GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

Análisis de Servicios en Red, Control de Acceso, Gestión de Usuarios

---

Laura Bezanilla Matellán

11 de octubre de 2022

Índice

1. Funciones y opciones básicas de *nmap* 2

2. Seguimiento de los métodos de *scanning* con una monitorización *tcp* 3

3. Barrido de puertos UDP en alice 7

4. Servicio web que se ejecuta en alice usando una conexión *telnet* 9

5. Vulnerabilidades de alice usando el analizador de barrido *openvas* 11

6. Recopilar la información anterior para la máquina de *bob* 15

a. Monitorización de los escaneos TCP 15

b. Monitorización de los escaneos UDP 16

c. Determinar información sobre el servicio web que se ejecuta en *bob* 17

d. Informe de vulnerabilidades de la máquina de *bob* 19

1. Funciones y opciones básicas de *nmap*

La herramienta *nmap* es el analizador de puertos por excelencia en el ámbito de la ciberseguridad, permitiendo hacer un descubrimiento de aquellos servicios que ofrece una máquina. Los protocolos de la capa de transporte que hay que tener en cuenta como auditor de seguridad a la hora de utilizar *nmap* son TCP y UDP. En total, existen 65.535 puertos para cada protocolo que habría que considerar a la hora de realizar un análisis.

Después de que se haya recopilado información tanto en el manual de la aplicación como en la página oficial, a continuación, se documentan las funciones básicas y más importantes que ofrece esta herramienta:

- *nmap <dirección IP>* : Realiza un escaneo por defecto. Analiza los servicios de los 1000 puertos más utilizados con el protocolo TCP.

- *-v* : Se activa el modo verbose para que se muestren mensajes en la pantalla explicando lo que está ocurriendo por debajo del escaneo.

- *-p-* : Hace un análisis completo de los 65.535 puertos que existen con el protocolo TCP.

- *-sU* : Hace un análisis de los puertos que existen con el protocolo UDP.

- *-p <puerto>* : Hace un análisis del puerto o puertos en concreto. Si se ponen varios van separados por comas. También existe la opción de poner un rango de puertos.

- *-sT* : Hace un escaneo con una conexión completa para servicios con el protocolo TCP.

- *-sS* : Hace un escaneo de tipo *SYN* para servicios con el protocolo TCP. Es decir, las conexiones TCP no son nunca completadas por lo que es más sigiloso que el anterior.

- *-n* : Hace un análisis sin hacer la resolución inversa de DNS. Si ya se tiene la dirección IP, no hace falta averiguar el nombre totalmente cualificado de la máquina. De esta forma, el escaneo será un poco más rápido.

- -R : Permite hacer siempre la resolución inversa de DNS.

- *-sV* : Permite la detección de la versión del servicio.

- -O : Permite detectar la versión del sistema operativo de la máquina remota.

- *-sI* : Permite hacer un escaneo ocultando completamente la dirección IP origen del escaneo. Esta técnica intenta evitar el filtrado de firewall.

2. Seguimiento de los métodos de *scanning* con una monitorización *tcp*

El objetivo de este apartado es poder demostrar que el escaneo con la herramienta *nmap* más la opción *-sT* realiza una conexión TCP completa, mientras que la opción *-sS* no lo hace.

Para monitorizar todos aquellos paquetes que utilicen TCP cuyo origen sea la máquina de *mallet*, se va a utilizar el analizador de tráfico de red *tcpdump* configurado de la siguiente forma.

```
mallet@mallet:~$ sudo tcpdump -i eth4 tcp port 80 -v -e
```

Figura 1. Configuración del analizador de tráfico tcpdump

Al ejecutar dicho comando con privilegios de *root* permite poner la tarjeta de red con alias *eth4* en modo promiscuo. Además, al especificar el protocolo adecuado solo se analizan los segmentos TCP cuyo puerto de origen o de destino sea el 80 para poder analizar el tráfico HTTP.

Para empezar, en la *Figura 2* se muestra cómo se realiza un escaneo TCP completo a un puerto concreto, en este ejemplo se usará el puerto 80, para poder analizar mejor los resultados arrojados por *tcpdump.*

```
mallet@mallet:~$ nmap 10.0.2.4 -n -p 80 -sT

Starting Nmap 5.00 ( http://nmap.org ) at 2022-09-23 12:57 CEST
Interesting ports on 10.0.2.4:
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
```

Figura 2. Escaneo con conexión TCP completa al puerto 80 de alice

```
mallet@mallet:~$ sudo tcpdump -i eth4 tcp port 80 -v -e
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
12:57:23.634035 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:06:ae:f6 (oui Unknown), ethe
length 74: (tos 0x0, ttl 64, id 33780, offset 0, flags [DF], proto TCP (6), length 60)
    mallet.local 60851 > alice.local.www: Flags [S], cksum 0x8399 (correct), seq 220592
ns [mss 1460,sackOK,TS val 1725233 ecr 0,nop,wscale 5], length 0
12:57:23.634442 08:00:27:06:ae:f6 (oui Unknown) > 08:00:27:0e:e1:9f (oui Unknown), ethe
length 74: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    alice.local.www > mallet.local.60851: Flags [S.], cksum 0xf43c (correct), seq 22054
, win 5792, options [mss 1460,sackOK,TS val 1674231 ecr 1725233,nop,wscale 5], length 0
12:57:23.634452 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:06:ae:f6 (oui Unknown), ethe
length 66: (tos 0x0, ttl 64, id 33781, offset 0, flags [DF], proto TCP (6), length 52)
    mallet.local.60851 > alice.local.www: Flags [.], cksum 0x38f0 (correct), ack 1, win
p,TS val 1725233 ecr 1674231], length 0
12:57:23.634726 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:06:ae:f6 (oui Unknown), ethe
length 66: (tos 0x0, ttl 64, id 33782, offset 0, flags [DF], proto TCP (6), length 52)
    mallet.local.60851 > alice.local.www: Flags [R.], cksum 0x38ec (correct), seq 1, ac
 [nop,nop,TS val 1725233 ecr 1674231], length 0
```

Figura 3. Resultado del analizador de tráfico para una conexión TCP completa

Como se puede ver en la imagen anterior, al principio la máquina de *mallet* le manda un paquete a *alice* con la bandera [S] (SYN) cuyo significado es que se desea iniciar una conexión TCP.

A continuación, la máquina de *alice* le contesta con un paquete de confirmación de la conexión con la bandera [S.] (SYN-ACK) el cual representa que la conexión se ha establecido con éxito.

