RUTA CYBERSEGURIDAD
De Nivel 0 a Senior

Plan de estudio completo en 5 fases · 52 semanas · ≈2.700 horas · 3 documentos · labs, checklists y recursos.

FASE 0 CIMIENTOS | FASE 1 PRINCIPIANTE | FASE 2 INTERMEDIO | FASE 3 AVANZADO | FASE 4 SENIOR

Regla 30/50/20 — 30 % teoría, 50 % práctica, 20 % escritura. Todo el material es de uso educativo y de laboratorio autorizado.

Índice de capítulos

Navegación por documento y fase.

- **01 · ENTORNO DE LABORATORIO**
  - Prólogo
  - 0. Evalúa tu hardware antes de comprar nada
  - 1. Arquitectura recomendada: "Perfil A" (el que debes usar)
  - 2. Instalación en Windows 11 (orden recomendado)
  - 3. Configuración óptima de la VM Kali (ya la tienes instalada)
  - 4. Docker en el host: labs web en contenedores
  - 5. Máquinas de Active Directory (Fase 2+)
  - 6. Seguridad del entorno de estudio (OPSEC básico)
  - 7. Checklist de entorno (antes de empezar la Fase 1) (18 objetivos)
- **02 · ROADMAP POR ETAPAS**
  - 02 · Roadmap completo: de Nivel 0 a Senior (2024-2025)
  - FASE 0 — CIMIENTOS (Nivel 0 absoluto) (11 objetivos)
  - FASE 1 — PRINCIPIANTE / JUNIOR (Pentester junior, SOC analyst junior) (12 objetivos)
  - FASE 2 — INTERMEDIO (Mid-level Pentester / SOC Tier 2) (12 objetivos)
  - FASE 3 — AVANZADO (Senior Security Engineer / Red Team Operator / Detection Engineer) (9 objetivos)
  - FASE 4 — SENIOR / EXPERTO (Security Engineer Senior / Lead / Researcher) (8 objetivos)
  - PARTE FINAL
- **03 · PLAN DE ESTUDIO Y RECURSOS**
  - PARTE A — Plan de 52 semanas (año 1) a 15-20 h/semana
  - PARTE B — Catálogo de recursos gratuitos (con enlaces)
  - PARTE C — Cómo no abandonar (la parte que nadie cuenta)
  - PARTE D — Resumen ejecutivo (para tener a mano)

Plan de 52 semanas

15-20 h/semana · 13-15 h con la rutina base.

SEM	TRIMESTRE	FOCO	ENTREGABLE
1	TRIMESTRE 1	Fundamentos de seguridad, CIA, ATT&CK	Tabla de 5 vulnerabilidades históricas
2	TRIMESTRE 1	Redes I: OSI, TCP/IP, subnetting, CIDR, ARP	20 problemas de subnetting resueltos
3	TRIMESTRE 1	Redes II: TCP/UDP, puertos, DNS, DHCP, NAT, routing	Diagrama de tu red + dig/nslookup documentados
4	TRIMESTRE 1	Redes III: HTTP/HTTPS/TLS, proxies, firewalls, VPN	Captura Wireshark de un TLS handshake exegética
5	TRIMESTRE 1	Montar lab de 3 VMs y romper/arreglar puertos	Lab funcionando y documentado
6	TRIMESTRE 1	Linux I: FHS, permisos, SUID, usuarios, shadow	Bitácora de comandos
7	TRIMESTRE 1	Linux II: grep/awk/sed/find, systemd, cron, SSH	40 ejercicios de shell resueltos
8	TRIMESTRE 1	Linux III: bash scripting + OverTheWire Bandit 0-10	Bandit 0-10 con writeups
9	TRIMESTRE 1	Windows I: NTFS, ADS, ACLs, procesos, registro	Tabla de permisos NTFS
10	TRIMESTRE 1	Windows II: PowerShell + Event Viewer	Script PS que exporta CSV y programa una tarea
11	TRIMESTRE 1	Sysmon: instalación, config y captura de eventos	20 Event IDs catalogados
12	TRIMESTRE 1	Criptografía + CryptoHack + revisión del trimestre	CryptoHack: Classical, Encoding, Hashes
13	TRIMESTRE 2	Fundamentos web, HTML/JS, DevTools	Explicar un flujo HTTP completo
14	TRIMESTRE 2	OWASP Top 10: SSRF, SSTI, OS command injection	8 labs de PortSwigger
15	TRIMESTRE 2	SQL injection (la más importante)	10 labs + 1 writeup largo
16	TRIMESTRE 2	SQLi avanzado, NoSQL, second-order	8 labs + writeup
17	TRIMESTRE 2	Cross-Site Scripting: reflected, stored, DOM	10 labs
18	TRIMESTRE 2	CSRF + Access Control/IDOR + Broken Auth	10 labs
19	TRIMESTRE 2	Path Traversal, LFI, SSRF, Information disclosure	10 labs
20	TRIMESTRE 2	Burp Suite a fondo + OWASP ZAP	3 workflows automatizados
21	TRIMESTRE 2	Nmap a fondo + SMB, LDAP, SNMP	Informe de enumeración de tu lab
22	TRIMESTRE 2	Cracking con john y hashcat	20 hashes rotos
23	TRIMESTRE 2	Privilege escalation Linux: linpeas, SUID, sudo	3 máquinas Hack The Box Linux
24	TRIMESTRE 2	Privilege escalation Windows: tokens, UAC, servicios	2 máquinas HTB Windows + informe PDF
25	TRIMESTRE 3	PortSwigger: deserialización y request smuggling	10 labs
26	TRIMESTRE 3	PortSwigger: business logic, JWT, OAuth/OIDC	10 labs
27	TRIMESTRE 3	Informe profesional de pentest (plantilla real)	Informe PDF de 20 páginas
28	TRIMESTRE 3	Primer AD lab + BloodHound intro	AD funcionando + primer grafo
29	TRIMESTRE 3	Kerberos y NTLM a fondo: Kerberoast, AS-REP, PtH	3 writeups de ataques Kerberos
30	TRIMESTRE 3	DCSync, ACL abuse, GPO abuse, delegation	Domain Admin documentado
31	TRIMESTRE 3	ADCS ESC1-ESC8 + Certipy + detección	Writeup de ADCS
32	TRIMESTRE 3	GOAD completo de principio a fin	Writeup de 3000+ palabras
33	TRIMESTRE 3	Blue team: Sigma, Atomic Red Team, Wazuh	15 reglas Sigma probadas
34	TRIMESTRE 3	Exploit dev: stack overflow, canarios, ROP	Exploit propio funcional

SEM	TRIMESTRE	FOCO	ENTREGABLE
35	TRIMESTRE 3	Exploit dev: pwntools, bypass de mitigaciones	2º exploit propio
36	TRIMESTRE 3	Malware análisis estático: PE/ELF, capa, .NET	Informe de análisis estático
37	TRIMESTRE 4	Malware análisis dinámico: x64dbg, WinDbg, sandbox	Informe de análisis dinámico
38	TRIMESTRE 4	Cloud security: AWS/Azure + CloudGoat	Informe de pentest cloud
39	TRIMESTRE 4	Kubernetes security: kube-hunter, container escape	Cluster comprometido
40	TRIMESTRE 4	Forense avanzado: Volatility 3, NTFS, timeline	Informe forense completo
41	TRIMESTRE 4	Blue team: 25 técnicas atómicas, 20 detectadas	Dashboard + 20 reglas propias
42	TRIMESTRE 4	Purple team: simulacro de ataque completo	Informe de misses de detección
43	TRIMESTRE 4	Bug bounty: primeros reports enviados	1 finding reportado
44	TRIMESTRE 4	Portfolio: GitHub, writeups, perfil público	Perfil público funcionando
45	TRIMESTRE 4	Repaso general de Fase 2	2 máquinas HTB Pro Lab
46	TRIMESTRE 4	Elige ruta de Fase 3 (Red / Blue / Cloud)	Plan de especialización escrito
47	TRIMESTRE 4	SANS Reading Room: whitepapers clave	3 resúmenes escritos
48	TRIMESTRE 4	Escribe 2 blogs técnicos	2 blogs publicados
49	TRIMESTRE 4	Simulación red team completa de 2 semanas	Informe final de la simulación
50	TRIMESTRE 4	Revisión del año y cierre de fases	Plan del año 2
51	TRIMESTRE 4	Repaso activo: releer tus writeups	Índice de writeups actualizado
52	TRIMESTRE 4	Cierre: ruta actualizada y objetivos del año 3	Checklist anual firmado

01 · Entorno de laboratorio — hardware de referencia

> **Nota.** Esta es la versión genérica de la guía, pensada para publicarse y que la use cualquiera. Los valores concretos de la tabla de abajo son los de una máquina de referencia, no los de tu equipo: sustitúyelos por los tuyos antes de dimensionar las VMs.

