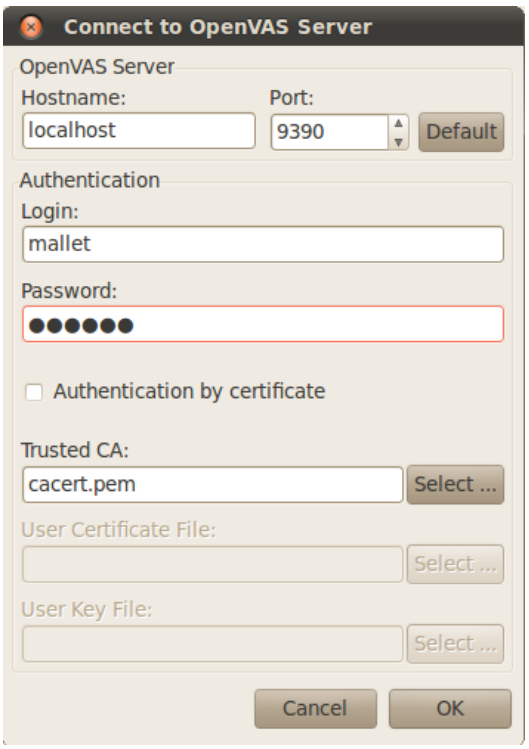
```
root@mallet:/home/mallet# sudo /etc/init.d/openvas-server status
OpenVAS daemon is running
root@mallet:/home/mallet#
```

Una vez levantado el servicio, iniciamos el *openvas* del cliente. Esto lo hacemos clicando en *Applications-Accesories-OpenVAS-Client*.



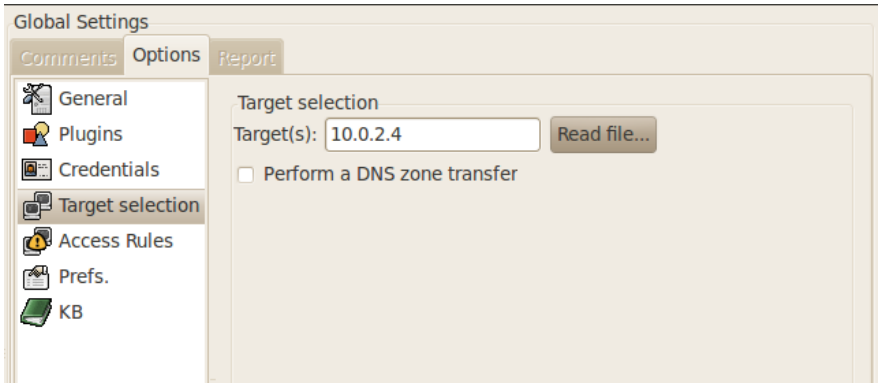
Nos aparecerá la siguiente ventana. Lo que debemos hacer ahora es crear una nueva tarea y darle un nombre, en este caso *alice*. Una vez creada la tarea, nos dirigimos a *Global Settings* y nos conectamos al servidor de OpenVAS.



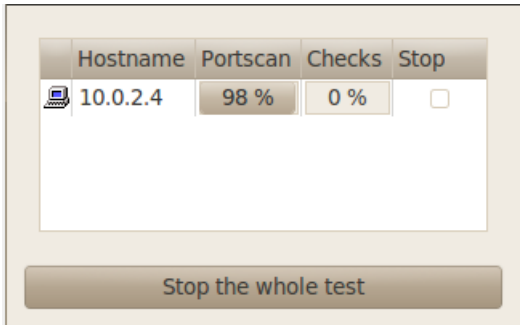


Insertamos la contraseña de *mallet* y pulsamos en OK. Esperamos a que se complete la conexión y una vez que acabe estaremos conectados al Servidor, y esto lo podemos verificar mirando si aparece debajo el mensaje correspondiente: ***Connection: mallet@localhost*.**