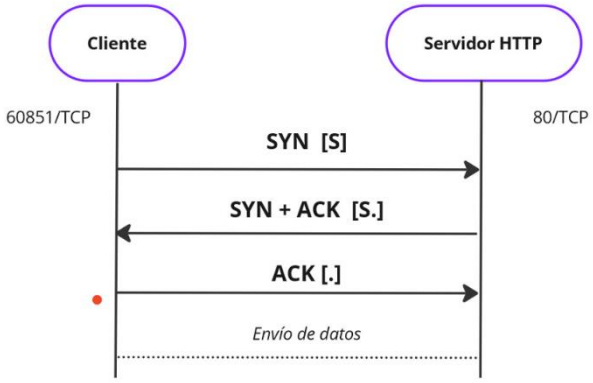

Figura 4. Esquema de una conexión TCP completa

En este caso, la máquina de *mallet* está haciendo uso del puerto 60851. Como se muestra en la *Figura 4* al estar produciéndose un *handshake* completo, la teoría dice que a partir del paquete ACK [.] se va a producir un envío de datos hasta que la conexión se cierre.

Cuando los datos se hayan enviado, la máquina de *mallet* le manda un último paquete con la bandera [R.] (RST) diciendo que la conexión se resetea.

Como TCP es un protocolo orientado a la conexión, es muy fácil saber en el punto marcado en rojo de la *Figura 4* si el puerto que se está escaneando está abierto o cerrado. De esta manera queda comprobado, gracias a las banderas que ofrece el analizador de tráfico *tcpdump,* el modo en el que se establece una conexión TCP completa.

A continuación, se va a hacer el mismo procedimiento anterior, pero con un escaneo de tipo *SYN*. Esto se logra gracias a la opción *-sS* que ofrece la herramienta *nmap*. Por defecto, la aplicación *nmap* realizará un escaneo de tipo *SYN*.

```
mallet@mallet:~$ sudo nmap 10.0.2.4 -n -p 80 -sS

Starting Nmap 5.00 ( http://nmap.org ) at 2022-09-23 13:15 CEST
Interesting ports on 10.0.2.4:
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:06:AE:F6 (Cadmus Computer Systems)

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

Figura 5. Escaneo con conexión SYN al puerto 80 de alice

```
mallet@mallet:~$ sudo tcpdump -i eth4 tcp port 80 -v -e
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
13:15:28.518435 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:06:ae:f6 (oui Unknown), ethertype
length 58: (tos 0x0, ttl 41, id 63225, offset 0, flags [none], proto TCP (6), length 44)
    mallet.local.44910 > alice.local.www: Flags [S], cksum 0x8114 (correct), seq 2754585371,
ns [mss 1460], length 0
13:15:28.518983 08:00:27:06:ae:f6 (oui Unknown) > 08:00:27:0e:e1:9f (oui Unknown), ethertype
length 60: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 44)
    alice.local.www > mallet.local.44910: Flags [S.], cksum 0x4bba (correct), seq 2020453899,
, win 5840, options [mss 1460], length 0
13:15:28.518990 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:06:ae:f6 (oui Unknown), ethertype
length 54: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)
    mallet.local.44910 > alice.local.www: Flags [R], cksum 0xa0cd (correct), seq 2754585372,
```

Figura 6. Resultado del analizador de tráfico para una conexión TCP de tipo SYN con el puerto abierto

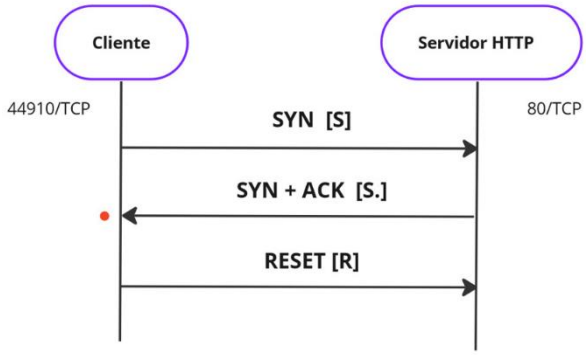

Figura 7. Esquema de una conexión TCP de tipo SYN

En la imagen anterior se muestra el esquema de una conexión TCP de tipo *SYN.* En el punto señalado por la marca roja, el cliente ya tiene la certeza de que el puerto está abierto, lo que significa que hay un servicio a la escucha de recibir peticiones.

Como se puede observar en la *Figura 7*, el puerto 80 siempre va a estar en el lado del servidor. Si un usuario actúa como cliente y quiere establecer una conexión a través de un mensaje de tipo *SYN* al puerto 80 del servidor, es una buena idea que dicho cliente abriese un puerto TCP.

La máquina de *mallet*, que es quien inicia la conexión de tipo *SYN*, está utilizando un puerto por encima del 1024. Si todo va bien y el servidor está de acuerdo con el cliente, este le tiene que mandar un SYN+ACK [S.] con origen la máquina de *alice* más el nombre del servicio relacionado con el puerto 80.

Como es un escaneo de tipo *SYN*, el cliente le devuelve un reset, cuyo significado es que se quiere cerrar la conexión de forma brusca para no completar el *handshake*. En este punto el cliente *mallet* ya tiene la certeza de que el puerto 80 del servidor está abierto debido a que el servidor ha devuelto un [S.].

```
mallet@mallet:~$ sudo nmap 10.0.2.4 -p 90 -sS

Starting Nmap 5.00 ( http://nmap.org ) at 2022-10-09 19:02 CEST
Interesting ports on 10.0.2.4:
PORT   STATE  SERVICE
90/tcp closed dnsix
MAC Address: 08:00:27:06:AE:F6 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Figura 8. Escaneo con conexión SYN al puerto 90 de alice

```
mallet@mallet:~$ sudo tcpdump -i eth4 tcp port 90 -v -e
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
19:04:45.123636 08:00:27:bf:6a:bf (oui Unknown) > 08:00:27:06:ae:f6 (oui Unknov
length 58: (tos 0x0, ttl 52, id 16534, offset 0, flags [none], proto TCP (6), l
    mallet.local.42361 > alice.local.90: Flags [S], cksum 0xc432 (correct), seq
s [mss 1460], length 0
19:04:45.123953 08:00:27:06:ae:f6 (oui Unknown) > 08:00:27:bf:6a:bf (oui Unknov
length 60: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length
    alice.local.90 > mallet.local.42361: Flags [R.], cksum 0xdfdb (correct), se
ength 0
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
```

Figura 9. Resultado del analizador de tráfico para una conexión TCP de tipo SYN con el puerto cerrado

No obstante, en caso de que el puerto estuviera cerrado, el servidor le hubiera contestado con un reset indicando que no puede conectarse a ese puerto porque no hay ningún servicio escuchando. Este ejemplo se muestra en las *Figura 8* y *Figura 9* respectivamente.