0. Evalúa tu hardware antes de comprar nada

La tabla siguiente resume una **configuración de referencia** que funciona bien y que es razonablemente común en portátiles de trabajo. No es un requisito: sustituye cada fila por los valores de tu equipo y vuelve a hacer las cuentas antes de dimensionar nada.

RECURSO	VALOR DE REFERENCIA	IMPLICACIÓN TÉCNICA
CPU i7-1185G7	4C/8T, VT-x, VT-d, sin vPro	**Máximo 4 vCPU útiles** en total para VMs. Nada de Kali dentro de Kali. Con 8 núcleos o más puedes dar más vCPU por VM.
RAM 48 GB	~10-12 GB los consume el sistema anfitrión	**Presupuesto real de VMs: 28-32 GB.** Es la mayor ventaja de esta configuración. Con 16 GB el laboratorio se queda corto en cuanto levantas dos controladores de dominio.
GPU dedicada de gama baja (p. ej. T500 4 GB)	~2048 CUDA cores, sin soporte estable de Hashcat	**Hashcat en modo CPU: ~15-35 kH/s en MD5, NTLM ~30-60 kH/s.** Suficiente para CTF/HTB/contraseñas de laboratorio. Insuficiente para campañas de migración de hashes reales. **No compres GPU para esto.**
NVMe 954 GB, ~113 GB usados	~840 GB libres	Espacio no es el límite en esta configuración. El límite real es **RAND** (lectura aleatoria 4K de la VM). Un disco mecánico sí lo convierte en el cuello de botella.
Windows 11 64-bit		Híbrido Hyper-V + WSL2 + VMware/VirtualBox. Hay que elegir arquitectura.

Conclusión estratégica: un equipo con CPU de 4 núcleos y mucha RAM es un *analyst/IR* y un *pentest* machine excelente, no un *malware reversing/cracking* machine. Aprovéchalo para AD/red/web/blue-team/cloud/SIEM, que es donde está el mercado. Para reversing, una GPU de gama baja sí alcanza para malware de Windows en x86/x64 (es CPU-bound, no GPU-bound). Para cracking masivo, usa servicios cloud (hashcat en Hetzner/AWS con GPU); se explica cómo montarlo en la Fase 3.

> **Cómo adaptar esto a tu caso:** la regla práctica es *RAM antes que GPU*. Si vas a montar Active Directory multi-DC, SIEM y una VM de pentest a la vez, cada GB de RAM extra se traduce en un componente más del laboratorio. Si tu equipo tiene menos de 32 GB, baja el número de máquinas simultáneas en lugar de recortar la memoria de cada una.

1. Arquitectura recomendada: "Perfil A" (el que debes usar)

CODE0

Por qué VMware Workstation Pro y no VirtualBox

1. **Desde mayo de 2024 VMware Workstation Pro es 100 % gratuito**, también para uso comercial (Broadcom lo liberó tras la adquisición de VMware). Funcionalidad completa: snapshots, clones, folders compartidos, VIX/API, mejor rendimiento con VMDK.
2. Los VMDK con streaming y TRIM/discard funcionan mucho mejor en NVMe que los VDI de VirtualBox.
3. Snapshots en caliente más fiables.
4. VirtualBox queda como alternativa ligera si quieres una tercera VM sin tocar la instalación principal.

⚠️ Decisión crítica: Hyper-V activado o no

Windows 11 con **WSL2 y Docker Desktop necesita "Plataforma de máquina virtual"**, que es la misma capa de hipervisor que Hyper-V. Activado eso, VMware Workstation pierde rendimiento (hasta 10-20 % en E/S).

SI TU PRIORIDAD ES...	CONFIGURACIÓN
Pentesting de red/AD con muchas VMs	Desactiva Hyper-V y Docker Desktop; usa **Docker dentro de la VM Kali** solo si lo necesitas. VMware rinde al 100 %.
Blue team / SIEM / análisis con muchos contenedores	Activa Hyper-V/WMP y usa **Hyper-V Manager** o VMware con backend Hyper-V. Acepta el ~15 % de pérdida.
Equilibrio (recomendado) — Perfil A	Activa WSL2 (lo necesitas), deja VMware con **"usar hypervisor de Windows" = sí**, y limita a **1 VM pesada a la vez**. El coste es aceptable si no lanzas 4 VMs simultáneas.

Comprobación del estado actual:

CODE1
Si quieres desactivarlos y volver al máximo rendimiento de VM:

CODE2

---

2. Instalación en Windows 11 (orden recomendado)

2.1 Base del sistema (hazlo primero)

CODE0

> Si un `winget install` devuelve "No se encontró el paquete", ve a la web oficial del fabricante. `winget search` te ayuda.

2.2 Hipervisor

CODE1

2.3 WSL2 + Docker (para labs web en contenedores)

CODE2
`.wslconfig` en `C:\Users\PC\.wslconfig` — **crítico** para que Docker/WSL no te devoren la RAM:

CODE3
Aplica con: `wsl --shutdown`.

2.4 Docker: límites de memoria

Docker Desktop → Settings → Resources → Advanced → **Memory limit: 6-8 GB**, CPU limit 4. Nunca dejes Docker por defecto (por defecto usa hasta la mitad de la RAM del host).

2.5 Herramientas de seguridad en Windows (instala por grupos)

Red / análisis de tráfico

CODE4
Extra: descarga **NetCat for Windows** (`https://gitlab.com/badbits/nc.exe` o `https://github.com/int0x33/nc.exe`) y **tshark**, que viene incluido con Wireshark.

Sysinternals Suite (descarga el .zip del enlace oficial y descomprime en `C:\Tools\Sysinternals`): `https://learn.microsoft.com/sysinternals/downloads` — `Procmon`, `Autoruns`, `Process Explorer`, `TCPView`, `Sigcheck`, `Streams`, `Regshot`, `AccessChk`, `Psexec`, `PsExec` (para practice), `WinDbg`.

Malware / reversing

- **x64dbg** → `https://x64dbg.com/` (descarga el `.zip` de la release *snapshot*)
- **Ghidra** (NSA, gratis) → `https://github.com/NationalSecurityAgency/ghidra/releases` (necesita JDK 21 → ya instalado)
- **Detect It Easy (DiE)** → `https://github.com/horsicq/Detect-It-Easy/releases`
- **PE-bear** → `https://github.com/hasherezade/pe-bear/releases`
- **FLOSS (flare-floss)** → `https://github.com/mandiant/flare-floss/releases` (extrae strings ofuscados)

- **capa (Mandiant)** → `https://github.com/mandiant/capa/releases` + `https://github.com/mandiant/capa-rules`
- **PEStudio** → `https://winitor.com/` (gratis, excelente para el triaje rápido de ejecutables)
- **Any.Run sandbox** (cuenta gratuita, 60 min/semana) → `https://any.run/`
- **Triage sandbox** (gratis, sin install) → `https://tria.ge/`
- **CAPE Sandbox público** → `https://capesandbox.com/`

Web

CODE5

- **OWASP ZAP** (segundo proxy, ideal para automatizar) → `https://www.zaproxy.org/download/`
- **Nuclei** → `winget install --id ProjectDiscovery.Nuclei` (o binario en GitHub)

Forense

- **Autopsy** → `https://www.autopsy.com/download/` (Windows, ~1 GB)
- **KAPE** (triaje) → `https://github.com/KAPE/KAPE`
- **Cyber Triage** (gratis, portable) → `https://cybertriage.com/`
- **Plaso / log2timeline** → `https://github.com/log2timeline/plaso` (mejor en Docker/WSL: `docker run -v ... ghcr.io/log2timeline/plaso`)
- **Timesketch** → `https://github.com/google/timesketch` (Docker)

Blue team / SIEM local

- **Wazuh** (SIEM/XDR open source, la mejor opción para tu RAM) → instalador en `https://wazuh.com` o Docker en Ubuntu Server.
- **Elastic Security** (gratis, ~500 MB/día de ingesta) → `https://www.elastic.co/downloads`. Ten en cuenta que Kibana + Elasticsearch se comen ~6-8 GB: **no lo corras junto a 4 VMs a la vez**.
- **Splunk BOTS datasets** (SIEM datasets gratuitos para práctica) → `https://www.splunk.com/en_us/training`. No necesitas cuenta Splunk Enterprise.
- **Security Onion** → EVITAR en este portátil (necesita 32 GB+). Úsalo solo en otro equipo o en cloud.

