HERRAMIENTAS DE AUDITORIA

1) Netcat — chat y transferencia de fichero (TCP simple)

Objetivo: entender la comunicación TCP básica cliente/servidor y verificar integridad de fichero.

Preparación (lab):

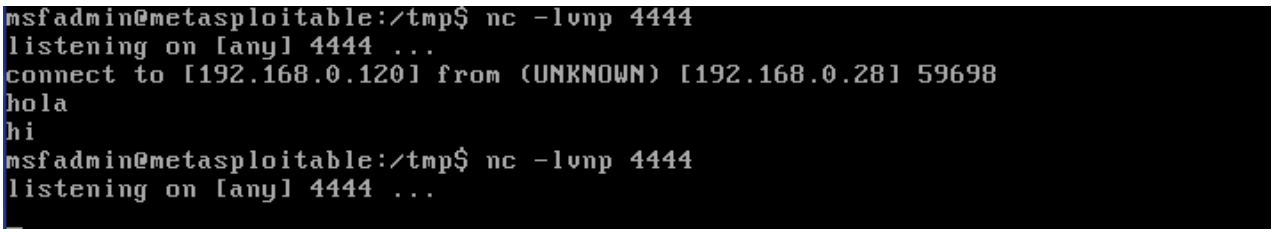
- VM-A (receptor): 192.168.56.10
- VM-B (emisor): 192.168.56.11
- prueba.txt en VM-B (pequeño texto de prueba).

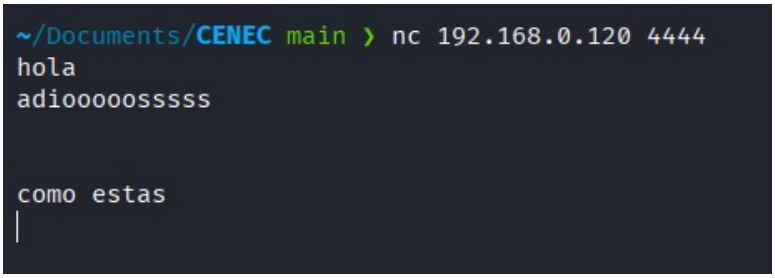
Comandos (copiar/pegar en el laboratorio):

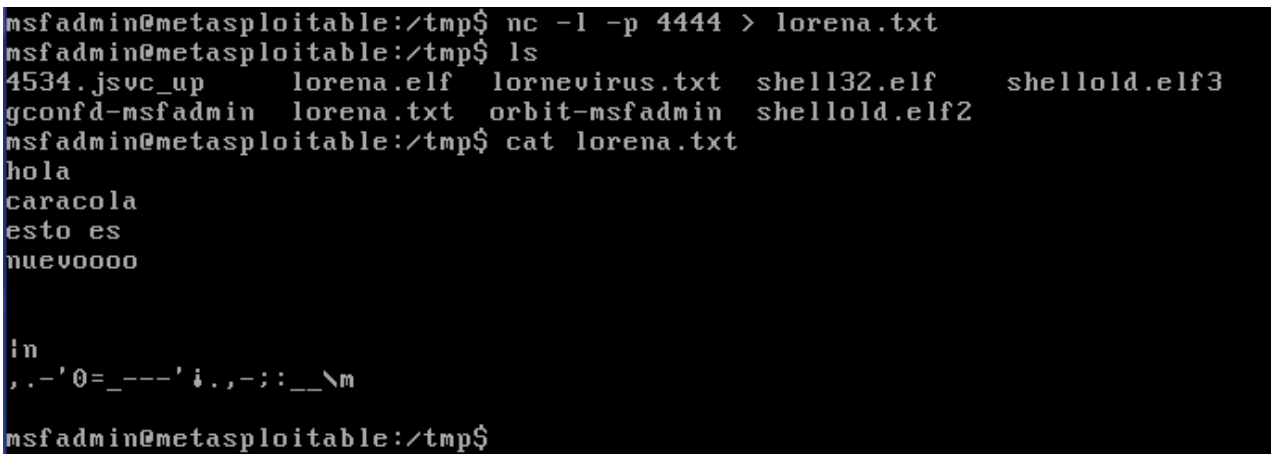
En VM-A (receptor):

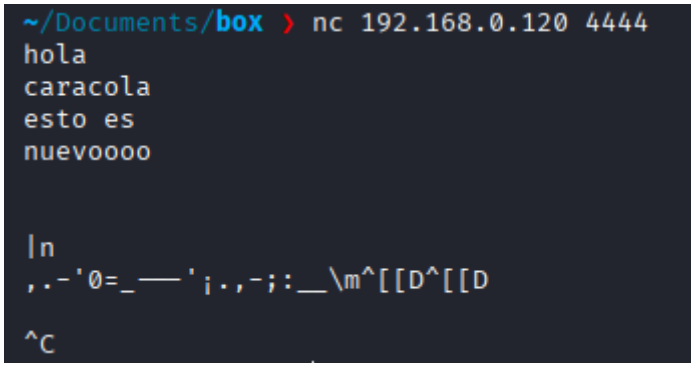
escuchar y guardar en recibido.txt

```
nc -l -p 4444 > recibido.txt
```





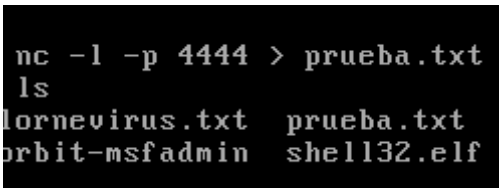


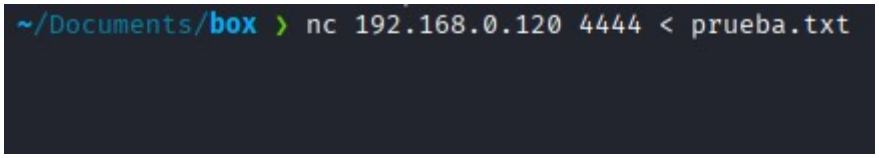


En VM-B (emisor):

enviar fichero

```
nc 192.168.56.10 4444 < prueba.txt
```



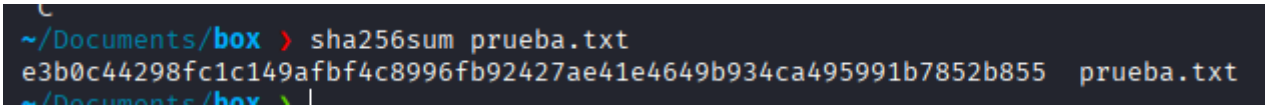


Solución / verificación (en VM-A o VM-B):

```
sha256sum prueba.txt recibido.txt
```

Salida esperada (ejemplo):

```
e3b0c44298fc1c149afbf4c8996fb924...  prueba.txt
e3b0c44298fc1c149afbf4c8996fb924...  recibido.txt
```



Si los hashes coinciden, el fichero se transfirió correctamente.

Nota pedagógica:

- Explica que nc crea una conexión TCP simple en el puerto indicado.
- Señala que si la red tiene firewalls/iptables, el puerto debe estar permitido en el laboratorio.
- Si quieres practicar recepción múltiple, usa nc -l -p 4444 > recibido_$(date +%s).txt para no sobrescribir.

---

2) nbtscan — descubrimiento NetBIOS

Objetivo: listar equipos con NetBIOS/SMB y entender la información visible.

Preparación: en la red de laboratorio tener al menos un equipo con Samba o Windows, por ejemplo 192.168.56.20.

Comando:

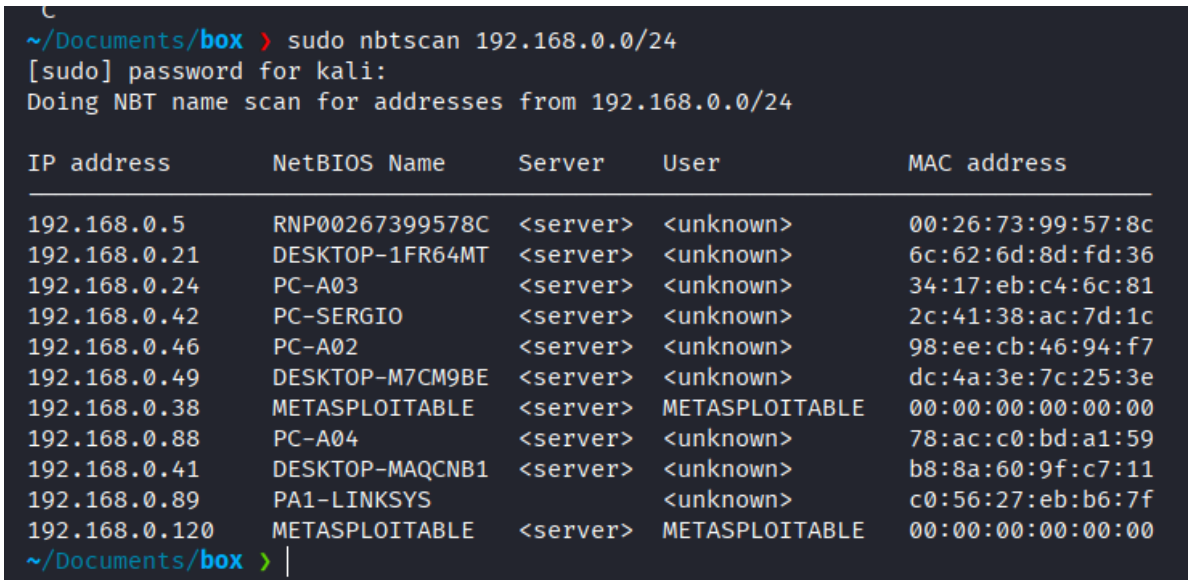
```
sudo nbtscan 192.168.56.0/24
```

Salida esperada (ejemplo):

```
192.168.56.20    WIN-VM20        WORKGROUP192.168.56.21    WINDOWS-PC      WORKGROUP
```