Finalmente, se hará un barrido de todos los puertos TCP de la máquina de *alice*. Como no es necesario hacer una resolución inversa de DNS para obtener el nombre de la maquina se utiliza la opción *-n*. Además, gracias a la opción *-r* se podrá hacer el escaneo de manera secuencial.

```
mallet@mallet:~$ sudo nmap 10.0.2.4 -n -p- -r
[sudo] password for mallet:

Starting Nmap 5.00 ( http://nmap.org ) at 2022-10-02 19:34 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
622/tcp   open  unknown
2049/tcp  open  nfs
6000/tcp  open  X11
54755/tcp open  unknown
54885/tcp open  unknown
59419/tcp open  unknown
MAC Address: 08:00:27:06:AE:F6 (Cadmus Computer Systems)
```

Figura 10. Escaneo TCP completo de alice en orden secuencial

3. Barrido de puertos UDP en alice

Antes de empezar, lo primero que se comprueba es si la máquina de *mallet* se puede comunicar con la de *alice*. Esto se puede verificar de distintas formas, pero en este ejemplo se realizará un *ping* entre ambas máquinas.

Para realizar un barrido de los puertos UDP en *alice* se hará un análisis de los 65.535 puertos UDP de la máquina de *alice* como se muestra a continuación en la *Figura 11.* Este tipo de escaneo es más lento porque hay que permitir un tiempo de espera más prolongado antes de que se pueda suponer que un puerto está cerrado.

```
mallet@mallet:~$ sudo nmap 10.0.2.4 -n -p- -sU

Starting Nmap 5.00 ( http://nmap.org ) at 2022-10-02 19:39 CEST
```

Figura 11. Escaneo UDP completo de alice

¿Cómo sabe la herramienta *nmap* que un puerto UDP está abierto? Para contestar a esta pregunta se va a hacer un escaneo al puerto 67/UDP, el cual está relacionado con el servicio DHCP, y al mismo tiempo se va a estudiar que ocurre por debajo con un analizador de tráfico de red.

```
mallet@mallet:~$ sudo nmap 10.0.2.4 -n -p67 -sU

Starting Nmap 5.00 ( http://nmap.org ) at 2022-09-29 12:51 CEST
Interesting ports on 10.0.2.4:
PORT   STATE  SERVICE
67/udp closed dhcps
MAC Address: 08:00:27:06:AE:F6 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Figura 12. Escaneo UDP al puerto 67 de alice

Se quiere únicamente que *tcpdump* muestre aquel tráfico que utiliza como protocolo UDP para el puerto 67 en los datagramas de origen o de destino.