OSINT

- **Maltego Community** → `https://www.maltego.com/downloads`
- **SpiderFoot** → `https://www.spiderfoot.net/download/`
- **theHarvester** → `https://github.com/laramies/theHarvester` (se instala en Kali)

Extras de Windows

CODE6

2.6 Exclusión de Defender (SOLO en tu máquina de laboratorio)

Windows Defender bloquea herramientas de pentest legítimas. **No lo desactives globalmente**; crea un usuario de laboratorio y aplica exclusiones **por ruta**, solo a las carpetas de herramientas:

CODE7
Documenta esta decisión en tu cuaderno: en un entorno real **jamás** se hace esto; se usan máquinas de análisis aisladas o permit-listing de hashes. Conocer la diferencia es parte de tu formación.

2.7 Extensiones de navegador (perfil de laboratorio)

Crea un **perfil de navegador nuevo y aislado** (no uses tu perfil personal):

- **FoxyProxy** o **Proxy SwitchyOmega** (para Burp/ZAP manual)
- **Wappalyzer** (fingerprinting)
- **Burp Suite Extension** (para el Chromium interno de Burp)
- **Cookie Editor** (manipulación de cookies/sesiones)
- **Disable JavaScript** (para bloquear JS en pruebas XSS)
- Desactiva el arranque con extensiones de VPN en ese perfil.

2.8 No sincronices las VMs

Nunca guardes VMs dentro de carpetas sincronizadas por OneDrive/Dropbox. OneDrive deja los `.vmdk` sparse en estado "online-only", los descarga bajo demanda y corrompe los discos de VM. Las VMs van siempre a `C:\VMs\`.

---

3. Configuración óptima de la VM Kali (ya la tienes instalada)

Si la VM Kali ya existe, verifica/aplica esta configuración. Si no la tienes, crea una nueva con estos parámetros.

3.1 Parámetros de VM (VMware Workstation Pro)

PARÁMETRO	VALOR	RAZÓN
Guest OS	**Debian 12 x64** (o "Other Linux 64-bit 3.x")	VMware no reconoce bien "Kali Linux"; declarar *Debian* da mejor rendimiento y tiempos de arranque.
vRAM	**8192 MB**	8 GB es el óptimo: 4 GB para el SO + 2 GB para Burp/navegador + 2 GB de margen. No pongas 12 GB: perjudica al host.
vCPU	**1 socket / 4 cores**	Coincide con tus 4C/8T reales. Virtualizar 8 vCPU sobre 4 físicos genera contención.
Virtualize Intel VT-x/EPT	**ON**	Necesario si más adelante montas otro hipervisor dentro de Kali (emulador de Android, o un DC anidado).
Disco	**120 GB preasignado, NVMe (0:0)**	"Preallocated" evita fragmentación; los NVMe actuales tienen espacio de sobra. "Allow to grow": OFF.
Trim/Discard	**ON**	Reclamación de bloques en NVMe. Evita que la VM crezca a 400 GB con archivos borrados.
Network adapter 1	**VMnet8 (NAT)** → Internet	Descarga de herramientas, `apt`, CTF online.
Network adapter 2	**VMnet1 (Host-only)** → `192.168.100.50/24`	Laboratorio aislado con tus otras VMs.
3D acceleration	**OFF**	Una GPU dedicada de gama baja no aporta nada dentro de la VM y genera inestabilidad.
USB controller	**3.1**	Necesario para pasar un dispositivo malicioso al sandbox.
Carpetas compartidas	**OFF** en la VM principal; crea una VM "sucia" separada con folders ON	Ver sección de OPSEC.

3.2 Configurar VMnet1 en Host-only con subnet propia

En VMware: **Edit → Virtual Network Editor → Change Settings (Admin)**.

- **VMnet1**: tipo *Host-only*, subnet `192.168.100.0`, máscara `255.255.255.0`. **Desmarca** "Use DHCP Server" si no quieres que asigne IPs (lo configuramos estático en la VM).
- **VMnet8**: deja NAT `192.168.x.0/24` (por defecto `192.168.186.0` o `192.168.203.0`).
- Añade `192.168.100.1/24` al Windows como IP estática si quieres que el host sea enrutador/lab host:

CODE0

3.3 Preparar Kali recién/desactualizada

CODE1 CODE2 CODE3 CODE4

> `NetExec` (`https://github.com/Pennyw0rth/NetExec`) es **el estándar en 2024-2025**:CrackMapExec está abandonado. Aprende NetExec primero.

CODE5 CODE6

3.4 Configuración manual de red dentro de Kali

CODE7
Añade `192.168.100.1` a `/etc/hosts`:

CODE8

3.5 Atajos y QoL en Kali

CODE9 CODE10 CODE11

3.6 Kali como "dirty box" (VM de análisis/malware)

Crea una **segunda VM Kali** de 4 GB / 2 vCPU con:

- Carpetas compartidas **OFF**
- Clipboard bidireccional **OFF** (`vmware-toolbox-cmd config clipboard.enabled false`)
- Red: **solo VMnet8 NAT** (o desconectada cuando analices muestras)
- **Nunca** la uses para pentest que requiera exfiltrar datos reales
- Es tu sandbox de reversing/forense.

3.7 Snapshots: política

CODE12
Desde la CLI de VMware:

CODE13

3.8 Backups del laboratorio

CODE14

- Guarda el **VMDK de BASE-LIMPIO comprimido** en un disco externo (VBoxManage/Vmware: exportar a OVF).
- Tus **notas y writeups** en Git (privado) o en un repo con `git` — el activo más valioso a largo plazo.

---

4. Docker en el host: labs web en contenedores

Con Docker Desktop (WSL2), lanza apps vulnerables bajo demanda (1 o 2 a la vez para no saturar):

CODE0

Acceso desde Kali VM (VMnet8 NAT): el gateway NAT de VMware suele ser `192.168.x.1`. Comprueba:

CODE1

Y desde Kali, apunta a esa IP del host, p. ej. `http://192.168.203.1`. O más limpio: usa la IP host-only `192.168.100.1`.

Cuidado con Docker en la VM Kali: evita `docker` dentro de Kali (requiere virtualización anidada y puede fallar). Usa el Docker del **host** y accede desde Kali por la red host-only.

---

5. Máquinas de Active Directory (Fase 2+)

Opción A: Windows Server Evaluation + manual (gratis, 100 % real)

1. Descarga la ISO **Windows Server 2022 Evaluation** (180 días gratis) → `https://www.microsoft.com/evalcenter/`
2. Instala 3 VMs:

- **DC01**: 4 GB / 2 vCPU → `promote to domain controller`, bosque nuevo `lab.local`, DNS + ADDS + **CA Enterprise** (necesario para labs de ADCS)
- **CA01** (opcional): 4 GB / 2 vCPU → segunda CA para ADCS abuse
- **WIN11-CLIENT**: 4 GB / 2 vCPU → Windows 11 Enterprise Evaluation (`https://www.microsoft.com/evalcenter/`)

1. Red: todas en **VMnet1 host-only** (`192.168.100.0/24`), **desconectadas de Internet**
2. Datos de ejemplo: crea usuarios, **AGEPASS (password en claro en description)**, `BackupAdmin` con grupo `Backup Operators` (historia clásica de DCSync), SPN en `MSSQLSvc` (para Kerberoasting), una cuenta de servicio con `SeImpersonatePrivilege` (para GodPotato / JuicyPotato).