Interpretación (solución):

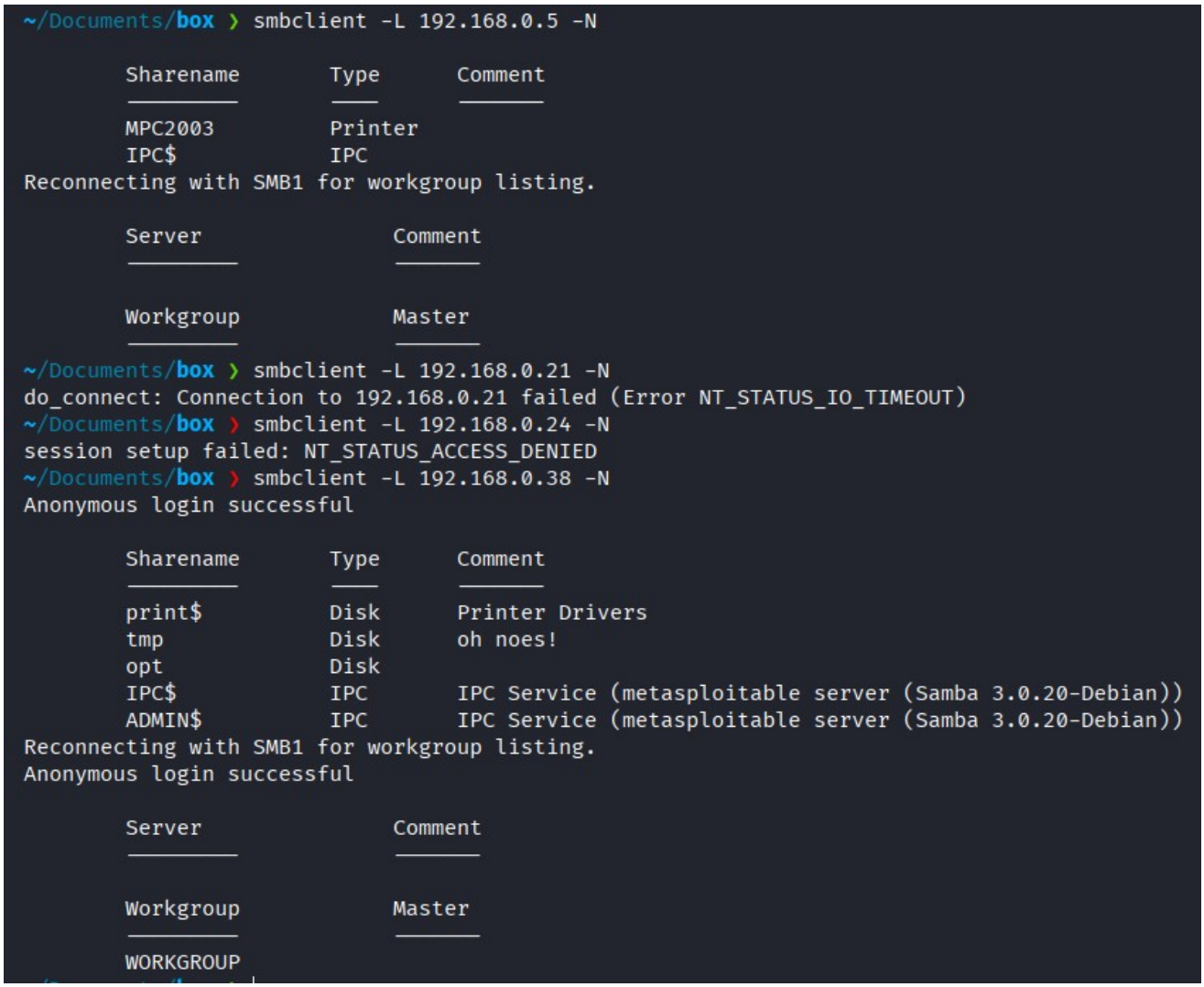
- La primera columna es la IP.
- La segunda columna es el nombre NetBIOS del equipo (identificador local).
- La tercera columna es el grupo de trabajo o dominio NetBIOS.
- Con esta info puedes identificar equipos que publican servicios Samba/Windows en la LAN.



Alternativa si nbtscan no está disponible:

```
smbclient -L 192.168.56.20 -N

nmap -p 137 --script nbstat.nse 192.168.56.0/24
```



Puntos de discusión para clase:

- NetBIOS puede filtrar metadatos que ayudan a un atacante a mapear la red.
- Medidas: deshabilitar servicios innecesarios, uso de firewalls, segmentación.

```
~/Documents/box ❯ nmap --script smb-enum-shares -p445 192.168.0.38
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-22 14:19 CEST
Nmap scan report for 192.168.0.38
Host is up (0.0021s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:E3:6A:E9 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Host script results:
| smb-enum-shares:
|   account_used: <blank>
|   \\192.168.0.38\ADMIN$:
|     Type: STYPE_IPC
|     Comment: IPC Service (metasploitable server (Samba 3.0.20-Debian))
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: <none>
|   \\192.168.0.38\IPC$:
|     Type: STYPE_IPC
|     Comment: IPC Service (metasploitable server (Samba 3.0.20-Debian))
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|   \\192.168.0.38\opt:
|     Type: STYPE_DISKTREE
|     Comment:
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: <none>
|   \\192.168.0.38\print$:
|     Type: STYPE_DISKTREE
|     Comment: Printer Drivers
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\var\lib\samba\printers
|     Anonymous access: <none>
|   \\192.168.0.38\tmp:
|     Type: STYPE_DISKTREE
|     Comment: oh noes!
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|_    Anonymous access: READ/WRITE
```

3) OpenVAS / Greenbone — escaneo básico y lectura de informe (GUI)

INSTALACIÓN DE OPENVAS EN KALI LINUX

1) Preparar el sistema

sudo apt update && sudo apt upgrade -y

sudo apt install -y gvm

Si el paquete gvm no existe en tu repositorio, prueba sudo apt install -y openvas (aunque lo recomendable en Kali moderno es gvm).

---

2) Configuración inicial (sincronizar feeds y crear certificados)

Hay un script que automatiza la puesta en marcha:

sudo gvm-setup

Ese script:

- crea certificados,
- descarga las firmas (NVTs),
- crea el usuario admin,
- arranca/activa servicios.

```
~ ❯ sudo gvm-setup
This script is provided and maintained by Debian and Kali.
 If you find any issue in this script, please report it directly to Debian or Kali

[>] Starting PostgreSQL service

[>] Creating GVM's certificate files

[>] Creating PostgreSQL database
[i] User _gvm already exists in PostgreSQL
[i] Database gvmd already exists in PostgreSQL
[i] Role DBA already exists in PostgreSQL

[*] Applying permissions
NOTICE:  role "_gvm" has already been granted membership in role "dba" by role "postgres"
GRANT ROLE
[i] Extension uuid-ossp already exists for gvmd database
[i] Extension pgcrypto already exists for gvmd database
[i] Extension pg-gvm already exists for gvmd database
[>] Migrating database
[>] Checking for GVM admin user
[*] Configure Feed Import Owner
```

---

4) Crear usuario administrador (si no lo creó gvm-setup)

sudo gvmd --create-user=admin --password='password'

(O para cambiar la contraseña existente)

sudo gvmd --user=admin --new-password='TuNuevaPass'

---

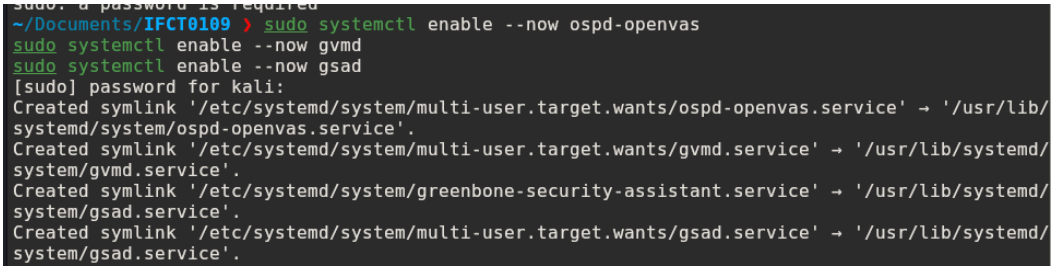
5) Iniciar y habilitar servicios (systemd)