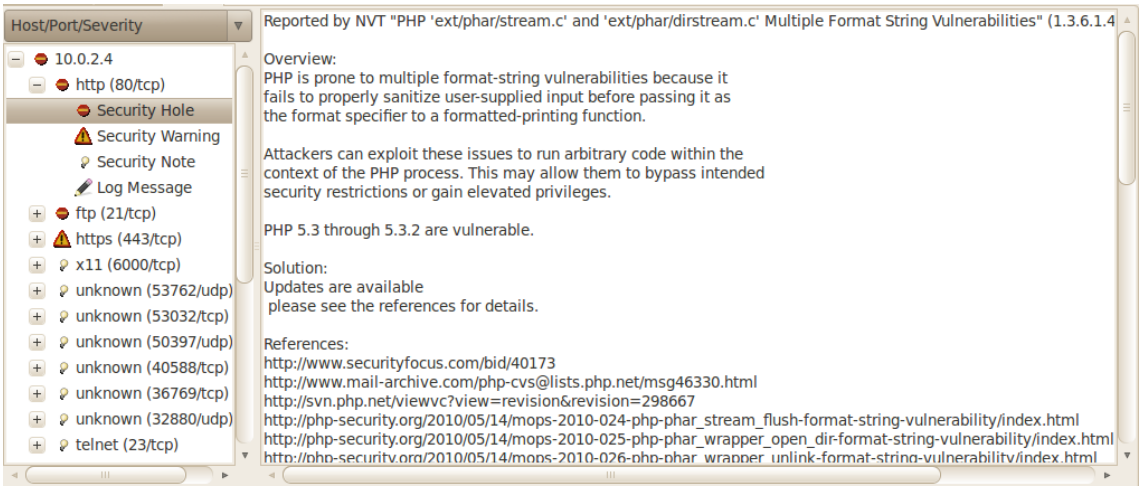
A continuación, en la pestaña de *Options* tenemos que seleccionar un objetivo mediante *Target selection*. Aquí tenemos que poner la IP de la máquina que vayamos a analizar.



Para el caso de *alice* utilizamos la IP: 10.0.2.4. Luego, nos dirigimos a la tarea renombrada con *alice* y pulsamos sobre *Scope* para crear uno nuevo y una vez creado, pulsamos nuevamente sobre *Scope* y lo ejecutamos. Nos pide nuevamente la contraseña para *mallet*, la introducimos y comenzará el barrido de *openvas*.



Nos aparece la siguiente ventana, en la que se indica el porcentaje realizado hasta el momento. Esperamos a que finalice. Una vez haya finalizado, nos aparece lo siguiente:



Donde se muestran todas las vulnerabilidades encontradas en *alice*. Si nos fijamos en la vulnerabilidad del puerto 80/tcp (http), aparece un cuadro en el que se nos muestra una visión general del problema y posibles soluciones.

Problema:

- PHP es propenso a múltiples vulnerabilidades de cadenas de formato porque no sanea adecuadamente la entrada proporcionada por el usuario antes de pasarla como especificador de formato a la función de impresión Formateada.

Solución

- Actualizar los datos disponibles.

f. Recopile la información anterior (desde el punto "b" al punto "e"), ahora para *bob* en lugar de *alice*.

Para realizar este apartado, se seguirá el mismo procedimiento realizado para la máquina *alice*. Por ello, se incluirá algunas explicaciones y las imágenes correspondientes, el resto de la explicación ya se encuentra dada en los apartados anteriores.

b)

Procedemos a realizar los escaneos. Tenemos 2 tipos distintos de escaneo, el **escaneo completo** [2] y el **escaneo SYN** [1].

- Comenzamos con el escaneo completo. Para ello utilizamos el siguiente comando: ***nmap 10.0.2.15 -n -p 80 -sT***

```
root@mallet:/home/mallet# sudo tcpdump -i eth4 tcp port 80 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
14:37:26.053329 IP (tos 0x0, ttl 64, id 26745, offset 0, flags [DF], proto TCP (
6), length 60)
    mallet.local.41624 > 10.0.2.15.www: Flags [S], cksum 0xaa14 (correct), seq 9
65948705, win 5840, options [mss 1460,sackOK,TS val 1400152 ecr 0,nop,wscale 5],
 length 0
14:37:26.053686 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 60)
    10.0.2.15.www > mallet.local.41624: Flags [S.], cksum 0x386d (correct), seq
2120013878, ack 965948706, win 5792, options [mss 1460,sackOK,TS val 6964 ecr 14
00152,nop,wscale 4], length 0
14:37:26.053693 IP (tos 0x0, ttl 64, id 26746, offset 0, flags [DF], proto TCP (
6), length 52)
    mallet.local.41624 > 10.0.2.15.www: Flags [.], cksum 0x7d1f (correct), ack 1
, win 183, options [nop,nop,TS val 1400152 ecr 6964], length 0
14:37:26.053944 IP (tos 0x0, ttl 64, id 26747, offset 0, flags [DF], proto TCP (
6), length 52)
    mallet.local.41624 > 10.0.2.15.www: Flags [R.], cksum 0x7d1b (correct), seq
1, ack 1, win 183, options [nop,nop,TS val 1400152 ecr 6964], length 0
```

- Ahora realizamos un escaneo SYN. Para ello utilizamos el siguiente comando: ***nmap 10.0.2.15 -n -p 80 -sS***

```
root@mallet:/home/mallet# sudo tcpdump -i eth4 tcp port 80 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
14:40:45.106815 IP (tos 0x0, ttl 43, id 53879, offset 0, flags [none], proto TCP
 (6), length 44)
    mallet.local.39761 > 10.0.2.15.www: Flags [S], cksum 0x220c (correct), seq 3
136485234, win 4096, options [mss 1460], length 0
14:40:45.107185 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 44)
    10.0.2.15.www > mallet.local.39761: Flags [S.], cksum 0xf9fb (correct), seq
2320471775, ack 3136485235, win 5840, options [mss 1460], length 0
14:40:45.107192 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 40)
    mallet.local.39761 > 10.0.2.15.www: Flags [R], cksum 0x49c5 (correct), seq 3
136485235, win 0, length 0
```

c)