Opción B: GOAD (Game of AD) — la más rápida y gratis

GOAD (Orange Cyberdefense, open source) levanta un AD vulnerable completo con Vagrant/Ansible:

CODE0

- **GOAD-Light** consume ~**8-10 GB de RAM** → **cabe en tu equipo**. Empieza aquí.
- **AD-Miner** (del mismo equipo, gratis) te audita el AD: `https://github.com/Orange-Cyberdefense/AD-Miner`
- Nota: Vagrant sobre VMware/VirtualBox necesita el plugin correcto; verifica la guía del repo.

Opción C: Snap Labs / Methodius (Azure) — solo si quieres cloud

Microsoft public labs de AD en Azure (gratis con suscripción Azure for Students / $200 de crédito) — úsalos en Fase 3.

---

6. Seguridad del entorno de estudio (OPSEC básico)

1. **Autorización primero.** Solo prueba sistemas propios o con autorización **escrita y con alcance definido**. Guarda un documento de "Reglas de Combate" (scope, horas, datos prohibidos) por engagement/lab.
2. **Nunca escanees IPs públicas desde tu IP doméstica** "a ver qué encuentras". Es un riesgo legal serio (en España, art. 264-A de la LSSI; en EE. UU., CFAA y similares). Si haces reconocimiento activo, hazlo desde una VPN de pago o Tor con salida de datacenter, y aun así solo sobre infraestructura **autorizada** (tus propios dominios, VM de pago, programas de bug bounty con scope definido).
3. **Snapshots antes que nada, y red aislada antes que desactivar el antivirus.** Cualquier muestra maliciosa se analiza en la "dirty box" offline.
4. **Cuentas burner** para registros: `33mail.com`, `guerrillamail.com`, aliases de correo.
5. **Perfil de navegador y usuario de Windows separados** para todo lo que toque seguridad.
6. **Activa BitLocker** en el disco del host si el portátil viaja.
7. **Nunca subas muestras de malware a Drive/Dropbox/OneDrive**: el cliente las escanea y las sube. Usa `bazaar.abuse.ch` y VirusTotal (subiendo solo muestras no identificadas, y nunca muestras privadas de terceros).
8. **Cifra tus backups** (restic ya lo hace) y guarda **una copia offline** de tu golden snapshot.
9. **Escribe writeups de todo.** En 6 meses tu cuaderno valdrá más que cualquier curso.

---

7. Checklist de entorno (antes de empezar la Fase 1)

- [ ] Windows 11 actualizado, sin restos de OEM innecesarios (deinstalar bloatware acelera)
- [ ] VMware Workstation Pro instalado y arranca VM
- [ ] Kali: 8 GB / 4 vCPU, dos NICs (NAT + host-only 192.168.100.50), `ip -br a` muestra ambas
- [ ] `sudo apt full-upgrade` limpio, `pipx install impacket netexec` funciona (`netexec --version`)
- [ ] `rockyou.txt` descomprimido, `seclists` instalado
- [ ] `open-vm-tools` instalado, clipboard activo en la VM *principal*
- [ ] Snapshot `BASE-LIMPIO` creado
- [ ] VMnet1 en 192.168.100.0/24; ping de Kali a 192.168.100.1 OK
- [ ] WSL2 Ubuntu instalado, `.wslconfig` limitado a 8 GB
- [ ] Docker Desktop instalado, memory limit 6-8 GB, `docker run hello-world` OK
- [ ] Burp Suite Community en Windows, arranca y abre Chromium
- [ ] Wireshark captura en la VMnet1 (captura promiscua)
- [ ] Sysinternals Suite en `C:\Tools\Sysinternals`
- [ ] Ghidra + x64dbg + Wireshark + Autopsy instalados
- [ ] VS Code con extensiones: Python, Pylance, Markdown All in One, GitLens
- [ ] Carpeta `C:\VMs\`, `C:\Tools\`, `C:\Labs\` creadas (fuera de OneDrive)
- [ ] Exclusiones de Defender aplicadas por ruta
- [ ] Bitácora de laboratorio creada: `C:\Labs\notes` + repo Git privado para writeups

02 · Roadmap completo: de Nivel 0 a Senior (2024-2025)

> **Filosofía del plan:** 30 % teoría / 70 % práctica. Cada fase termina con un entregable verificable, nunca con "he visto el vídeo". Escribe writeup de todo lo que hagas: en 12 meses tu cuaderno es tu portfolio real.
>
> **Regla de oro:** no avances de fase hasta cumplir el checklist. Saltarse la Fase 0 es la razón nº1 de-atascarse en Fase 2.

FASE 0 — CIMIENTOS (Nivel 0 absoluto)

Duración: 8-10 semanas · 120-150 horas · Edad: 0-1 años de experiencia

0.1 Objetivos medibles

1. Explicar qué es la seguridad de la información y la tríada CIA con ejemplos propios, sin leer.
2. Configurar una red doméstica virtual de 3 máquinas (VMnet1 host-only) y hacer ping entre ellas.
3. Usar la terminal de Linux con soltura: 60 comandos del día a día sin buscar en Google.
4. Leer un paquete en Wireshark y explicar qué pasa en un flujo HTTP completo (TCP handshake, GET, 200, cookies).
5. Escribir un script en Python que automatice un escaneo TCP y imprima los puertos abiertos.
6. Leer y resumir un informe de CVSS de una vulnerabilidad real (NVD).

0.2 Bloque A — Fundamentos de seguridad (semanas 1-2, ~20 h)

Teoría

- **Qué es la seguridad de la información**: confidencialidad, integridad, disponibilidad (CIA), autenticación, autorización, no repudio, rendición de cuentas.
- **Conceptos clave**: activo, amenaza, vulnerabilidad, riesgo, impacto, explotación, control/mitigación, defensa en profundidad, menor privilegio, separación de privilegios, superficie de ataque.
- **Amenazas**: malware (ransomware, RAT, stealer, wipers), phishing (spear, whaling, vishing, smishing, AiTM, ClickFix), ingeniería social, ataques de cadena de suministro, insider threat, APT, ataques a la cadena de confianza de actualizaciones.
- **Métricas y normativa** (solotz conceptual, sin examen todavía): ISO 27001, NIST CSF 2.0, CIS Controls, ENS (España), NIS2, DORA, RGPD. Entiende para qué sirve cada marco.
- **Marco MITRE ATT&CK**: qué es una matriz de tactics/techniques/groups/software. **Mapea 15 técnicas a ejemplos reales** (T1566 Phishing, T1059.001 PowerShell, T1003.001 LSASS Memory, T1021.001 RDP, T1078 Valid Accounts, T1486 Data Encrypted for Impact).
- **Historia reciente para study cases** (lee los post-mortem, no solo las noticias): WannaCry, NotPetya, SolarWinds, Log4Shell (CVE-2021-44228), Colonial Pipeline, MOVEit (CVE-2023-34362), xz-utils (CVE-2024-3094), Snowflake/UNC5537, FortiOS CVE-2024-47575, Ivanti CVE-2024-21887, Storm-2460, toolhs of Cl0p, Akira, Snowflake, Microsoft SharePoint ToolShell (CVE-2025-5528). Para cada uno responde: ¿qué falló? ¿qué control lo habría evitado? ¿qué técnica ATT&CK?

Práctica

- Crea en Notion/Obsidian una tabla: **Evento | CVE | Sector | Vector | Técnica ATT&CK | Control que lo habría evitado | Pequeño writeup de 200 palabras**.
- Completa los 10 ejercicios de la página de principios de ATT&CK (`https://attack.mitre.org/resources/`).

- Lee el resumen ejecutivo de un DBIR (Verizon DBIR 2025 / Mandiant M-Trends 2025) y extrae 5 estadísticas que te sorprendan.