```
mallet@mallet:~$ sudo tcpdump -i eth4 udp port 67 -v
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
12:51:07.146011 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 328)
    alice.local.bootpc > 10.0.2.3.bootps: BOOTP/DHCP, Request from 08:00:27:06:ae:f6 (oui Unknown)
 length 300, xid 0xc534d245, Flags [none]
          Client-IP alice.local
          Client-Ethernet-Address 08:00:27:06:ae:f6 (oui Unknown) [|bootp]
12:51:07.163213 IP (tos 0x0, ttl 255, id 30, offset 0, flags [none], proto UDP (17), length 576)
    10.0.2.3.bootps > alice.local.bootpc: BOOTP/DHCP, Reply, length 548, xid 0xc534d245, Flags [no
e]
          Client-IP alice.local
          Your-IP alice.local
          Client-Ethernet-Address 08:00:27:06:ae:f6 (oui Unknown) [|bootp]
12:51:14.119258 IP (tos 0x0, ttl 59, id 52975, offset 0, flags [none], proto UDP (17), length 28)
    mallet.local.33324 > alice.local.bootps:  [|bootp]
^C
3 packets captured
3 packets received by filter
0 packets dropped by kernel
mallet@mallet:~$
```

Figura 13. Resultado de tcpdump para un escaneo UDP al puerto 67

Como se puede observar en la imagen anterior, para que *nmap* pueda inferir que el servicio UDP está activo lo que tiene que hacer es mandarle peticiones al servidor. Si se manda una petición como cliente y el servidor no responde, en conclusión, dicho cliente puede inferir que el puerto está cerrado. En esta ocasión, el analizador de tráfico está mostrando mensajes relacionados con el servicio DHCP como se puede observar en la *Figura 13*.

En el caso de hacer un escaneo de tipo UDP al puerto 53, *tcpdump* mostraría mensajes sobre el servicio DNS, que es el sistema de nombres de dominio, como se observa en las siguientes imágenes.

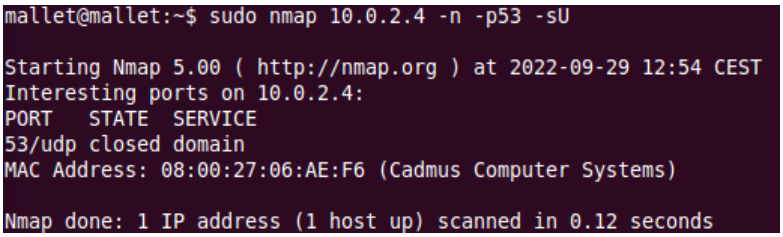

Figura 14. Escaneo UDP al puerto 53 de alice

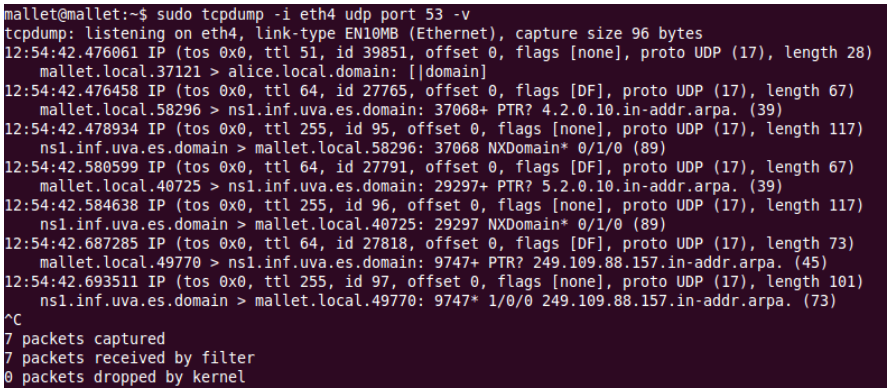

Figura 15. Resultado de tcpdump para un escaneo UDP al puerto 53

Como conclusión, la herramienta *nmap* sabe si el puerto está abierto mandando una petición. Si el servidor responde al cliente, este puede estar seguro de que el servicio está activo. Sin embargo, si al cliente no le llega una respuesta desde el servidor el estado del puerto es cerrado por lo que no existiría un servicio escuchando en dicho puerto.

4. Servicio web que se ejecuta en alice usando una conexión *telnet*

El protocolo de red Telnet ofrece una comunicación bidireccional, por lo que se utiliza para administrar a distancia sistemas informáticos.

Lo primero que es necesario hacer es iniciar el sniffer de contraseñas *dsniff.* Se van a utilizar las opciones *-m* para permitir la detección automática de los protocolos y *-i* para definir la interfaz en la que el sniffer debe escuchar.

Una vez hecho esto, desde la máquina de *mallet* se abre otro terminal para realizar una conexión Telnet hacia *alice*. Para ello en la máquina del cliente se le proporciona la dirección IP de la máquina a la que nos queremos conectar, en este caso sería la de *alice*. Como dato curioso, no es necesario especificar el puerto TCP de *alice* porque el cliente ya sabe que dicho puerto, por defecto, es el 23 para el servicio de Telnet.

Una vez se está dentro de la máquina de *alice*, se puede probar a ejecutar diferentes comandos. Como Telnet no utiliza ningún mecanismo de encriptación, todo lo que se haga durante la sesión no va protegido, por lo que puede ser fácilmente interceptado. Si alguien se autentica con la cuenta de *root*, para tener privilegios en la máquina a la que se conecta, los ciberdelincuentes le pueden hacer mucho daño.

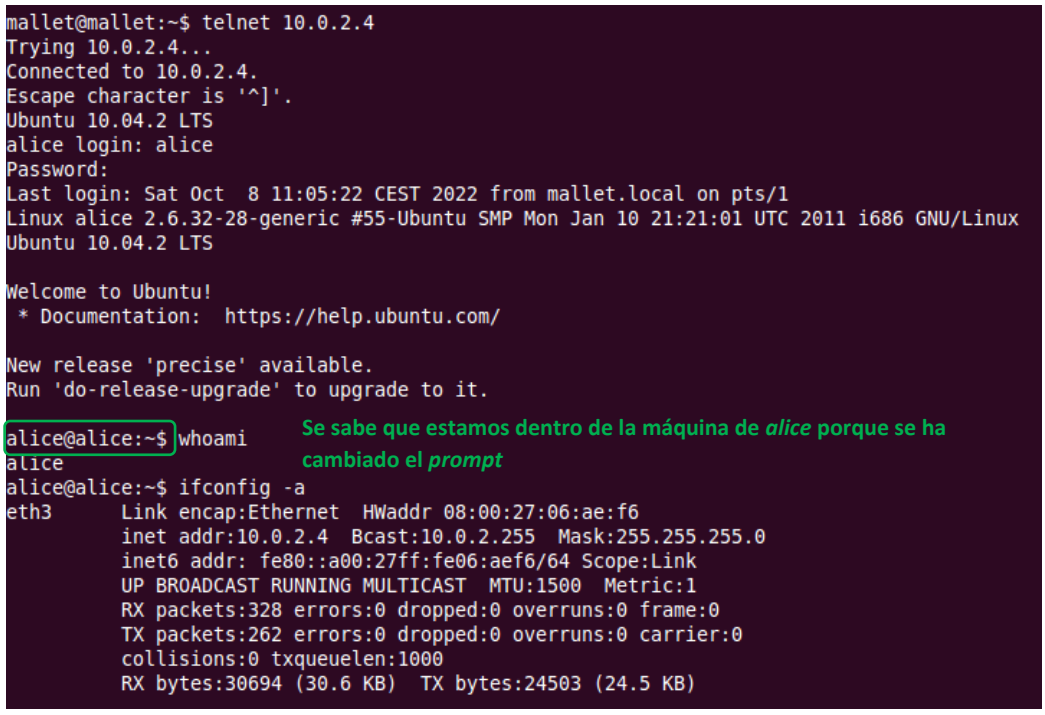

Figura 16. Conexión Telnet con alice

Para ver si realmente se ha capturado toda la información que se ha tecleado durante la sesión hay que salir de la conexión Telnet. Como por defecto Telnet no cifra el tráfico, gracias al sniffer se puede ver informacion privilegiada como el nombre del usuario y su contraseña, además de todos los comandos ejecutados hasta finalizar la conexión.

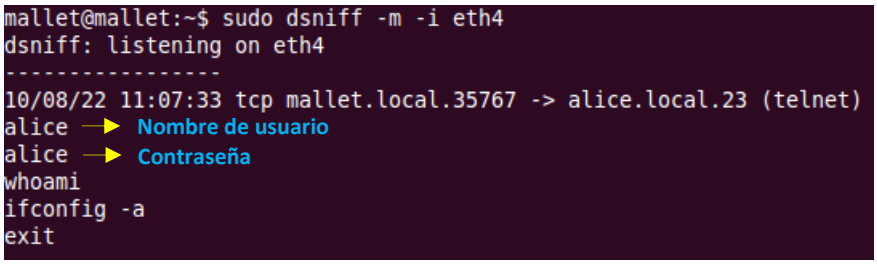

Figura 17. Información capturada por el sniffer durante la sesión Telnet con alice

Para comprobar la disponibilidad del servicio web que se ejecuta en *alice* basta con conectarse al puerto 80 con una simple conexión Telnet. Esta herramienta tiene la capacidad de mostrar cualquier salida del proceso que sirve al puerto. Dicha salida puede proporcionar pistas útiles para poder identificar tanto el servicio como su versión.

Para ello se realiza una conexión al servidor web vía Telnet a través del puerto 80. Una vez se está dentro, se va a lanzar una petición HTTP en la que se solicita el documento completo gracias a la petición GET.

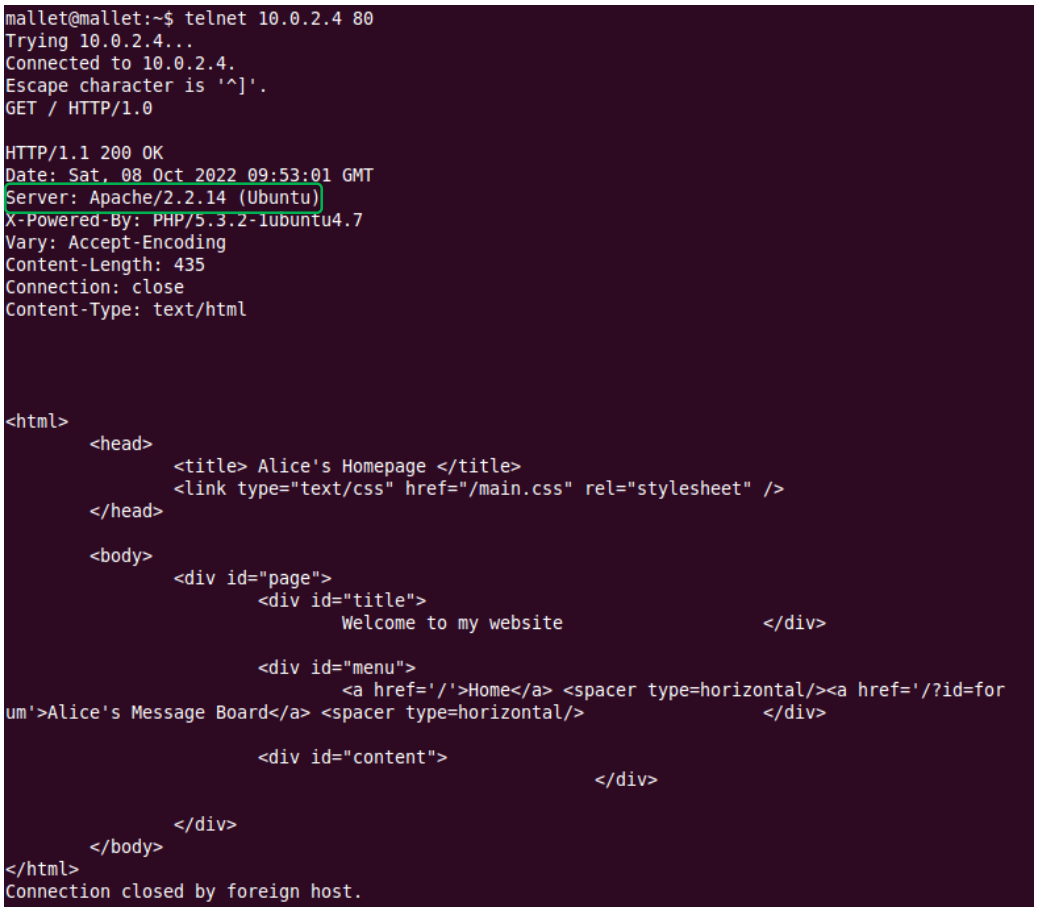

Figura 18. Documento completo de la petición GET del servicio web de alice

Como se puede observar en la *Figura 18* el servidor web que se ejecuta en *alice* es Apache (Ubuntu) con la versión 2.2.14.

5. Vulnerabilidades de alice usando el analizador de barrido *openvas*

Navegando un poco por las aplicaciones que vienen incluidas en la máquina de *mallet*, se puede ver que existe el cliente de este analizador. Sin embargo, un cliente no sirve de nada si no está su servicio activo.

Por ello, lo primero que se debe hacer es iniciar el servicio de *openvas*. Para comprobar que de verdad se ha activado correctamente se comprueba su estado como se muestra en la siguiente imagen.