Servicios principales: ospd-openvas (daemon), gvmd (manager), gsad (web UI). Arranca y habilita:

sudo systemctl enable --now ospd-openvas

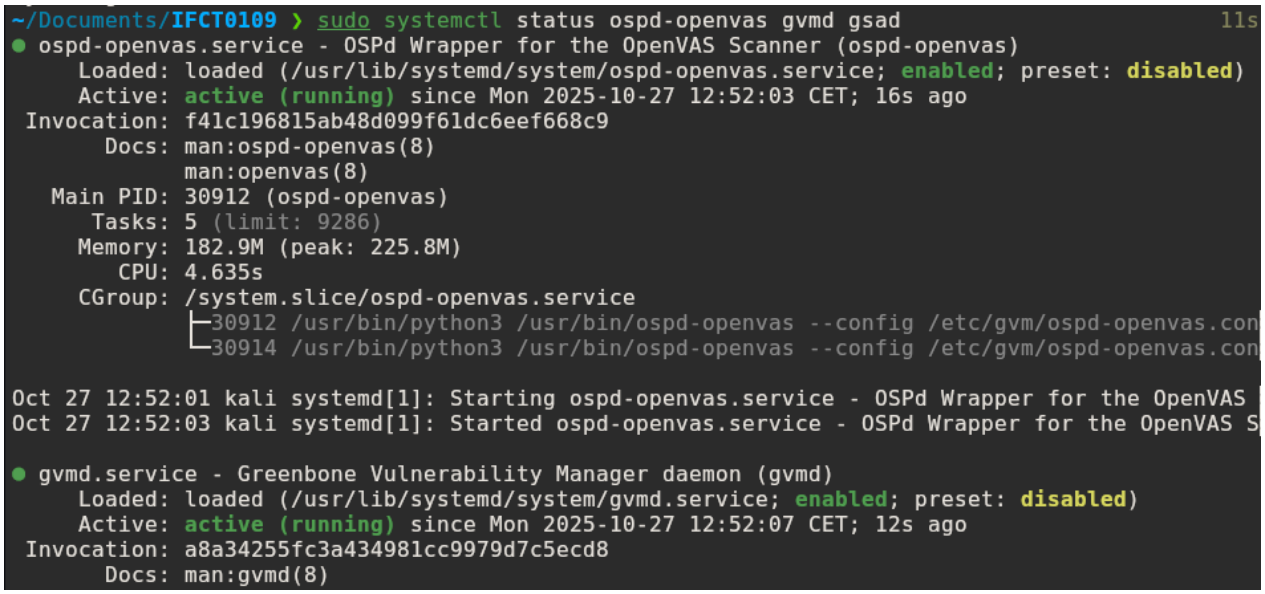
sudo systemctl enable --now gvmd

sudo systemctl enable --now gsad



Comprobar estado:

sudo systemctl status ospd-openvas gvmd gsad



---

6) Comprobar instalación

Hay una utilidad de comprobación:

sudo gvm-check-setup

```
~/Documents/IFCT0109 ❯ sudo gvm-check-setup                                    1m
gvm-check-setup 25.04.0
 This script is provided and maintained by Debian and Kali.
  Test completeness and readiness of GVM-25.04.0
Step 1: Checking OpenVAS (Scanner)...
        OK: OpenVAS Scanner is present in version 23.23.1.
        OK: Notus Scanner is present in version 22.7.2.
        OK: Server CA Certificate is present as /var/lib/gvm/CA/servercert.pem.
Checking permissions of /var/lib/openvas/gnupg/*
        OK: _gvm owns all files in /var/lib/openvas/gnupg
        OK: redis-server is present.
        OK: scanner (db_address setting) is configured properly using the redis-server sock
/var/run/redis-openvas/redis-server.sock
        OK: the mqtt_server_uri is defined in /etc/openvas/openvas.conf
        OK: _gvm owns all files in /var/lib/openvas/plugins
        OK: NVT collection in /var/lib/openvas/plugins contains 94791 NVTs.
        OK: The notus directory /var/lib/notus/products contains 506 NVTs.
Checking that the obsolete redis database has been removed
        OK: No old Redis DB
        OK: ospd-openvas service is active.
        OK: ospd-OpenVAS is present in version 22.9.0.
```

Lee la salida: te indicará cosas faltantes (por ejemplo feeds no sincronizados, permisos, certificados). Si la salida sugiere ejecutar comandos concretos, hazlo.

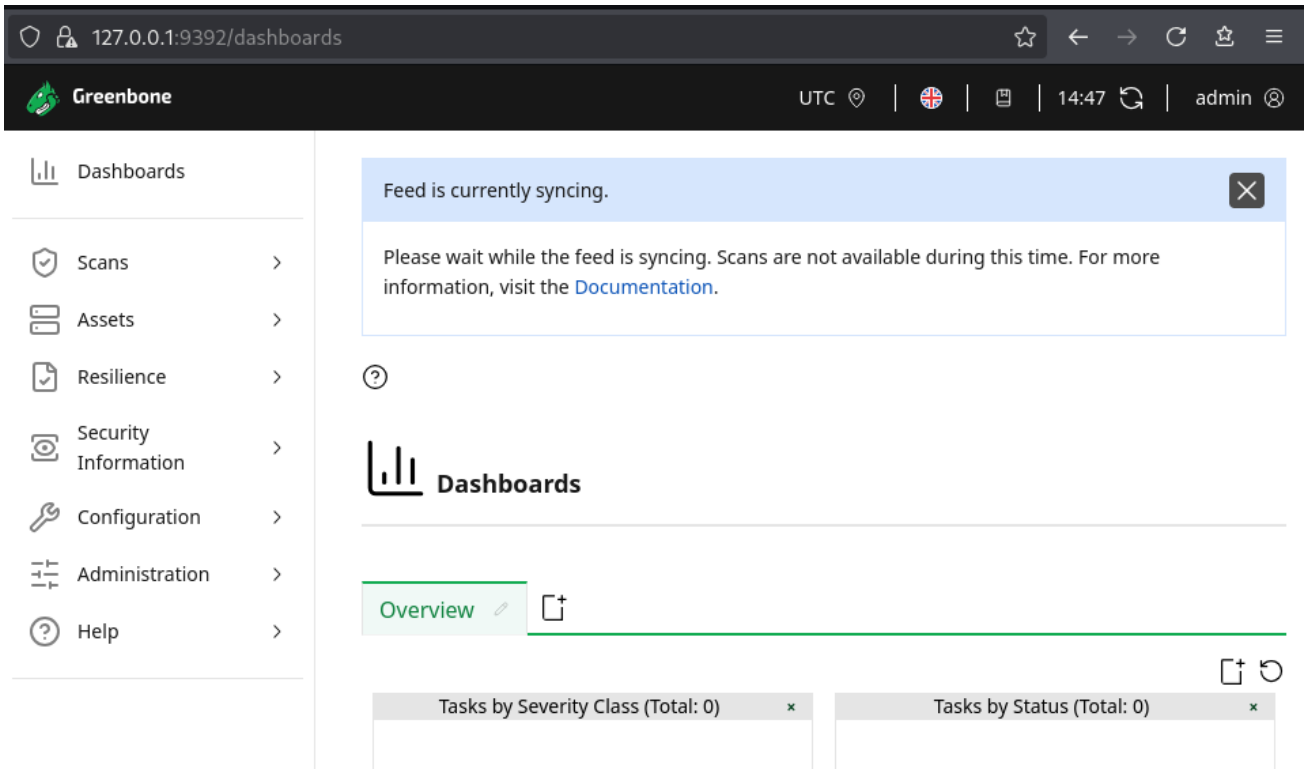
---

7) Acceder a la interfaz web

Abre en el navegador (local):

https://127.0.0.1:9392

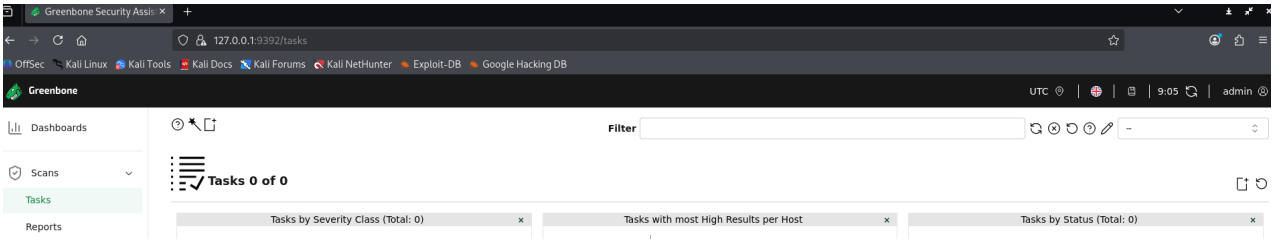
- Usuario: admin
- Contraseña: la que creaste con gvmd --create-user o la que indicó gvm-setup.



Se tiene que cambiar la clave con la siguienteinstrucción.

```
┌──(kali㉿kali)-[~]
└─$ sudo runuser -u _gvm -- gvmd --user=admin --new-password=tukey1
```

El sitio usa HTTPS con certificado auto-firmado. Acepta la excepción de seguridad en el navegador.