Recursos

- **MITRE ATT&CK**: `https://attack.mitre.org` · Navigator: `https://mitre-attack.github.io/attack-navigator/`
- **NIST CSF 2.0** (gratis): `https://www.nist.gov/cyberframework`
- **INCIBE** (Espana, gratis, muy buen material en espanol): `https://www.incibe.es` - Guias y "Desafios" de concienciacion
- **CCN-CERT / CCN (España)**: `https://www.ccn.cni.es` — boletines y guías defensive
- **Google Cybersecurity Professional Certificate** (Coursera, ~$49/mes, puedes auditar gratis parte): `https://www.coursera.org/professional-certificates/google-cybersecurity`
- **Canal**: NetworkChuck, The Cyber Mentor (intro)
- **Libros gratis**: *Computer Security: Principles and Practice* no es gratis; usa los whitepapers de **SANS Reading Room** (`https://www.sans.org/reading-room/`) y los informes de The DFIR Report (`https://dfir.report`)

0.3 Bloque B — Redes (semanas 2-5, ~40 h) ← LA HABILIDAD MÁS CRÍTICA DEL CARRERA

Si solo puedes aprender una cosa bien, que sea esto. El 80 % de las pruebas de pentesting y el 100 % del blue team (tráfico, detección, incidente) son Red.

Teoría (en este orden exacto)

1. **Modelo OSI** (7 capas) y **modelo TCP/IP** (4 capas): qué capa hace qué, qué protocolos viven en cada una, dónde se rompe cada cosa.
2. **Capa 2**: Ethernet, tramas, MAC, ARP (ARP spoofing, ARP poisoning), STP, VLANs (y VLAN hopping), switching vs routing,NICs y teaming.
3. **Capa 3**: IPv4, IPv6, **subnetting y CIDR** (hazlo a mano hasta que sea automático), private ranges, subnetting VLSM, DHCP (rogue server, starvation), ICMP, routing estático/dinámico (OSPF, RIP, BGP — concepto), NAT/PAT, port forwarding, ARP tables.
4. **Capa 4**: TCP vs UDP, 3-way handshake, SYN flood, flags (SYN/ACK/RST/FIN/PSH/URG), **números de puerto**, `ss` / `netstat` para ver conexiones y estados, sockets.
5. **DNS**: cómo resuelve DNS de arriba abajo (root → TLD → autoritativo), tipos de registro (A, AAAA, CNAME, MX, TXT, NS, SRV, PTR), DNS recursivo vs iterativo, **DNS zone transfer (AXFR)**, DNS rebinding, DNS tunneling, registros TXT para SPF/DKIM/DMARC, cache poisoning.
6. **Capa 7 / HTTP(s)**: métodos (GET/POST/PUT/DELETE/PATCH/HEAD/OPTIONS), cabeceras (Host, Cookie, Set-Cookie, Authorization, X-Forwarded-For, CORS, CSP), cookies (HttpOnly, Secure, SameSite), códigos de estado, Content-Type, **TLS/SSL** (handshake, certificados, CA, cadena de confianza, TLS 1.2 vs 1.3), proxies inversos, balanceadores, **HTTP request smuggling** (concepto).
7. **Servicios de red comunes y sus riesgos**: SSH, Telnet, FTP, SMB (445 y NetBIOS), RDP, LDAP/LDAPS, Kerberos, SNMP, SMTP/IMAP/POP3, MySQL/MSSQL/PostgreSQL, Redis,

MongoDB, WinRM, VNC, TFTP, NFS, Modbus/ICS.

8. **Modelo de seguridad de red**: cortafuegos (stateful, L3/L4/L7, NGFW, ACLs), DMZ, VPN (IPsec, WireGuard, OpenVPN), IDS/IPS (NIDS vs HIDS, firmas vs anomalías), proxies (forward/reverse), Zero Trust Network Access.
9. **Práctica de red con herramientas** (esto es lo que se te queda):

- `ip`, `ip route`, `ss -tulpn`, `ping`, `traceroute`, `mtr`, `dig`, `nslookup`, `host`, `curl`, `wget`, `netcat` (nc -lvp, nc -e), `tcpdump`, `tshark`, `arping`, `arp -a`, `netdiscover`, `arp-scan`.
- Practica **subnetting en papel** 20 problemas.

Práctica obligatoria

1. **Configura un lab de 3 máquinas** en VMnet1: tu Kali (192.168.100.50), una VM Windows Server 2022 (192.168.100.10), una VM Ubuntu Server (192.168.100.20). Todos estáticos. Comprueba conectividad, y luego **rompe la conectividad bloqueando un puerto con `iptables` / `nft`** y arréglalo. (Esto esoor exactamente lo que harás en producción).
2. **Wireshark**: 10 capturas distintas. Para cada una, escribe qué entiendo: handshake TCP, DNS query, TLS handshake (Client Hello, Server Hello, Certificate), HTTP GET, ARP request/reply, ping (ICMP echo). Usa **filtros de display**: `tcp.flags.syn==1 && tcp.flags.ack==0`, `dns`, `tls.handshake.type==1`, `http`. Aprende a leer el resumen de conversación.
3. **Sniffing offensive**: captura tu propio tráfico SSH y FTP en claro; descifra credenciales FTP con `tshark -Y ftp`. Captura un `nc -lvp 4444` y un `nc -e /bin/sh` para entender la reverse shell a nivel de paquete.
4. **Mini-CTF de red**: 15 ejercicios en `https://overthewire.org/wargames/bandit` (nivel 0-10) — son SSH + grep/sed/find, pero te obligan a usar la terminal.
5. **Crea tu propio "cheatsheet"** de red personal (subnetting, puertos, flags, herramientas) en `C:\Labs\notes`.

Recursos

- **Kurose & Ross, "Computer Networking: A Top-Down Approach" — PDF gratis y legal del autor**: `https://gaia.cs.umass.edu/kurose_ross/index.php`. Este es EL libro. Capítulos 1-3 y 6-7.
- **The Linux Command Line** (William Shotts) — **gratis online**: `https://linuxcommand.org`
- **Wireshark User Guide** (oficial, gratis): `https://www.wireshark.org/docs/wsdg_html_chunked/`
- **PacketLife Cheat Sheets** (redes, subredes, campos): `https://packetlife.net/campus/cheat-sheets/`
- **NetSecNotes** (diagrams de protocolos): `https://www.netsecnotes.com`
- **TryHackMe: "Network Fundamentals" path** (gratis, muy bien estructurado, 2-3 h)
- **Canales**: **Professor Messer** (Network+ completo, gratis), **NetworkChuck** (redes anomalous), **David Bombal** (labs de red gratuit).
- **CCNA gratuit**: **Cisco Skills for All** `https://skillsforall.com` (curso "Introduction to Networking" gratis, labs oficiales) — incluso si no te vas a certificar, los labs son excelentes.

0.4 Bloque C — Linux y Windows (semanas 4-7, ~40 h)

Linux (Kali como SO principal de trabajo, no como adorno)

1. **Sistema de ficheros**: jerarquía `/` (`/etc`, `/var`, `/home`, `/tmp`, `/proc`, `/dev`, `/opt`, `/usr`, `/root`), FHS, montajes, enlaces simbólicos.
2. **Permisos**: `chmod` (rwx, octal), `umask`, `chown`/`chgrp`, SUID/SGID/Sticky bit (**clave para escalar privilegios en Linux**), `getfacl`/`setfacl`.
3. **Usuarios y grupos**: `/etc/passwd`, `/etc/shadow`, `/etc/group`, `useradd`, `usermod`, `passwd`, `su`/`sudo`, `visudo`.
4. **Procesos y servicios**: `ps`, `top`, `htop`, `kill`, `jobs`, `bg`/`fg`, `nohup`, `&`, **`systemctl`** (start/stop/enable/status), `journalctl`, `crontab`/systemd timers.
5. **Gestión de paquetes**: `apt` (update/upgrade/install/remove/purge), `dpkg`, `apt-cache search`, `snap` (no necesario).
6. **Red en línea de comandos**: todo lo de la sección 0.3-B9.
7. **Manipulación de texto**: `grep` (y `-r`, `-E`, `-i`, `-v`, `-n`, `-c`), `awk`, `sed`, `cut`, `sort`, `uniq`, `tr`, `xargs`, `find` (`-name`, `-type`, `-perm`, `-user`, `-size`, `-mtime`, `-exec`), `diff`, `tee`, `less` (`/`, `n`, `q`), `head`/`tail`, `column`.
8. **Archivos y compresión**: `tar` (`-czvf`, `-xzvf`), `gzip`/`gunzip`/`zip`/`unzip`, `dd` (imágenes de disco, forense), `strings`, `file`, `stat`, `xxd`, `base64`, `md5sum`/`sha256sum`.
9. **SSH**: cliente, generar claves (`ssh-keygen -t ed25519`), config en `~/.ssh/config`, túneles (`-L`, `-R`, `-D`), `scp`/`rsync`, port forwarding, hardening (`PermitRootLogin no`, `PasswordAuthentication no`), jump hosts.
10. **Bash scripting**: variables, condicionales, bucles, funciones, arrays, here-docs, parámetros, `set -euo pipefail`, manejo de errores, `trap`, input/output de ficheros, `jq` para parsear JSON.
11. **Sysinternals y Windows como sistema cliente:**