```
mallet@mallet:~$ sudo /etc/init.d/openvas-server start
[sudo] password for mallet:
W32.Sasser.Worm.nasl could not be added to the cache and is likely to stay invisible to the client.
openvasd.
mallet@mallet:~$
mallet@mallet:~$ sudo /etc/init.d/openvas-server status
OpenVAS daemon is running
```

Figura 19. Iniciar el servicio de Openvas y comprobación de su estado

Otra forma de comprobarlo es utilizar *nmap* para escanear el puerto de *openvas* de la siguiente forma.

```
mallet@mallet:~$ sudo nmap localhost -p 9390

Starting Nmap 5.00 ( http://nmap.org ) at 2022-10-03 20:20 CEST
Warning: Hostname localhost resolves to 2 IPs. Using 127.0.0.1.
Interesting ports on localhost (127.0.0.1):
PORT     STATE SERVICE
9390/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Figura 20. Escaneo del puerto perteneciente a Openvas

Una vez se tiene activo el servicio, el siguiente paso es abrir el cliente de *openvas*, el cual se encuentra en la máquina de *mallet* donde muestra la siguiente imagen.

Figura 21. Ubicación del cliente Openvas

A continuación, hay que conectarse al servidor. Lo primero que hay que hacer es añadir una nueva tarea con el nombre de *alice*. Luego, desde la configuración global, hay que conectarse al servidor haciendo *click* en el símbolo de los cables conectados.