---

8) Actualizar manualmente después (si quieres refrescar feeds)

sudo greenbone-nvt-sync

sudo greenbone-scapdata-sync

sudo greenbone-certdata-sync

Y luego reinicia servicios si es necesario:

sudo systemctl restart ospd-openvas gvmd gsad

```
~/Documents/IFCT0109 ❯ sudo greenbone-nvt-sync
sudo greenbone-scapdata-sync
sudo greenbone-certdata-sync
Running as root. Switching to user '_gvm' and group '_gvm'.
Trying to acquire lock on /var/lib/openvas/feed-update.lock
Acquired lock on /var/lib/openvas/feed-update.lock
  Downloading Notus files from
rsync://feed.community.greenbone.net/community/vulnerability-feed/24.10/vt-data/notus/ to
/var/lib/notus
  Downloading NASL files from
rsync://feed.community.greenbone.net/community/vulnerability-feed/24.10/vt-data/nasl/ to
/var/lib/openvas/plugins
```

---

9) Logs y resolución de problemas

- Ver logs systemd:

  sudo journalctl -u ospd-openvas -f

```
~/Documents/CENEC main ❯ sudo journalctl -u ospd-openvas -f
[sudo] password for kali:
Oct 27 12:52:01 kali systemd[1]: Starting ospd-openvas.service - OSPd Wrapper for the OpenVAS S
canner (ospd-openvas)...
Oct 27 12:52:03 kali systemd[1]: Started ospd-openvas.service - OSPd Wrapper for the OpenVAS Sc
anner (ospd-openvas).
```

  sudo journalctl -u gvmd -f

  sudo journalctl -u gsad -f

```
~/Documents/CENEC main ❯ sudo journalctl -u gsad -f                                    ✘ INT
Oct 27 12:52:08 kali systemd[1]: Starting gsad.service - Greenbone Security Assistant daemon (g
sad)...
Oct 27 12:52:08 kali systemd[1]: Started gsad.service - Greenbone Security Assistant daemon (gs
ad).
Oct 27 12:52:08 kali gsad[31165]: gsad main:MESSAGE:2025-10-27 11h52.08 utc:31165: Starting GSA
D version 24.5.4~git
Oct 27 12:52:08 kali gsad[31170]: gsad main:WARNING:2025-10-27 11h52.08 utc:31170: main: start_
http_daemon redirect failed !
Oct 27 12:54:37 kali gsad[31165]: gsad gmp:MESSAGE:2025-10-27 11h54.37 utc:31165: Authenticatio
n success for 'admin' from 127.0.0.1
```

- Logs adicionales (según instalación):
  - /var/log/gvm/
  - /var/log/openvas/
    Revisa errores de permisos, fallos de sincronización o problemas con feeds.

```
~/Documents/CENEC main ❯ sudo cat /var/log/gvm/openvas.log
libgvm util:MESSAGE:2025-10-27 11h57.21 utc:31205: Updated NVT cache from version 0 to 20251027
0701
```

Ejercicio 2: Primer escaneo local

Objetivo: analizar vulnerabilidades del propio Kali.
Pasos:

- Accede a la interfaz gráfica.
- Crea un *Target* con la IP 127.0.0.1.
- Crea un *Task* (tarea) tipo “Full and Fast”.
- Lanza el escaneo y espera resultados.

Entregable:

- Captura del informe generado.
- Indica cuántas vulnerabilidades detectó (bajas, medias, altas, críticas).

🌐 Ejercicio 3: Escanear otra máquina en laboratorio

Objetivo: identificar vulnerabilidades en una máquina virtual vulnerable (p. ej., Metasploitable2 o DVWA).
Pasos:

- Añade la IP del objetivo (por ejemplo, 192.168.56.101).
- Crea una tarea con política “Full and Fast Ultimate”.
- Espera al informe y analízalo.

Entregable:

1. Captura de la lista de vulnerabilidades.
2. Describe una vulnerabilidad crítica detectada y qué servicio la causa.

⚙️ Ejercicio 4: Escaneo selectivo por puerto

Objetivo: usar OpenVAS para centrarse en un servicio específico.
Pasos:

- En el target, selecciona *Port List* → crea una personalizada (por ejemplo, solo puerto 22 o 80).
- Lanza el escaneo de ese target.

Entregable:

- Captura del resultado.
- Explica qué vulnerabilidades están relacionadas con ese servicio.

🔐 Ejercicio 5: Interpretar un CVE

Objetivo: analizar un CVE reportado.
Pasos:

- En el informe, elige una vulnerabilidad con CVE (por ejemplo, CVE-2017-0144).
- Busca en https://nvd.nist.gov o cve.mitre.org.
- Explica:
    - Qué vulnerabilidad es.

- Qué impacto tiene.
- Cómo se puede mitigar.

🧠 Ejercicio 6: Comparar políticas de escaneo

Objetivo: comprobar diferencias entre distintos niveles de análisis.
Pasos:

- Lanza dos tareas sobre el mismo objetivo:
  - “Full and Fast”
  - “Full and Very Deep”
- Compara el número de vulnerabilidades encontradas.

Entregable: tabla comparativa y conclusiones.

⚡ Ejercicio 7: Exportar y documentar informes

Objetivo: generar informes técnicos.
Pasos:

- Abre cualquier informe en la interfaz web.
- Exporta el resultado en PDF o HTML.
- Añade al documento tus observaciones.

Entregable: informe exportado + resumen de hallazgos

---

4) Nexpose / InsightVM

Instalación y primer uso de Nexpose (InsightVM) en Kali Linux

Nota: estas instrucciones están pensadas para la Community / Trial installer de Rapid7 (archivo .bin). Los nombres/URLs pueden variar según versión; si el .bin no se puede descargar, el profesor debe proporcionar el archivo.

1) Preparar Kali (actualizar + utilidades)

Abre una terminal y ejecuta:

```
sudo apt update && sudo apt upgrade -y

sudo apt install wget net-tools unzip -y
```

```
~/Documents/IFCT0109 ❯ sudo apt install wget net-tools unzip -y
[sudo] password for kali:
wget is already the newest version (1.25.0-2).
net-tools is already the newest version (2.10-2).
net-tools set to manually installed.
unzip is already the newest version (6.0-29).
unzip set to manually installed.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```

Comprueba la IP de la VM (anótala; la usarás para acceder al GUI):

```
ip a | grep -E "inet .*brd" -n
```

```
~/Documents/IFCT0109 ❯ ip a | grep -E "inet .*brd" -n
9:    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
17:    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-37885286c3f9
21:    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
~/Documents/IFCT0109 ❯
```

---

2) Descargar el instalador (ejemplo genérico)

Descarga el instalador (sustituye el nombre si el fichero es distinto)

```
wget -O Rapid7Setup-Linux64.bin
"https://download2.rapid7.com/download/InsightVM/Rapid7Setup-Linux64.bin"
```

```
~/Documents/IFCT0109 ❯ wget -O Rapid7Setup-Linux64.bin "https://download2.rapid7.com/download/I
nsightVM/Rapid7Setup-Linux64.bin"
--2025-10-28 09:33:26--  https://download2.rapid7.com/download/InsightVM/Rapid7Setup-Linux64.bi
n
Resolving download2.rapid7.com (download2.rapid7.com)... 99.84.9.73, 99.84.9.20, 99.84.9.11, ..
.
Connecting to download2.rapid7.com (download2.rapid7.com)|99.84.9.73|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2015232303 (1.9G) [application/octet-stream]
```

Hazlo ejecutable:

```
chmod +x Rapid7Setup-Linux64.bin
```

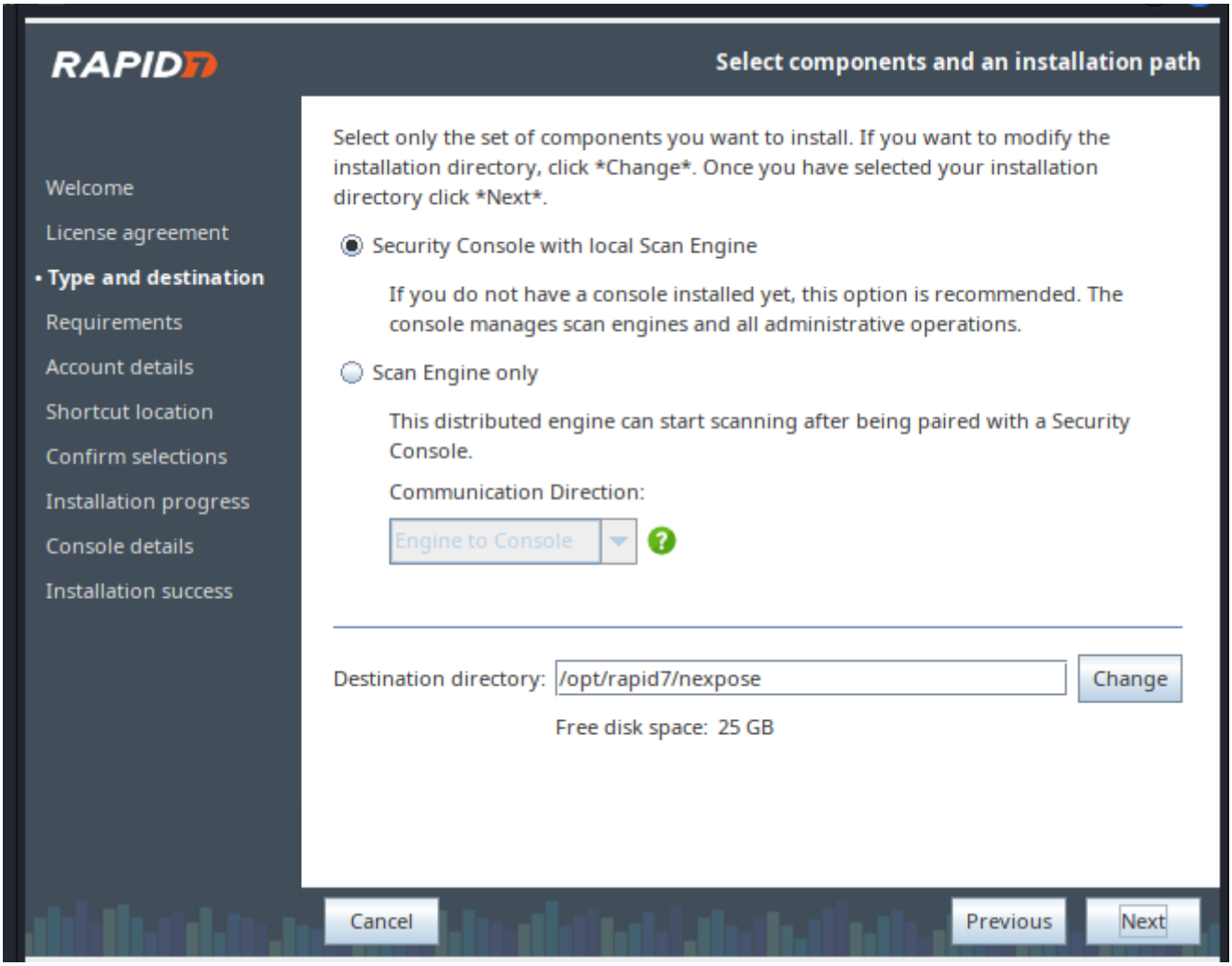
---

3) Ejecutar el instalador (modo texto / consola)

Lánzalo con sudo:

sudo ./Rapid7Setup-Linux64.bin

- **El instalador puede abrir un instalador gráfico o modo texto en terminal.**
- **Acepta EULA/Next.**
- **Instálalo en la ruta por defecto (p. ej. /opt/rapid7/nexpose).**
- **Si te pregunta por puertos, deja por defecto (UI: 3780).**



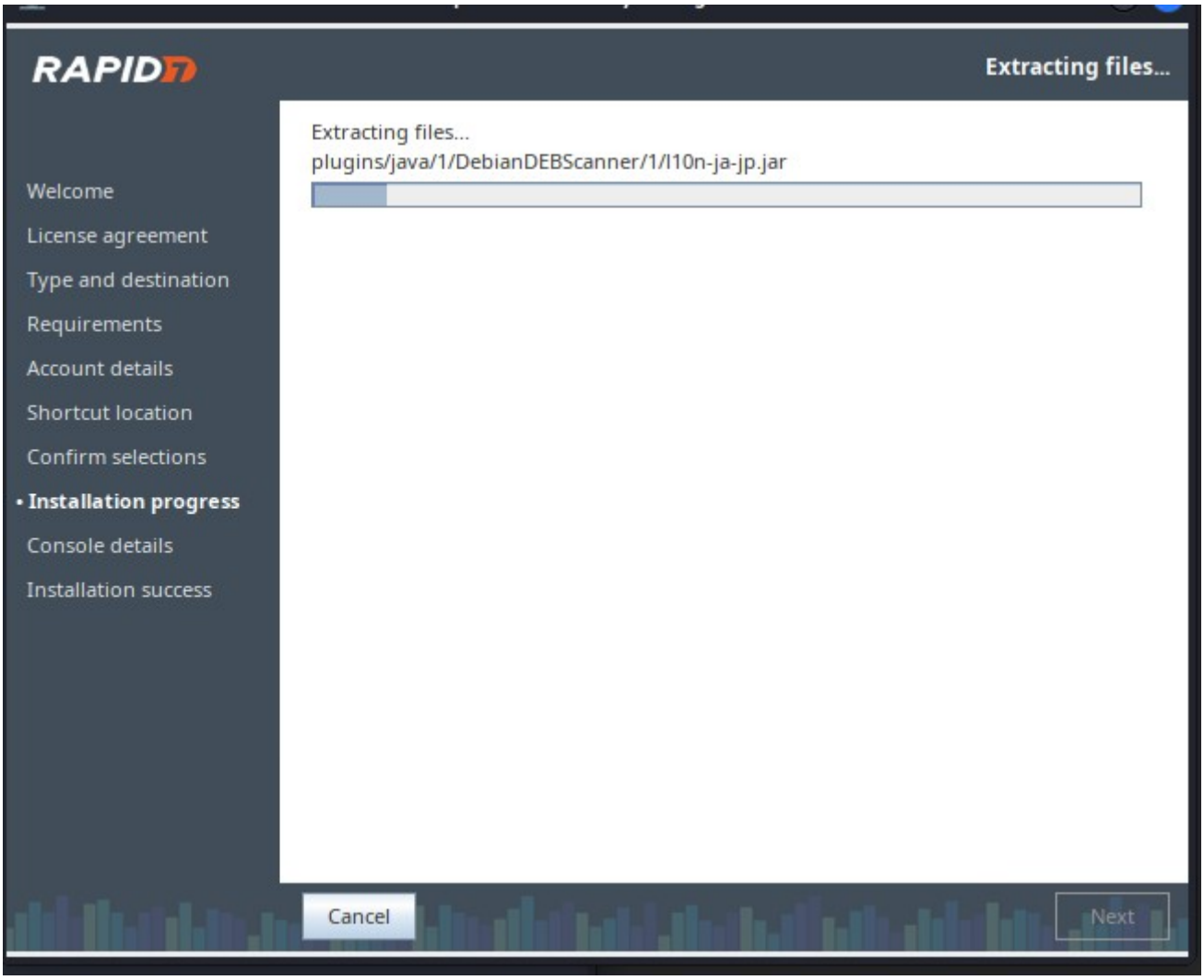
Si no ves interfaz gráfica durante la instalación (instalador gráfico necesita entorno X), el instalador suele funcionar también en modo texto. Sigue las instrucciones en pantalla.

---

4) Iniciar/gestionar servicios Nexpose

Tras la instalación, inicia el servicio (comandos típicos):

```
sudo /etc/init.d/nexposeconsole start
```

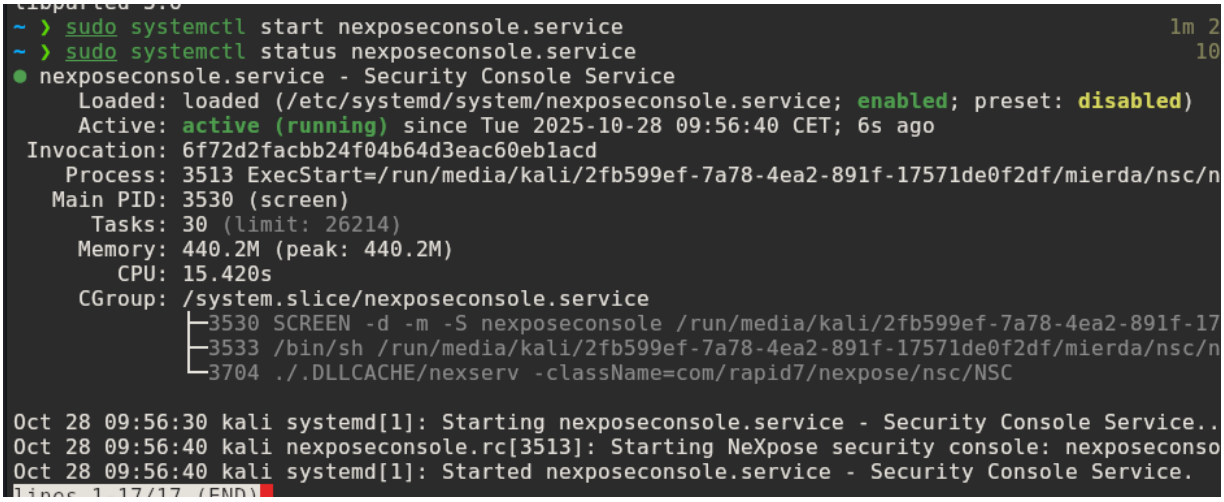


 # o (systemd)

```
sudo systemctl start nexposeconsole.service

sudo systemctl status nexposeconsole.service
```

Salida esperada: Active: active (running) o similar. Si ves errores, copia la salida para diagnóstico.



Comprueba que el puerto web está escuchando:

```
sudo ss -tulpn | grep 3780 || sudo ss -tulpn | grep nexpose
```

---

5) Abrir la interfaz web (desde Kali o cualquier máquina con acceso)

Abre un navegador en Kali (o en otra máquina de la red) e introduce:

```
https://<IP-de-la-Kali>:3780
```

- **Acepta el certificado autofirmado si aparece.**
- **Usuario por defecto suele ser nxadmin (o te pedirá crear uno durante la instalación). Usa la contraseña que hayas definido.**

Si no accede desde otra máquina, revisa firewall local:

```
sudo ufw status
```

 # si está activo y bloquea, permitir puerto:

```
sudo ufw allow 3780/tcp
```

---

6) Activar licencia Community / crear usuario

- **En el primer acceso te pedirá crear/activar la cuenta.**
- **Para Community Edition puede solicitar un activation key (gratis tras registro en Rapid7). El flujo GUI te guía: registra un correo y pega la clave en la consola.**
- **Si no puedes obtener clave en clase, pide al profesor que la proporcione.**

---

7) Crear primer Site y lanzar un escaneo (pasos GUI simples)

- **Login → menú principal.**
- **Assets → Sites → New Site**
  - **Name: Lab_192.168.56.30**
  - **Add asset: 192.168.56.30 → Add → Save.**
- **Abrir el Site → botón Scan → elegir política por defecto (ej.: *Full audit without Web Spider* o similar) → Start.**
- **Espera a que termine. Captura pantalla de Site con status Finished.**

---

8) Validaciones / comandos útiles (en terminal del equipo del grupo)

Usa estos comandos sencillos para comprobar hallazgos que veáis en el informe.

- **Comprobar SSH (versión/banners):**

```
nmap -sV -p 22 192.168.56.30
# salida ejemplo: 22/tcp open  ssh OpenSSH 7.2p2
```

- **Comprobar HTTP / server-status (mod_status):**

```
curl -I http://192.168.56.30/server-status
curl http://192.168.56.30/server-status | head -n 30
```

- **Comprobar certificado TLS (fechas):**

```
echo | openssl s_client -connect 192.168.56.30:443 -servername 192.168.56.30 2>/dev/null | openssl x509 -noout -dates
# devuelve notBefore / notAfter
```

1. **Comprobar puertos y servicios:**

```
nmap -sV 192.168.56.30
```

2. **Comprobar conectividad al servidor Nexpose (desde un cliente):**

```
ping -c 3 <IP-Nexpose>
curl -k https://<IP-Nexpose>:3780
```

---

9) Exportar informe desde la GUI

- **Tras terminar el escaneo: Reports → Generate Report → elegir formato PDF o HTML → Download.**
- **Guardad el PDF como nexpose_report_<grupo>.pdf.**

---

10) Problemas comunes y soluciones rápidas

- **Instalador no se ejecuta / falta permisos → asegúrate chmod +x y ejecuta con sudo.**
- **Instalación gráfica falla en entorno headless → ejecuta el instalador en modo texto (seguir prompts en terminal).**
- **Servicio no arranca → ver logs en /opt/rapid7/nexpose/nsc/logs/ o /var/log/ (ruta puede variar).**
- **sudo journalctl -u nexposeconsole.service -n 200**
- **No podéis activar la licencia → pedir al profesor que active o proporcione la key**

5) dsniff / tcpdump — captura pasiva (solo lectura)

Objetivo: capturar tráfico HTTP no cifrado y analizar headers visibles.

Preparación/seguridad: usar solo la red de laboratorio, páginas HTTP locales. No capturar tráfico de terceros.

Comandos (captura y análisis):

Capturar 20 segundos en interfaz eth0

```
sudo tcpdump -i eth0 -w captura_lab.pcap -c 500
```

```
~/Documents/box ❯ sudo tcpdump -i eth0 -w captura_lab.pcap -c 500
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes

^C495 packets captured
497 packets received by filter
0 packets dropped by kernel
~/Documents/box ❯ ls
captura_lab.pcap  probando2.txt  Rapid7Setup-Linux64.bin
```

Analizar HTTP en pcap con tshark

```
tshark -r captura_lab.pcap -Y http.request -T fields -e ip.src -e http.request.method -e http.host -e http.request.uri
```

```
~/Documents/box ❯ tshark -r captura_lab.pcap -Y http.request -T fields -e ip.src -e http.reques
t.method -e http.host -e http.request.uri
```

Salida de ejemplo:

```
192.168.56.11    GET    interno.lab    /pagina1.html

192.168.56.11    GET    interno.lab    /assets/logo.png
```

Interpretación (solución):

- Se ven las URLs solicitadas (host + path) y la IP origen.
- Explicar que en HTTP los headers van en texto claro; con TLS (HTTPS) estos datos están cifrados y solo se vería SNI (o encriptado si TLS1.3 sin ESNI).

Conclusión para el alumno: TLS protege el contenido y los headers; obligar HTTPS evita exposición de navegación.

---

6) ARP poisoning / MitM — versión simulada / análisis forense (NO ejecutar en redes reales)

Aviso: No proporciones ni ejecutes técnicas MitM en redes que no sean laboratorio controlado y autorizadas. Para enseñar, usa pcap de ejemplo que muestre el ataque o una red aislada preparada por el profesorado.

Actividad segura (pasos):

- Usar el fichero un mitm_capture.pcap donde en las tablas ARP aparecen respuestas ARP maliciosas.
- Los alumnos analizarán el pcap para detectar señales de ARP poisoning: múltiples ARP replies con la misma IP pero distinta MAC, o cambio súbito de la MAC asociada a la gateway.

Comandos para análisis (en la VM del alumno con el pcap):

listar paquetes ARP

```
tshark -r mitm_capture.pcap -Y arp -T fields -e frame.time -e arp.src.proto_ipv4 -e arp.src.hw_mac -e arp.dst.proto_ipv4 -e arp.dst.hw_mac
```

```
~/Documents/box ❯ tshark -r captura_lab.pcap -Y arp -T fields -e frame.time -e arp.src.proto_ipv4 -e arp.src.hw_mac -e arp.dst.proto_ipv4 -e arp.dst.hw_mac
```

```
# mostrar conversaciones IP para ver si el atacante aparece en medio (tcpdump/tshark)

tshark -r mitm_capture.pcap -q -z conv,ip
```

```
~/Documents/box ❯ tshark -r captura_lab.pcap -q -z conv,ip
================================================================================
IPv4 Conversations
Filter:<No Filter>
                                               |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                               | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
10.0.2.15            <-> 162.159.61.4             106 22 kB          93 11 kB         199 33 kB         1.749079000         1.7435
10.0.2.15            <-> 104.18.27.90              21 35 kB          21 1,907 bytes      42 37 kB         1.409591000         2.0431
10.0.2.15            <-> 23.90.190.145             21 13 kB          21 2,571 bytes      42 16 kB         1.742942000         1.1846
10.0.2.15            <-> 98.96.213.145             16 7,476 bytes      15 5,234 bytes      31 12 kB         2.424569000         1.1074
10.0.2.15            <-> 3.224.3.210               15 7,764 bytes      13 5,713 bytes      28 13 kB         2.584089000         1.2159
10.0.2.15            <-> 18.97.36.13               13 6,666 bytes      12 4,384 bytes      25 11 kB         3.493853000         0.3633
10.0.2.15            <-> 13.224.83.57              12 7,400 bytes      11 3,016 bytes      23 10 kB         2.347877000         0.0530
10.0.2.15            <-> 104.18.95.41              12 4,966 bytes      10 2,947 bytes      22 7,913 bytes     2.102297000         0.1138
10.0.2.15            <-> 163.181.49.183             9 6,568 bytes       9 2,942 bytes      18 9,510 bytes     2.518469000         0.2780
10.0.2.15            <-> 18.97.36.52                9 808 bytes       8 961 bytes      17 1,769 bytes     0.000000000         1.5142
10.0.2.15            <-> 47.246.45.185              9 2,894 bytes       8 4,115 bytes      17 7,009 bytes     1.977113000         0.4626
10.0.2.15            <-> 155.102.46.10              8 1,762 bytes       7 3,504 bytes      15 5,266 bytes     2.286867000         0.2296
194.124.205.65       <-> 10.0.2.15                  3 202 bytes       3 252 bytes       6 454 bytes      5.034381000         1.0915
188.114.97.5         <-> 10.0.2.15                  1 54 bytes        1 91 bytes        2 145 bytes      6.423552000         0.0003
================================================================================
```

Salida/ejemplo esperado (fragmento):