- **NTFS**: MFT, ADS (Alternate Data Streams, muy relevante para deteccion), permisos, `attrib`, `dir /r`, `streams.exe`. ADS ocultos: `echo hidden > file.txt:hidden`.
- **Procesos y servicios**: `tasklist`, `taskkill`, `Get-Process`, `Get-Service`, `sc.exe`, Process Explorer, Procmon.
- **Registro**: `reg query`, `reg add`, `regedit`, claves y valores, hives (`HKLM`, `HKCU`, `HKCR`, `Run`/`RunOnce` de persistencia), `reg save`, consultas al registro como hunting.
- **PowerShell (lo más importante de Windows hoy)**: sintaxis, cmdlets, pipelines, objetos (no cadenas), `Get-*`, `Select-Object`, `Where-Object`, `ForEach-Object`, `Invoke-Expression`, `IEX`, **descarga y ejecución en memoria** (`IEX (New-Object Net.WebClient).DownloadString( ... )`), **AMSI y su bypass conceptual**, `Get-Member`, formateo (`Format-Table`, `Out-File`, `ConvertTo-Json`), **remote execution** (`Invoke-Command`, `Enter-PSSession`, `New-PSSession`), JEA, módulos, `Import-Module`, `Get-Help`, **escritura de scripts .ps1**, firma de scripts (`ExecutionPolicy`).
- **Red en Windows**: `ipconfig`, `netstat -ano`, `route print`, `arp -a`, `nslookup`, `ping`, `Test-NetConnection`, `Get-NetTCPConnection`, `netsh`, **PowerShell TcpClient** para scripts de red.

- **Autenticación Windows**: NTLM vs Kerberos, caché de credenciales, DPAPI, SAM/SYSTEM/NTDS.dit, **WDigest**, **LSASS**, tokens de acceso (impersonation), **SeDebugPrivilege**, UAC (qué hace y por qué se bypasea).
- **Defensa**: Windows Defender (y sus exclusiones), Windows Firewall, UAC, AppLocker/WDAC/AppLocker basics, BitLocker, Event Viewer, **Sysmon** (`https://github.com/Sysinternals/Sysmon`) y sus Event IDs, **Windows Event IDs clave** (4624, 4625, 4648, 4672, 4688, 4720, 4728, 4768, 4769, 4776, 5140, 5145, 7045, 1102, 4722, 4732).

1. **Sistemas de archivos y permisos Windows**: ACLs de NTFS (`icacls`, `Get-Acl`, `AccessChk`), *inheritance*, *deny ACEs*, *share vs NTFS permissions* (diferencia crítica enpentest e IR).

Práctica obligatoria

1. **OverTheWire Bandit 0-20** completo (`https://overthewire.org/wargames/bandit/`) -> 20 dias, documentando cada solucion.
2. **Linux Upskill Challenge** (gratis, `https://linuxupskillchallenge.org`) → root. Excelente, son ~20 h de comando real.
3. **Windows**: crea un script PowerShell `Get-ADUserReport.ps1` que:

- liste todos los usuarios locales y de dominio simulado,
- exporte a CSV,
- filtre por última fecha de inicio de sesión,
- programalo con **Task Scheduler** para que corra cada noche y deje un log.

1. **Sysmon en Windows**: instala Sysmon con la config de **SwiftOnSecurity** (`https://github.com/SwiftOnSecurity/infosec-baseline` o el de Olaf `https://github.com/Neo23x0/SysmonConfig` **y el suyo), genera eventos y** captura 20 Event IDs distintos**. Entiende qué genera cada uno. Esto te servirá en blue team desde el día 1.
2. **Crea 5 cuentas de usuario** con distintos privilegios, y documenta qué puede y qué no puede hacer cada una (una tabla). Entiende UAC de forma práctica.

Recursos

- **OverTheWire**: `https://overthewire.org/wargames/` (Bandit, Natas, Krypton, Advent of Code)
- **Linux Journey** (`https://linuxjourney.com`) y **Linux Upskill Challenge** (gratis, el segundo es superior)
- **TryHackMe**: rooms "Linux Fundamentals 1/2/3", "Windows Fundamentals 1/2/3" (gratis)
- **PowerShell**: "PowerShell for Security" (Josh Atack) / el canal **The Cyber Mentor**; el libro gratuito **"PowerShell for Pentesters"** no existe legal gratis, pero el canal de **John Hammond** y los cursos de **Palo Alto** cubren lo esencial
- **Microsoft Learn** (gratis, oficial): `https://learn.microsoft.com/es-es/training/`
- **Sysinternals**: `https://learn.microsoft.com/sysinternals/`
- **Navegador NE**: **Professor Messer** (Windows Server), **NetworkChuck**

0.5 Bloque D — Scripting y Python (semanas 6-8, ~30 h)

Sin scripting no eres un pentester, eres un usuario avanzado de herramientas.

Teoría

- **Python 3**: tipos, operadores, control de flujo, funciones, listas/diccionarios/sets/tuplas, comprensión de listas, f-strings, lectura/escritura de ficheros, `os`, `subprocess`, `sys`, `argparse`, excepciones, `json`, `csv`, `re` (regex), módulos, clase/POO básica, decoradores, generadores, concurrencia (`threading`, `multiprocessing`, `asyncio` básico).
- **Bash scripting avanzado**: ya visto arriba.
- **PowerShell scripting**: ya visto arriba.
- **Conceptos de scripting para seguridad**: qué es un **payload**, un **one-liner**, un **loader**, un **stager**; cómo se abusa de herramientas de administración (LOLBAS) para ejecutar código.

Práctica obligatoria

1. **Write your own Nmap wrapper en Python**: recorta la salida XML de `nmap -oX`, parsea con `xml.etree`, e imprime una tabla de host/puerto/estado. Añade opción de guardar en JSON.
2. **SMTP/POP3 IMAP mail sniffer** de 40 líneas con `smtplib` o `socket`: captura credenciales de un servidor de prueba local.
3. **Port scanner de sockets puro** (no uses librerías de terceros): connect scan asíncrono con `socket` + `ThreadPoolExecutor`. Compara el resultado con Nmap.
4. **Web scraper / crawler** con `requests` + `beautifulsoup4`: rastrea 2 niveles de un sitio propio y extrae emails y enlaces.
5. **Generador de payloads**: script que tome un objetivo y un tipo de shell y genere el comando de reverse shell correspondiente (base64/URL/none). Aprende a decodificar los distintos esquemas de ofuscación.
6. **Script de post-explotación**: automatiza enumeración de un objetivo (usuarios, puertos, banners) y genera un informe Markdown.

Recursos

- **CS50's Introduction to Programming with Python** (Harvard, gratis, excelente): `https://cs50.harvard.edu/python/`
- **Automate the Boring Stuff with Python** (libre en línea: `https://automatetheboringstuff.com`)
- **freeCodeCamp** (cursos Python completos, gratis)
- **PortSwigger Python API / "Python for Penetration Testing"** (recursos de la comunidad)
- **PicoCTF / picoGym** (gratis, retos de Python)
- **Curso de Python de freeCodeCamp en español** disponible

0.6 Bloque E — Fundamentos de web y criptografía (semanas 7-9, ~25 h)

Web

- Cómo funciona un navegador: URL → DNS → TCP → TLS → HTTP → render.
- HTML, CSS, JavaScript (mínimo: DOM, eventos, `fetch`/XHR, localStorage, cookies, same-origin policy, CORS).
- **Conceptos de seguridad web**: XSS, CSRF, SSRF, XXE, SQLi, LFI/RFI, IDOR, deserialización, business logic flaws, broken access control, race conditions, open redirect, insecure deserialization.
- **OWASP Top 10 2021** (+ **OWASP Top 10 for LLM Applications 2025**): la referencia del sector.