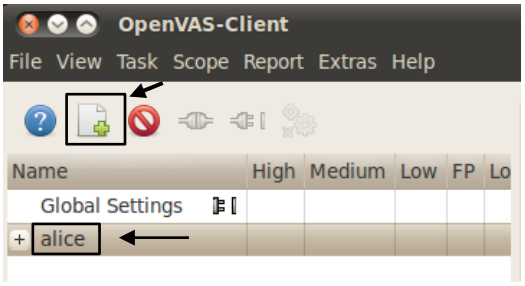

Figura 22. Botón para añadir una nueva tarea con el nombre de alice

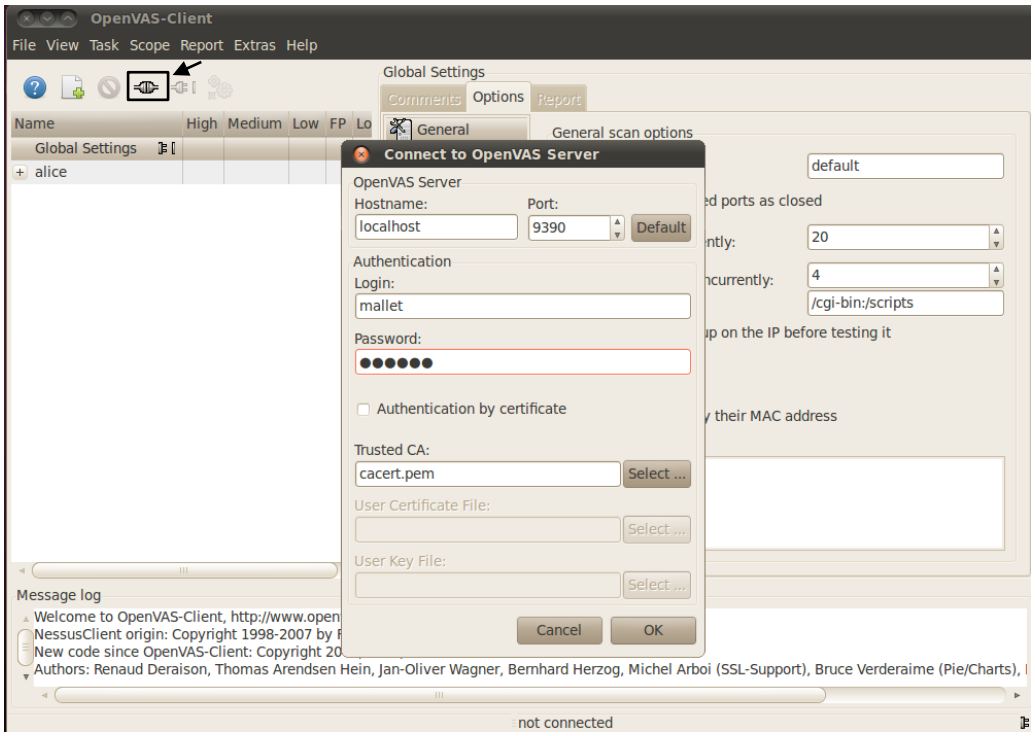

Figura 23. Botón para conectarse al servidor de openvas

Posteriormente, se rellenan los datos que se piden en la nueva ventana emergente. Debido a que tanto el cliente como el servidor van a estar en la misma máquina, en el campo *hostname* se escribe *localhost*. Luego se introduce la contraseña de la máquina de *mallet* y se pulsa el botón de OK. Se puede comprobar que se ha creado la conexión correctamente por el mensaje que aparece en la parte inferior de la ventana.

Figura 24. Comprobación de la correcta conexión

En la zona de *target selection,* como únicamente se quiere escanear la máquina de *alice*, solo se indica su dirección IP. Sin embargo, existe la opción de poner un fichero con una lista de máquinas que se quieran analizar. La opción "*Perform a DNS zone transfer*" sirve, en caso de que la máquina de *alice* sea un servidor de nombres, para poder averiguar su información.

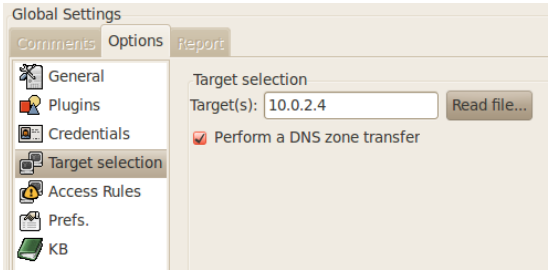

Figura 25. Configuración del target selection del cliente openvas

El siguiente paso sería hacer click en la tarea de *alice*. Irse a la barra de opciones de la parte superior donde dice *Scope*. En el menú desplegable que aparece se pincha en la opción *New*. Esto creará un nuevo scope que podrá ser ejecutado. Para ello, se hace click sobre el nuevo scope, se va a la opción del menú superior y se pulsa *Execute*.

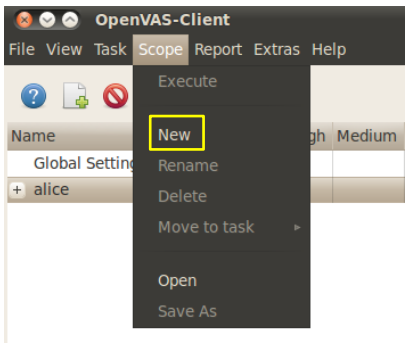

Figura 26. Crear un nuevo scope

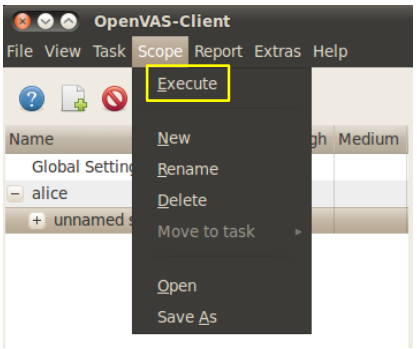

Figura 27. Ejecutar el nuevo scope

Una vez se han completado todos los pasos anteriores, debería aparecer una nueva ventana donde se muestra el avance del proceso del analizador de *openvas*.

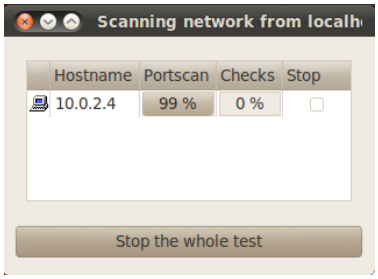

Figura 28. Información del proceso del escaneo

Cuando el escaneo llega al 100%, se podrá comprobar cómo se ha creado un informe con todas las vulnerabilidades que se han encontrado en la máquina de *alice*.

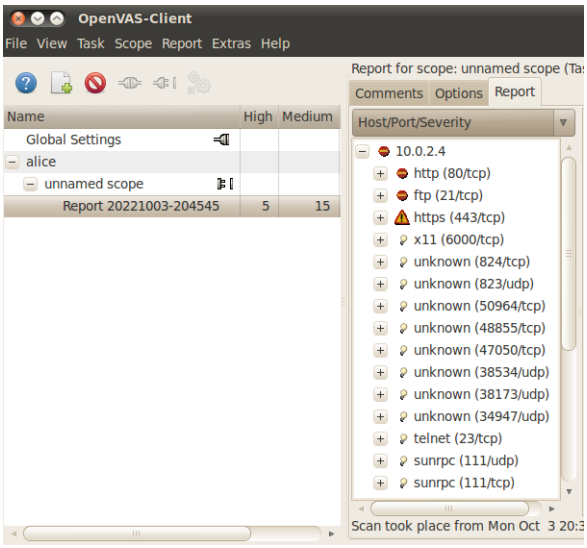

Figura 29. Informe de las vulnerabilidades de la máquina de alice

Documentación de los resultados que ofrece el analizador de barrido *openvas*:

- Vulnerabilidad HTTPS (443/tcp):
    - **Problema**: phpMyAdmin es propenso a múltiples vulnerabilidades de validación de entrada, incluyendo un problema de inyección SQL. Un ataque exitoso puede robar credenciales de autenticación basadas en cookies, comprometer la aplicación, acceder o modificar datos.
    - **Solución**: Actualizar a la versión disponible en la página web.
- Vulnerabilidad SMTP (25/tcp):
    - **Problema**: Algunos antivirus mueren cuando procesan un correo electrónico con una cadena demasiado larga sin saltos de línea.
    - **Solución**: Si hay un antivirus, es posible que se haya bloqueado. Se debe comprobar su estado, ya que no es posible hacerlo de forma remota.
- Vulnerabilidad TCP:
    - **Problema**: Como esta definido en el RFC1323, la máquina de *alice* implementa marcas de tiempo TCP. Una consecuencia de esta característica es que el tiempo de actividad del host a veces puede ser calculado.