Primera Parte

Comenzamos con el barrido de los puertos tcp. Para ello utilizamos el siguiente comando: **sudo nmap 10.0.2.15 -n -p-**

```
root@mallet:/home/mallet# sudo nmap 10.0.2.15 -n -p-

Starting Nmap 5.00 ( http://nmap.org ) at 2022-10-02 14:44 CEST
Interesting ports on 10.0.2.15:
Not shown: 65530 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
12345/tcp open  netbus
MAC Address: 08:00:27:63:40:98 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 3.25 seconds
```

Para realizar el barrido de los puertos udp, utilizamos el siguiente comando: **sudo nmap 10.0.2.4 -n -p- -sU.**
Pero como ya vimos con alice, el proceso resulta lento. Por este motivo, nos vamos a centrar en un puerto en específico, por ejemplo, el puerto 80.
Finalmente utilizamos la siguiente sentencia:

sudo nmap 10.0.2.15 -n -p 80 -sU

```
root@mallet:/home/mallet# sudo nmap 10.0.2.15 -n -p 80 -sU

Starting Nmap 5.00 ( http://nmap.org ) at 2022-10-02 14:46 CEST
Interesting ports on 10.0.2.15:
PORT   STATE  SERVICE
80/udp closed http
MAC Address: 08:00:27:63:40:98 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
```

Como se puede ver en la imagen, el puerto 80/udp se encuentra **cerrado**. Esto se puede deber a que cuando el cliente realizó una petición a este servicio, no obtuvo una respuesta o el tiempo que transcurrió fue elevado, por lo que se cancela la petición y se marca el puerto como cerrado.

Segunda Parte

En esta segunda parte, analizaremos cuáles son las diferencias que hay entre escaneos TCP y escaneos UDP utilizando la analizador de tráfico de red *tcpdump*. Para este caso, nos centraremos en analizar un puerto en específico, por ejemplo, el puerto 80.

Comenzamos con el escaneo tipo TCP.

```
root@mallet:/home/mallet# sudo tcpdump -i eth4 tcp port 80 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
14:37:26.053329 IP (tos 0x0, ttl 64, id 26745, offset 0, flags [DF], proto TCP (
6), length 60)
    mallet.local.41624 > 10.0.2.15.www: Flags [S], cksum 0xaa14 (correct), seq 9
65948705, win 5840, options [mss 1460,sackOK,TS val 1400152 ecr 0,nop,wscale 5],
 length 0
14:37:26.053686 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 60)
    10.0.2.15.www > mallet.local.41624: Flags [S.], cksum 0x386d (correct), seq
2120013878, ack 965948706, win 5792, options [mss 1460,sackOK,TS val 6964 ecr 14
00152,nop,wscale 4], length 0
14:37:26.053693 IP (tos 0x0, ttl 64, id 26746, offset 0, flags [DF], proto TCP (
6), length 52)
    mallet.local.41624 > 10.0.2.15.www: Flags [.], cksum 0x7d1f (correct), ack 1
, win 183, options [nop,nop,TS val 1400152 ecr 6964], length 0
14:37:26.053944 IP (tos 0x0, ttl 64, id 26747, offset 0, flags [DF], proto TCP (
6), length 52)
    mallet.local.41624 > 10.0.2.15.www: Flags [R.], cksum 0x7d1b (correct), seq
1, ack 1, win 183, options [nop,nop,TS val 1400152 ecr 6964], length 0
```

Por otra parte, realizamos el escaneo UDP. Realizaremos el análisis sobre el puerto 67 el cual se corresponde con el Servidor de protocolo Bootstrap.

```
root@mallet:/home/mallet# sudo tcpdump -i eth4 udp port 67 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
14:49:46.198025 IP (tos 0x0, ttl 42, id 3547, offset 0, flags [none], proto UDP
(17), length 28)
    mallet.local.38939 > 10.0.2.15.bootps:  [|bootp]
```

Como se puede ver en la imagen, se utiliza el protocolo de configuración de hosts Bootstrap (BOOTP).

d)

Siguiendo el proceso descrito anteriormente, obtenemos lo siguiente:

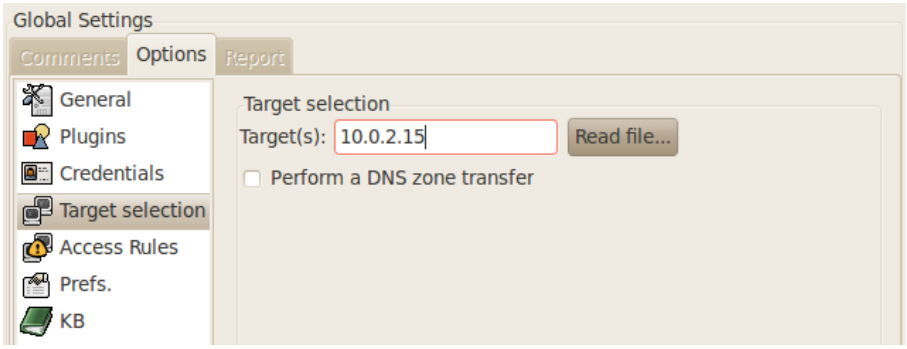
```
root@mallet:/home/mallet# telnet 10.0.2.15 80
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
HEAD
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny8 with Suhosin-Patch Server at 1
27.0.1.1 Port 80</address>
</body></html>
Connection closed by foreign host.
```

Como se puede ver en la imagen, al introducir el comando **HEAD**, uno de los campos obtenidos es *<address>*, en el cual se especifica el servidor web utilizado y en este caso tenemos el siguiente: **Apache/2.2.9**.

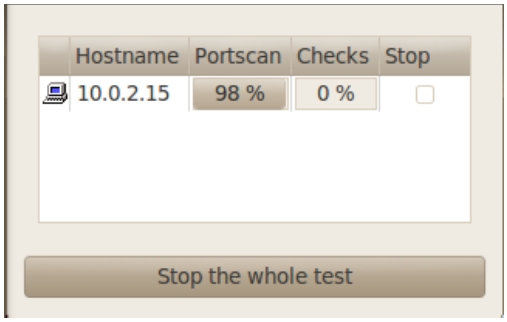
e)

Comenzamos levantando el servicio de *openvas* mediante: **sudo /etc/init.d/openvas-server start**. Una vez levantado, iniciamos el *openvas* del cliente.

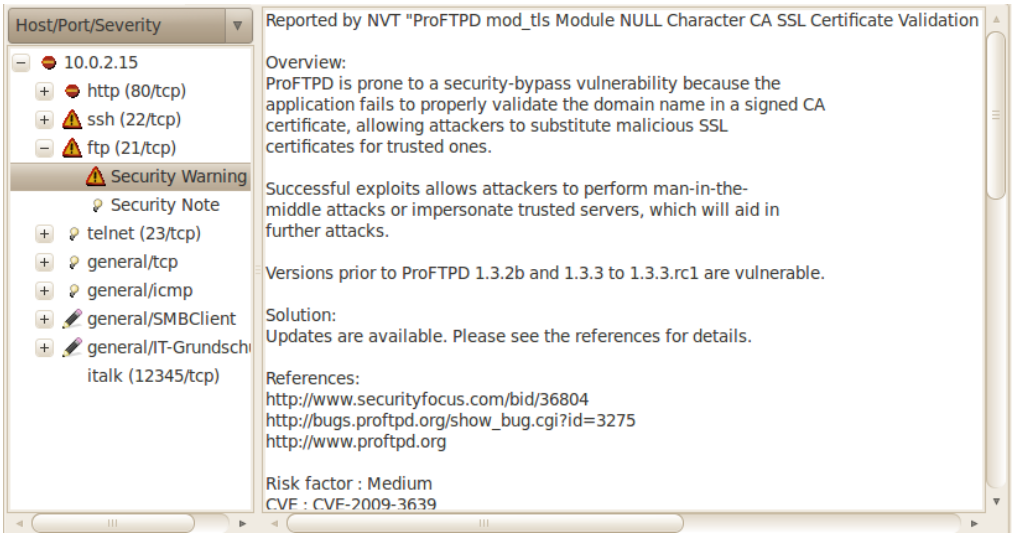
Creamos una nueva tarea llamada *bob* y a continuación, nos conectamos al servidor de OpenVAS.



Ahora nos dirigimos a *Target selection* y ponemos la IP de *bob*, que es la máquina que estamos analizando. Una vez hecho, nos dirigimos a *bob*, creamos un nuevo *Scope* y lo ejecutamos. Ponemos la contraseña de *mallet* y comenzará el barrido de *openvas*.



Una vez haya finalizado, obtenemos todas las vulnerabilidades encontrada en *bob.*



Procedemos a analizar la vulnerabilidad encontrada en el puerto 21/tcp (ftp):

Problema:

- ProFTPD es propenso a una vulnerabilidad de seguridad porque la aplicación no valida correctamente el nombre de dominio en un certificado CA firmado, lo que permite a los atacantes sustituir certificados SSL maliciosos por otros de confianza.

Solución:

- Utilizar las actualizaciones.