Criptografía (esto se subestima y luego duele)

- **Hashing**: MD5, SHA-1, SHA-256, BLAKE2; **sales**; por qué los hashes no son cifrado; rainbow tables; **cracking offline vs online**; GPU y parallelism.
- **Cifrado simétrico**: AES, ChaCha20, ECB vs CBC vs GCM, IV/nonce, modo de operación, keystream.
- **Cifrado asimétrico**: RSA, ECC, Diffie-Hellman, firma digital, DSA/ECDSA, **confidencialidad vs integridad vs autenticidad**.
- **PKI**: CA, certificados X.509, cadena de confianza, revocación (CRL/OCSP), certificate pinning.
- **Protocolos**: TLS handshake paso a paso (ClientHello → ServerHello → Certificate → KeyExchange → Finished), **y qué falla** (servidores sin cipher moderno, sin HSTS, con certificados expirados, con DH params débiles).
- **Autenticación**: passwords + sales + KDF (PBKDF2, bcrypt, scrypt, Argon2), tokens vs session, **JWT** (estructura, algorítmo, claim, `none` algorithm, HS vs RS confusion), **OIDC/OAuth 2.0** (Authorization Code + PKCE, implicit flow peligroso, redirect_uri), **MFA/TOTP**, passkeys, WebAuthn/FIDO2, **session management** (hijacking, fixation, CSRF vs CORS vs SameSite).
- **Aplicaciones de la cripto en pentest**: attacking weak crypto, padding oracle (concepto), hash length extension, JWT attacks, TLS downgrade, **CrackMapExec/NetExec Kerberoasting** (explica la cripto detrás: TGS-REP, AES/RC4, etype), **AS-REP roasting** (no requiere cracking), **NTLM relay**.

Práctica obligatoria

1. **CryptoHack** (`https://cryptohack.org`): resuelve los bloques de "Classical Ciphers", "Encoding" y "Hashes". Completa los 5 primeros "General" (Block Ciphers).
2. **cryptopals.com**: Cryptohack es la versión interactiva. Si te animas, los 6 primeros challenges de **Set 1**.
3. **A Graduate Course in Applied Cryptography** (Dan Boneh, gratis, PDF): `https://crypto.stanford.edu/~dabo/courses/OnlineCrypto/` — lee los capítulos 1-6.
4. **TryHackMe "Crypto Crack"** y **"Room: Intro to Cryptography"** (gratis).
5. **Burp Suite: "Encryption Attacks"** labs (gratis, parte de la Web Security Academy).
6. **PortSwigger Web Security Academy**: completa los 12 servidores de **"Encryption Attacks"** y los 8 de **"Authentication"**.

0.7 PROYECTOS OBLIGATORIOS DE LA FASE 0

#	PROYECTO	ENTREGABLE	HORAS
P0.1	**Bitácora de seguridad en Git**	Repo privado con notas, comandos, capturas y writeups desde el día 1	2 h/mes
P0.2	**Red doméstica virtual de 3 VMs**	Documento con topología, IPs, tabla de puertos, 3 capturas Wireshark comentadas	8 h
P0.3	**Script de enumeración propio**	Python que escanea un rango, devuelve JSON y genera informe Markdown	6 h
P0.4	**Análisis de 5 vulnerabilidades**	5 fichas de CVEs reales: CVSS vector, impacto real, exploits públicos, control de mitigación, técnica ATT&CK	6 h
P0.5	**Infraestructura de AD de prueba**	1 Windows Server 2022 con AD DS + DNS + CA, promoting manual, documentado paso a paso	8 h
P0.6	**Breakdown de un flujo TLS**	Captura de un handshake TLS con Wireshark + Exegesis de cada campo	4 h
P0.7	**Pentest de tu propio lab web**	Levanta DVWA en Docker, encuentra 10 vulns documentadas con capturas	8 h

0.8 Certificaciones de la Fase 0 (gratis / muy baratas)

- **Fortinet NSE 1 + NSE 2 (gratis)**: `https://training.fortinet.com` -> ademas, te da vocabulario de seguridad de red.
- **Cisco Introduction to Networking (gratis)**: `https://skillsforall.com`
- **Microsoft SC-900 Security, Compliance & Identity Fundamentals**: examen **GRATIS** con Microsoft Learn; módulos gratis.
- **Google Cloud Digital Leader** o **AWS Cloud Practitioner**: gratis (examen de cortesía), good para cloud.
- **SANS Reading Room** — sin coste, material equivalente a un master.

0.9 ✅ CHECKLIST "He terminado la Fase 0 cuando..."

- [ ] Configuré una red virtual de 3 máquinas y las tengo comunicando y bloqueando puertos a voluntad.
- [ ] Sé hacer subnetting VLSM a mano sin calculadora, en menos de 2 minutos.
- [ ] Abro Wireshark y explico un handshake TCP, un DNS lookup y un TLS handshake, paquete a paquete.
- [ ] Uso 60+ comandos de Linux/Windows con soltura, sin buscar.
- [ ] Escribo un script en Python que hace algo útil y lo explico línea a línea.
- [ ] Explico la tríada CIA, un ataque real reciente y qué control lo habría evitado.
- [ ] Entiendo qué es un certificado, una CA, un hash y una sal, y por qué importa.
- [ ] Tengo el repo de notas funcionando con al menos 10 entradas.
- [ ] He completado OverTheWire Bandit 0-15 y al menos 1 de los 3 primeros CryptoHack.

- [ ] Consigo las certificaciones gratuitas de Fortinet/Cisco/Microsoft SC-900.
- [ ] Duración real: **~130 h**. Si llevas menos, no avances.

FASE 1 — PRINCIPIANTE / JUNIOR (Pentester junior, SOC analyst junior)

Duración: 4-5 meses (semanas 9-30) · 260-330 horas · 1-2 años de experiencia

En esta fase **eliges un eje principal** (Ofensivo, defensivo o web) y te conviertes en competente. Al final debes ser capaz de leading un pentest web de una app de complejidad media y escribir un informe de calidad.

1.1 Objetivos medibles

1. Enumerar y vulnerar una aplicación web completa (OWASP Top 10) de principio a fin.
2. Enumerar, crackear y escalar privilegios en una máquina Linux y en una Windows.
3. Realizar un pentest de red interna de una subnet /24 y saber moverse lateralmente.
4. Escribir un informe de pentest de 15-20 páginas con CVSS, evidencia y recomendaciones de remediación reales.
5. Escribir 20 detecciones Sigma y 10 queries KQL que funcionen de verdad.
6. Enfrentar un incidente: recoger evidencia, timeline y redactar el primer informe.

1.2 EJE OFENSIVO — Fundamentos de pentesting y exploit dev

1.2.1 Metodología yReporting (semana 9, ~8 h)

- **Fases de un pentest**: reconocimiento, enumeración, análisis de vulnerabilidades, explotación, post-explotación, informe, retrest. **Escribe siempre un "Reglas de Combate" y mantén notas en tiempo real.**
- **Tipos de pentest**: black/grey/white box, externo vs interno, ofensivo vs defensivo.
- **Informes**: escribe el primero, no el último. Secciones: Executive summary, Scope & methodology, Findings, Risk rating (CVSS v3.1/v4), Evidence (texto + imagen + request/response), Impact, Reproduction, Remediation, References, Retest. Usa **CWE** para clasificar.
- Aprende a **medir y reportar riesgo de negocio**, no solo CVSS técnico. (Es lo que te pagarán.)