6. Recopilar la información anterior para la máquina de *bob*

a. Monitorización de los escaneos TCP

En primer lugar, se ha realizado un escaneo TCP completo al puerto 80 de la máquina de *bob*. Como se puede observar en la salida por pantalla del analizador de tráfico *tcpdump,* se puede comprobar como se establece una conexión TCP completa de manera análoga a cuando se realizó sobre la máquina de *alice*.

```
mallet@mallet:~$ sudo nmap 10.0.2.15 -n -p 80 -sT

Starting Nmap 5.00 ( http://nmap.org ) at 2022-10-08 18:30 CEST
Interesting ports on 10.0.2.15:
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:CE:A3:62 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Figura 30. Escaneo TCP completo al puerto 80 de bob

```
mallet@mallet:~$ sudo tcpdump -i eth4 tcp port 80 -v -e
tcpdump: listening on eth4, link-type EN10MB (Ethernet), capture size 96 bytes
18:42:31.247992 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:ce:a3:62 (oui Unknown),
0800), length 74: (tos 0x0, ttl 64, id 29267, offset 0, flags [DF], proto TCP (6),
    mallet.local.54818 > 10.0.2.15.www: Flags [S], cksum 0x3570 (correct), seq 2547
options [mss 1460,sackOK,TS val 304010 ecr 0,nop,wscale 5], length 0
18:42:31.248345 08:00:27:ce:a3:62 (oui Unknown) > 08:00:27:0e:e1:9f (oui Unknown),
0800), length 74: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), leng
    10.0.2.15.www > mallet.local.54818: Flags [S.], cksum 0x94cf (correct), seq 271
41434, win 5792, options [mss 1460,sackOK,TS val 248021 ecr 304010,nop,wscale 4], l
18:42:31.248354 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:ce:a3:62 (oui Unknown),
0800), length 66: (tos 0x0, ttl 64, id 29268, offset 0, flags [DF], proto TCP (6),
    mallet.local.54818 > 10.0.2.15.www: Flags [.], cksum 0xd981 (correct), ack 1, w
op,nop,TS val 304010 ecr 248021], length 0
18:42:31.248646 08:00:27:0e:e1:9f (oui Unknown) > 08:00:27:ce:a3:62 (oui Unknown),
0800), length 66: (tos 0x0, ttl 64, id 29269, offset 0, flags [DF], proto TCP (6),
    mallet.local.54818 > 10.0.2.15.www: Flags [R.], cksum 0xd97d (correct), seq 1,
tions [nop,nop,TS val 304010 ecr 248021], length 0
```

Figura 31. Resultado del analizador para una conexión TCP completa en bob

La única diferencia existente en este proceso es el número de puerto que se abre en la máquina local de *mallet.* Esto se debe a que es una petición distinta a la que se hizo previamente.

Acto seguido, se escaneará el mismo puerto, pero esta vez se configurará la herramienta *nmap* para que realice un escaneo de tipo *SYN*.

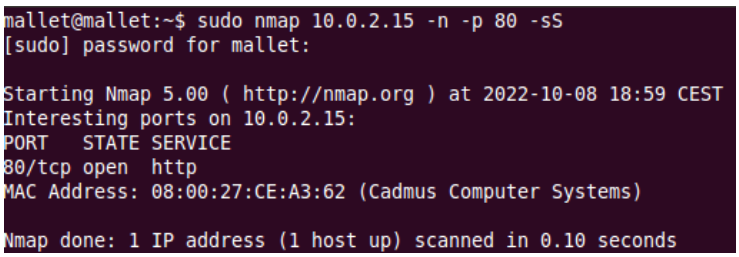

Figura 32. Escaneo con conexión SYN al puerto 80 de bob

En este caso, la conexión TCP de tipo *SYN* se desarrolló solo con los tres mensajes necesarios para no completar el handshake.

Para finalizar esta primera parte, se ejecutará un barrido de todos los puertos TCP de la máquina de *bob*. No será necesario hacer una resolución inversa de DNS y se hará el escaneo de manera secuencial. En la siguiente imagen se muestra otra forma de realizar un escaneo a todos los puertos utilizando rangos.

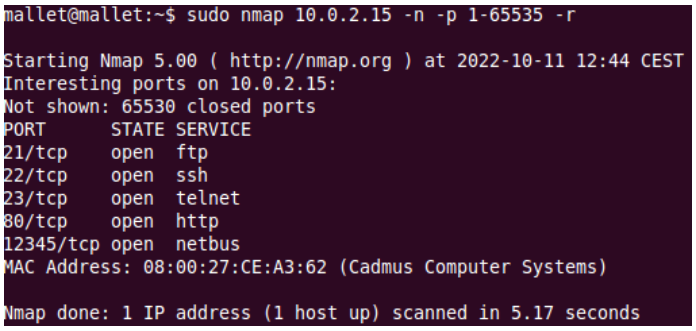

Figura 33. Escaneo TCP completo de alice en orden secuencial

Observando la imagen anterior, es necesario destacar el hecho de que la máquina de *bob* tiene muchos menos servicios activos en comparación con la de *alice*.

b. Monitorización de los escaneos UDP

Una vez activado un proceso de monitorización TCP en la máquina de *bob* para poder seguir los diferentes métodos de scanning, se va a hacer lo mismo, pero en este caso con UDP.

Para empezar, se ha hecho un escaneo UDP del puerto 68 de la máquina de *bob*. En este ejemplo, el puerto aparece como abierto|filtrado esto quiere decir que nmap no es capaz de determinar si el puerto se encuentra abierto o si por el contrario existe un firewall de paquetes.

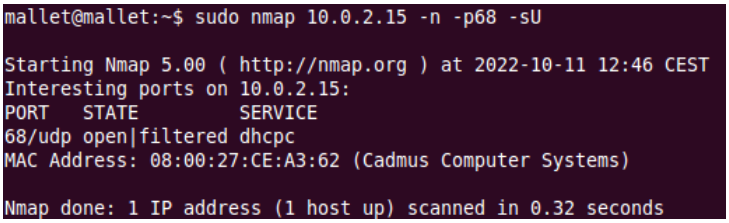

Figura 34. Escaneo UDP del puerto 80 de bob

c. Determinar información sobre el servicio web que se ejecuta en *bob*

En este punto, se va a documentar toda la información sobre el servicio web que se ejecuta en *bob* usando una conexión Telnet. Para ello hay que iniciar en un terminal el sniffer de contraseñas *dsniff* con las opciones adecuadas.

En otro terminal aparte, en la máquina de *mallet,* se abre una conexión Telnet hacia *bob* como se muestra en la siguiente imagen.

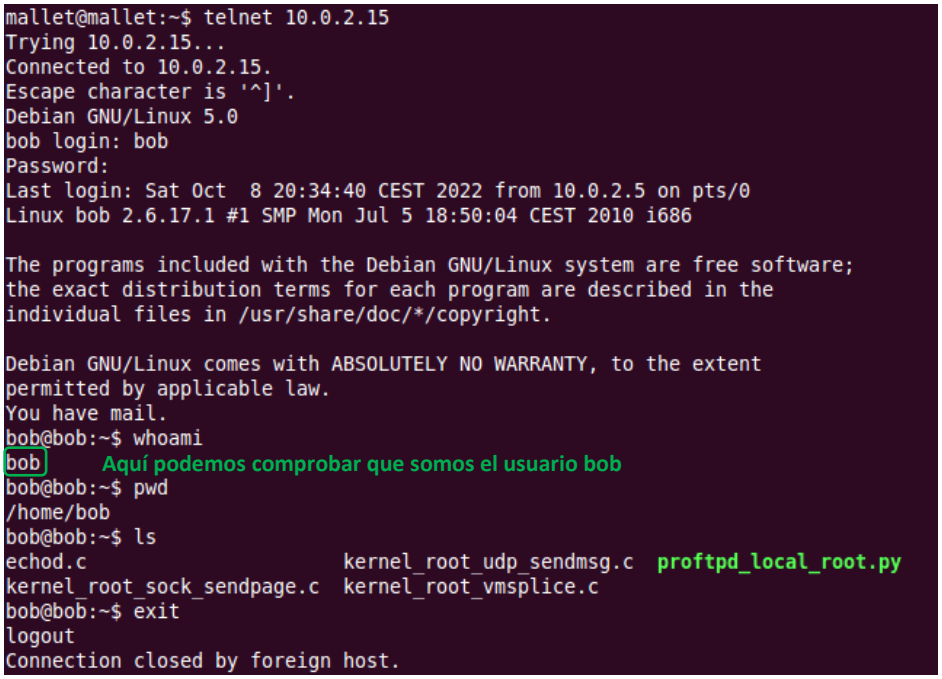

Figura 35. Conexión Telnet con bob

De la misma forma que ocurría anteriormente con la conexión Telnet hacia *alice*, en este caso también se pueden ver todos los comandos usados durante la sesión gracias al *sniffer*.