```
2026-04-01 10:02:12  192.168.56.1  aa:bb:cc:11:22:33  192.168.56.11  00:11:22:33:44:55
2026-04-01 10:02:13  192.168.56.1  66:77:88:99:aa:bb  192.168.56.11  00:11:22:33:44:55
```

Observa que la dirección MAC asociada a 192.168.56.1 cambia de aa:bb:cc... a 66:77:88... → indicio de poisoning.

Qué pedir al alumno (solución):

- Identificar el timestamp donde cambió la MAC de la gateway.
- Explicar por qué ip_forward=1 es necesario en un atacante real (permite que el atacante reenvíe tráfico para no provocar caída).
- Proponer mitigaciones: ARP estático en hosts críticos, DHCP snooping, port-security en switches.

---

7) dnspoof — demo de falsificación DNS (versión segura)

Objetivo: ver el efecto de una resolución DNS falsificada sin realizar spoofing en red real.

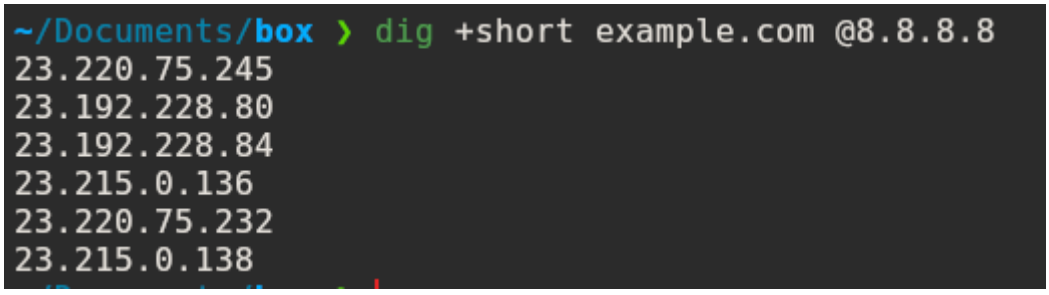
Versión segura (recomendada): editar /etc/hosts del cliente víctima en el laboratorio (o usar un archivo hosts-spoof) para simular redirección y comparar resoluciones dig antes y después.

Pasos (simulación):

- En la VM cliente, ejecutar:

```
dig +short example.com @8.8.8.8
```

guarda IP_original



- Editar /etc/hosts (necesitas permiso) y añadir:

```
192.168.56.50   example.com
```

1. Volver a consultar:

```
dig +short example.com

# ahora devuelve 192.168.56.50
```

```
~/Documents/box ❯ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

10.0.2.15   example.com
```

```
~/Documents/box ❯ dig +short example.com
23.215.0.136
23.215.0.138
23.220.75.232
23.220.75.245
23.192.228.80
23.192.228.84
```

Explicación (solución):

- Mostrar la diferencia en resolución: antes → IP original pública; después → IP del servidor controlado en lab.
- Concluir que la respuesta DNS del resolver fue suplantada por /etc/hosts local, simulando el efecto del spoofing sin riesgos.

Alternativa de análisis de pcap: proporcionar dnsspoof_capture.pcap y pedirles que extraigan la respuesta DNS falsa con tshark -r dnsspoof_capture.pcap -Y "dns.flags.rcode == 0 && dns.qry.name == \"example.com\"".

8) filesnarf — ver nombres de ficheros transferidos (metadatos)

Objetivo: mostrar que los nombres/tamaños de ficheros pueden ser visibles en transferencias no cifradas.

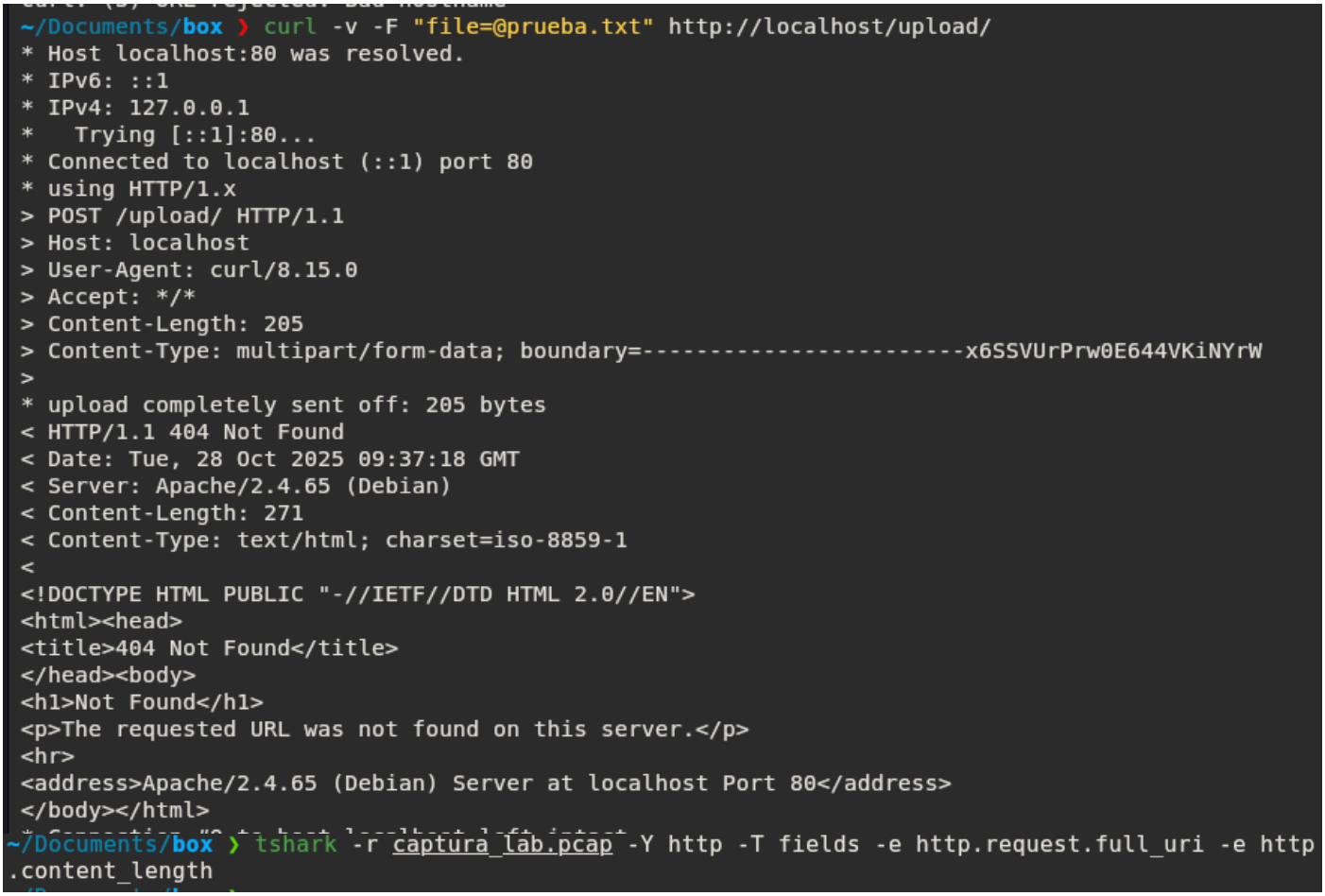
Preparación: servicio HTTP/FTP local en la red de laboratorio; subir/descargar prueba.txt.

Comando de análisis (usar pcap o filesnarf si está disponible):

```
# alternativa con tshark para extraer headers HTTP que contienen nombres

tshark -r captura_http.pcap -Y http -T fields -e http.request.full_uri -e http.content_length
```

Salida ejemplo:




```
http://192.168.56.30/upload/prueba.txt    1024
```

Interpretación (solución):

- Se ve la URL exacta y la longitud de contenido; en protocolos que no cifran metadatos (p.ej. FTP sin TLS o HTTP), estos metadatos son visibles.
- Recomendación: usar HTTPS/FTPS/SFTP para proteger el contenido y metadatos cuando sea posible.

---

9) macof — inundación de MAC (simulado)

Objetivo: mostrar el **efecto observable** (aumento de tráfico broadcast / inundación) sin necesidad de acceder a una CAM table real. Fácil de montar con 2–3 VMs en VirtualBox.

Topología (VirtualBox)

- VM **Kali-attacker** (donde se genera la inundación)
- VM **Victim** (máquina objetivo, p. ej. Ubuntu)
- (Opcional) VM **Monitor** o usar la Victim para capturas

Todas las VMs en la **misma red interna** de VirtualBox (por ejemplo, red llamada red_lab — NIC configurada como *Internal Network*).

Pasos (preparación)

- Crear snapshots de todas las máquinas.
- Asegurarse de que las VMs pueden hacer ping entre sí (p. ej. ping 192.168.56.11).
- En la VM Monitor (o Victim), preparad tcpdump/tshark:

```
# en la VM Monitor (o Victim) como root
sudo apt update
sudo apt install tcpdump tshark -y
```

Preparar fichero de captura

```
sudo tcpdump -i eth0 -w /tmp/mac_flood_before.pcap &
```

(luego ctrl+c para detener cuando corresponda)

Generar la inundación (Kali-attacker)

Tienes dos opciones: usar macof (si está disponible) o usar un script Python/Scapy (más controlable). **Recomiendo Scapy** por ser didáctico y reproducible.

Opción 1 — Scapy (recomendado)

En Kali instala scapy si hace falta:

```
sudo apt update
sudo apt install python3-scapy -y
```

Script flood_mac.py (genera tramas Ethernet con **múltiples MACs fuente aleatorias** hacia la red, provocando que el “switch virtual” vea muchas MACs nuevas):