```
mallet@mallet:~$ sudo dsniff -m -i eth4
dsniff: listening on eth4
-----------------
10/08/22 20:35:35 tcp mallet.local.51023 -> 10.0.2.15.23 (telnet)
bob
bob
whoami
pwd
ls
exit
```

Figura 36. Información capturada por el sniffer durante la sesión Telnet con bob

Para poder documentar toda la información sobre el servicio web que se ejecuta en *bob* se usará una conexión Telnet para conectarse a su puerto 80. Cuando se haya ejecutado el comando para establecer la conexión, se le envía una petición HTTP en la que se solicita la cabecera de la misma como se muestra en la imagen.

```
mallet@mallet:~$ telnet 10.0.2.15 80
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sat, 08 Oct 2022 18:44:30 GMT
Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny8 with Suhosin-Patch
X-Powered-By: PHP/5.2.6-1+lenny8
Set-Cookie: ca835771a6589ed0ff94355cd5a5841f=f614c8fc07f4a8b22edf711ca3134ba3; path=/
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Expires: Mon, 1 Jan 2001 00:00:00 GMT
Last-Modified: Sat, 08 Oct 2022 18:44:31 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=utf-8

Connection closed by foreign host.
```

Figura 37. Información sobre la petición HEAD del servicio web de bob

Cabe destacar que esta vez el servidor web que se ejecuta en *bob* es Apache (Debian) con la versión 2.2.9. A diferencia del servidor web de *alice*, en este caso también se muestra la versión de PHP que se ejecuta en *bob*.

d. Informe de vulnerabilidades de la máquina de *bob*

Para finalizar, se va a elaborar un informe de las vulnerabilidades de *bob* usando el analizador de barrido *openvas*. Después, de realizar todos los pasos tanto para activar el servicio *openvas* en la máquina de *mallet* como para configurar correctamente el cliente, en la siguiente imagen se puede observar como en la máquina de *bob* existen muchas menos vulnerabilidades que en la máquina de alice.

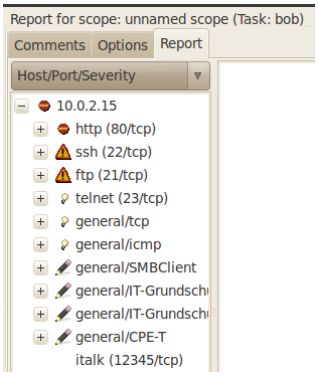

Figura 38. Informe de vulnerabilidades de bob con openvas

Documentación de los resultados que ofrece el analizador de barrido *openvas*:

- Vulnerabilidad ssh (22/tcp):
    - **Problema**: Bob tiene instalado OpenSSH siendo propenso a una vulnerabilidad de divulgación de información. El fallo se debe a la gestión inadecuada de los errores dentro de una sesión SSH cifrada. Esto permitirá a los ciberdelincuentes obtener cuatro bytes de texto plano de una sesión cifrada.
    - **Solución**: Actualizar a una versión superior de OpenSSH.

- Vulnerabilidad ftp (21/tcp):
    - **Problema**: Bob está ejecutando el servidor ProFTPD, que está expuesto a una vulnerabilidad de falsificación de petición en sitios cruzados. El fallo se debe a que la aplicación trunca un comando FTP demasiado largo e interpreta incorrectamente la cadena restante como un nuevo comando FTP. Esto puede ser explotado para ejecutar comandos FTP arbitrarios.
    - **Solución**: La corrección está disponible en el repositorio SVN, desde donde puede ser descargada.