```
#!/usr/bin/env python3
from scapy.all import Ether, sendp, RandMAC
```

```
import argparse

import time


parser = argparse.ArgumentParser()

parser.add_argument("--iface", required=True, help="Interface to send on (e.g. eth0)")

parser.add_argument("--target", required=False, help="Destination MAC (broadcast by default)", default="ff:ff:ff:ff:ff:ff")

parser.add_argument("--count", type=int, default=5000, help="Number of frames to send")

parser.add_argument("--interval", type=float, default=0.0, help="Interval between frames")

args = parser.parse_args()


for i in range(args.count):

   src = str(RandMAC())

   pkt = Ether(src=src, dst=args.target)/b'\x00'*46

   sendp(pkt, iface=args.iface, verbose=False)

   if args.interval>0:

      time.sleep(args.interval)

print("Done")
```

```
~/Documents/box ❯ vim run.sh
~/Documents/box ❯ chmod +x run.sh
~/Documents/box ❯ ./run.sh
usage: run.sh [-h] --iface IFACE [--target TARGET] [--count COUNT] [--interval INTERVAL]
run.sh: error: the following arguments are required: --iface
~/Documents/box ❯ sudo ./run.sh --iface eth0 --count 3000
```

Ejecutar (como root):

```
sudo python3 flood_mac.py --iface eth0 --count 3000
```

Ajustad --count según lo que queráis demostrar (3000 es suficiente en lab).

Opción 2 — macof (si preferís usar herramienta existente)

Instalar:

```
sudo apt install dsniff -y
```

Ejecutar (como root — **solo en lab**):

```
sudo macof -i eth0
```

(macof enviará tramas con MACs aleatorias hasta ser detenido con Ctrl+C)

Capturar y observar

En Monitor/Victim, antes y durante la inundación:

```
# iniciar captura antes

sudo tcpdump -i eth0 -w /tmp/flood_before.pcap

# detener con Ctrl+C cuando empecéis la inundación y queráis analizar "antes"


# iniciar captura durante la inundación

sudo tcpdump -i eth0 -w /tmp/flood_during.pcap

# detener con Ctrl+C tras unos segundos
```

Analizar rápido con tshark:

```
# Contar paquetes por tipo (ARP/IPv4/broadcast)

tshark -r /tmp/flood_during.pcap -q -z io,phs


# Contar tramas broadcast / multicast

tshark -r /tmp/flood_during.pcap -Y "eth.dst == ff:ff:ff:ff:ff:ff" -T fields -e frame.number | wc -l
```

```
# Ver primeras 20 tramas para comprobar src MACs distintas
tshark -r /tmp/flood_during.pcap -T fields -e eth.src | sort | uniq -c | sort -nr | head -n 30
```

Qué observar (discusión / solución)

1. Aumento notable de **tramas broadcast**/multicast en flood_during.pcap frente a flood_before.pcap.
2. Lista larga de **MACs fuente distintas** en el análisis eth.src.
3. En un switch real, si la CAM table se llena, el switch **no tiene entradas para muchas MACs** y **comienza a floodear** (reenvía tramas por todos los puertos) — en nuestro lab observaréis más tráfico broadcast en la Victim/Monitor.

10) mailsnarf — capturar correos sin TLS (solo en lab)

Objetivo: mostrar exposición de cabeceras y cuerpos si SMTP/IMAP/POP no usan TLS.

Preparación: servidor SMTP/IMAP de laboratorio sin TLS y cliente que envía correo de prueba.

Comando para capturar:

```
sudo tcpdump -i eth0 -w mail_capture.pcap port 25 or port 110 or port 143
# analizar con tshark
tshark -r mail_capture.pcap -Y smtp -V | less
```

Salida ejemplo (fragmento):

```
From: alumno@lab.local
To: profesor@lab.local
Subject: Prueba
Body: Este es un correo de ejemplo...
```

Explicación (solución):

- Se muestran cabeceras y cuerpo en texto claro.
- Contramedidas: habilitar STARTTLS/SMTPS/IMAPS/POP3S, autenticación segura, DANE/TLSA si procede.

11) urlsnarf / urlenarf — URLs visitadas

Objetivo: listar URLs visitadas desde una máquina cliente en HTTP no cifrado.

Preparación: navegar desde VM cliente a páginas HTTP en el servidor local.

Comando:

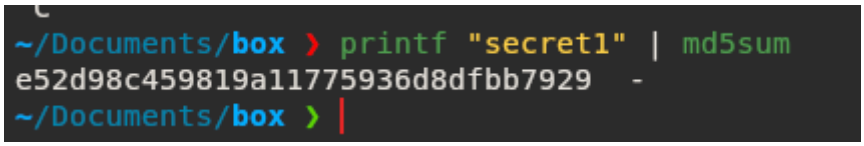
```
tshark -r captura_http.pcap -Y http.request -T fields -e ip.src -e http.host -e http.request.uri
```

o

```
sudo urlsnarf -i eth0
```

Salida ejemplo:

```
sudo urlsnarf -i eth0
```




```
192.168.56.11   interno.lab   /pagina1.html

192.168.56.11   interno.lab   /assets/logo.png
```

Interpretación (solución):

- Se puede correlacionar qué URLs visita cada IP.
- Recomendación: forzar HTTPS y HSTS en servidores.

---

12) Brutus / hydra — fuerza bruta controlada (solo cuenta de laboratorio)

---

13) RainbowCrack — mini-demo de tabla arcoíris (conceptual y local)

Objetivo: entender el poder de tablas precomputadas y por qué el *salt* las inutiliza.

Actividad práctica (local, sin usar tablas públicas):

- Generar hash MD5 de la palabra secret1:

```
printf "secret1" | md5sum
```

salida: 5ebe2294ecd0e0f08eab7690d2a6ee69

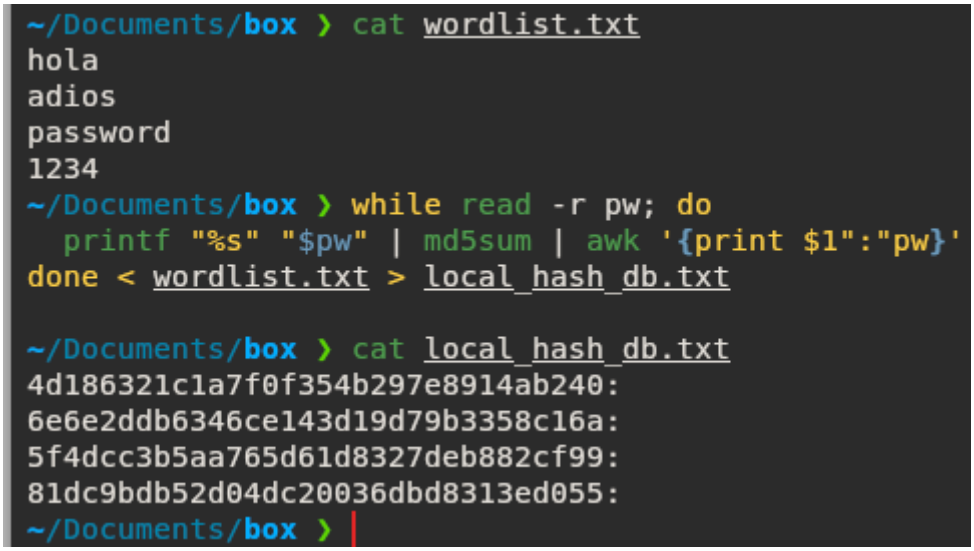
1. Crear una pequeña base local con hashes de una wordlist y buscar:

```
# crear local_hash_db.txt
while read -r pw; do
  printf "%s" "$pw" | md5sum | awk '{print $1":"pw}'
done < wordlist.txt > local_hash_db.txt


# buscar
grep "^5ebe2294ecd0e0f08eab7690d2a6ee69:" local_hash_db.txt
# salida esperada: 5ebe2294...:secret1
```

Interpretación (solución):

- Si el hash está en la base local, se recupera la contraseña.
- Añade un *salt* (por ejemplo salt+password) y vuelve a calcular: entonces el hash resultante no aparece en la tabla precomputada.
- Conclusión: usar salt + algoritmos KDF (bcrypt, Argon2, PBKDF2) aumenta el coste de ataques.



14) CrackStation / consulta de hashes (uso didáctico, offline)

Objetivo: simular consulta de hashes contra una base local.

Pasos (local):

1. Crear local_hash_db.txt como en el ejercicio anterior con wordlist pequeña.
2. Buscar el hash de password123 (482c811da5d5b4bc6d497ffa98491e38) en la DB local con grep.

Comando:

```
grep "^482c811da5d5b4bc6d497ffa98491e38:" local_hash_db.txt

# salida: 482c811...:password123
```

Explicación (solución):

2. Si la contraseña es común, aparecerá.
3. Medidas: usar hashing fuerte con salt y KDF; evitar contraseñas comunes.

GLOSARIO

Qué significa NetBIOS

NetBIOS significa **Network Basic Input/Output System**.
Es una **API (interfaz de comunicación)** creada en los años 80 para que las aplicaciones pudieran comunicarse entre ordenadores dentro de una red local (LAN).

👉 No es un protocolo de red por sí mismo, sino una **capa de servicios** que permite a los equipos de una red local:

3. Reconocerse por **nombre NetBIOS** (por ejemplo, EQUIPO01 en lugar de una IP).
4. **Compartir recursos**, como carpetas o impresoras (a través de SMB).
5. Enviar mensajes o sesiones entre ordenadores (algo que hoy casi no se usa).

¿Qué es Samba?

Samba es un **software libre** que permite que un sistema **Linux o Unix** pueda **compartir archivos e impresoras** con equipos **Windows** dentro de una red local.