1.2.2 Reconocimiento y OSINT (semanas 9-11, ~25 h)

- **OSINT pasivo**: Google dorks (`site:`, `filetype:`, `intitle:`, `inurl:`, `-site:`), Shodan/Censys, **crt.sh** (certificados = subdominios), Wayback Machine, SecurityTrails/Netlas (free tiers), Hunter.io, email harvesting, GitHub/GitLab secret scanning (`https://github.com/search?q=password+extension%3Apem&type=code`), **GitHub dorking**, cloud storage enumeration (Bucketsniper, s3scanner), metadata EXIF, Google Analytics/AdSense ID extraction.
- **OSINT activo**: subdomain enumeration con **subfinder** + **amass** + **assetfinder**, `dnsrecon`, `theHarvester`, Maltego, SpiderFoot, portscanning con Nmap.
- **OSINT en fuentes en español**: series de **Maltego CE** en la **Escuela de_OSINT**; recursos del **CN-CERT/CNI** y **INCIBE**. Canales: **S4vitar** (Android/Flutter/WhatsApp security y OSINT), **Demente0**

(explotación binaria en español).

1.2.3 Enumeración de red y servicios (semanas 10-13, ~35 h)

- **Nmap a fondo**: `-sS -sV -sC -Pn -p- --min-rate`, **NSE scripts** (`vuln`, `safe`, `default`, `http-enum`, `smb-enum`, `ldap-*`), output XML/grepable, comparación de revisiones de script (`http://nmap.org/commits/`), **fingerprinting de versiones → CVE** con `searchsploit`, `nuclei`, `wpscan`,Nikto, `feroxbuster`, `gobuster`, `ffuf`, `dirsearch`.
- **Otros**: `masscan` (rápido), `unicornscan`, `nping`, `hping3`, `zgrab2`, `snmpwalk`, `ldapsearch`, `smbclient`/`rpcclient`/`smbmap`/`enum4linux`, `smbclient -L //host -N`, `netcat` variants, `hydra` (solo lab), `medusa` (solo lab).
- **Enumerate SMB**: SMB1 vs SMB2, null sessions, anonymous logon, `NTLM` info leaks, `mS17-010` style bugs, SMB signing, SMB over TCP/NetBIOS.
- **Enumerate LDAP**: `ldapsearch -x -H ldap://IP -b "DC=lab,DC=local"`, LDAP anonymous bind, LDAP injection.
- **Enumerate SNMP**: `onesixtyone` v1/v2c, `snmpwalk -v2c -c public IP`, community strings, community bruteforce, SNMP write.
- **Enumerate DNS**: zone transfer, `dig axfr`, bruteforce de subdominios, `dnsrecon`, DNS tunneling exfil.

1.2.4 Web application pentesting (semanas 12-20, ~60 h) ← **el bloque más rentable de tu carrera**

> Recurso principal y **obligatorio**: **PortSwigger Web Security Academy** (`https://portswigger.net/web-security`) — gratis, structured, y es literalmente lo que se prueba en entrevistas y en las certificaciones (eJPT, PNPT, CPTS, OSCP web).

Orden exacto de los topics de PortSwigger (y su dificultad):

1. **Server-side request forgery (SSRF)** — novato
2. **Server-side template injection (SSTI)** — novato
3. **OS command injection** — novato
4. **SQL injection (SQLi)** — intermedio (aquí hay que invertir horas: UNION, error-based, boolean-blind, time-based, stacked queries, ORDER BY, second-order, NoSQL injection)
5. **Cross-site scripting (XSS)** — intermedio (reflected, stored, DOM, blind, context-based, CSP bypass conceptual)
6. **Cross-site request forgery (CSRF)** — intermedio
7. **Server-side request forgery / path traversal / LFI / RFI**
8. **Insecure deserialization / prototype pollution**
9. **Information disclosure** (**robots.txt, .git, backup files, source code disclosure, verbose errors, sitemap, admin panels, debug pages**)
10. **XML / XXE / XSLT injection**
11. **Authentication** (password attacks, 2FA bypass, JWT attacks, OAuth/OIDC: open redirect, **identity theft via XSS on OIDC**, auth logic flaws)

12. **Access control / IDOR / LPE** (user roles, broken access control, forced browsing)
13. **HTTP request smuggling** (CL.TE, TE.CL, obfuscation, h2c smuggling)
14. **Web cache poisoning / cache deception**
15. **Clickjacking / UI redressing**
16. **Business logic vulnerabilities** (payment tampering, race conditions, workflow bypass)
17. **Cryptographic failures** (padding oracle, AES ECB, weak JWT)

Herramientas obligatorias: **Burp Suite Community** (intercept, repeater, intruder, sequencer, repeater/decoder, logger, extensions: Logger+, Autorify, Param Miner, JWT Toolkit, InQL), **OWASP ZAP**, `ffuf`, `nuclei`, `sqlmap`, `xsser`, `dalfox`, `gobuster`, `feroxbuster`, `dirsearch`, `waybackurls`, `gf patterns` (de `tomnomnom`), `qsreplace`.

Práctica obligatoria (mínima):

- Completa **todos** los servidores de PortSwigger Academy hasta los "intermediate" en los 17 topics (≈100 labs). Mínimo **60 labs** antes de pasar a la Fase 2.
- **PortSwigger Web Security Academy — solo labs** empieza a partir de ahí.
- **DVWA** (bajo, medio, alto, imposible) → **WebGoat** (todos los niveles de las 8 categorías) → **Juice Shop** (niveles 1-5) → **Mutillidae II** → **OWASP Benchmark** (para pruebas automatizadas).
- **Bug Bounty con fines de aprendizaje**: resuelve writeups públicos de HackerOne (todo el **"Pentester's Bible"** está en `https://books.bugcrowd.com/pentesting-foreword-port-swigger-web-security-handbook`).
- Escribe **2 writeups completos** de tu una máquinas.

1.2.5 Explotación y post-explotación: Linux y Windows (semanas 14-22, ~50 h)

- **Exploit development básico**:
  - **Buffer overflow (stack)**: entendiendo los registros, el marco de pila, EIP/RIP, `ret2libc`, `ret2shellcode`, `No-DEP`, `No-NX`, ASLR, canaries/stack cookies, `checksec`, gadgets, `ROP` básico con **ROPgadget / pwntools**.
  - **Funcionamiento de la memoria**: virtual address space, `mmap`, PIE, GOT/PLT, secciones de PE/ELF, **ret2libc**, **leak → restart** para PIE bypass.
  - Practica en: **pwn.college** (gratis, ASU, cursos completos de x86-64 y reversing) `https://pwn.college`, **OverTheWire**, **Exploit Education**, **VulnHub**.
  - **OpenSecurityTraining2** (gratis, x86-64 Assembly, excelente): `https://www.ostraining.org`
- **Privilege escalation — Linux**:
  - Enumeración: `linpeas` (casi todo el trabajo), `pspy`, `sudo -l`, `find / -perm -4000`, `getcap`, cron jobs, systemd timers, writable PATH, `LD_PRELOAD`, `/etc/ld.so.preload`, Capabilities, namespaces, `ip netns`, sudo abuse, sudoedit, `SUDO_ASKPASS`, `nfs` (root_squash off), `.bashrc`/`.profile` abused, `writable /etc/passwd`, `docker.sock` present, Kubernetes service account tokens, **GTFOBins** (`https://gtfobins.github.io/` — memorízalo, es tabla de referencia pura).
  - Exploits comunes: **DirtyCow (CVE-2016-5195)**, **PwnKit (CVE-2021-4034)**, **GameOver(lay)**, **CVE-2021-22555 netfilter**, **CVE-2019-14287 sudo**, **OverlayFS**, **polkit pkexec**.

    - **Shells y pivot:** reverse shells (`bash -i`, `nc`, `python`, `socat`), shell-less (con `curl`/`wget` → payload a memoria), **pivot con** `socat`, port forwarding con `ssh -L/-R`, chisel, ligolo-ng.
- **Privilege escalation — Windows**:
    - Enumeración: `winPEASx` / `SharpUp` / `PowerUp` / `PrivescCheck`, `whoami /priv`, `net user`, `net group`, `net localgroup`, `Get-LocalUser`, `secretsdump.py` / NetExec, `BloodHound` (con SharpHound), **LOLBAS** (`https://lolbas-project.github.io/`), `PowerView`, `SharpCollection`, `SessionGopher` (WiFi), `Rubeus` (tickets), **WinPEAS** (en `C:\Tools` o en la VM Windows).
    - Mecanismos: **UAC bypass** (fodhelper, computerdefaults, eventvwr, sdclt, CMSTP, `AutoBinPath` registry), **token impersonation** (SeImpersonate → GodPotato/JuicyPotato/RoguePotato), **SeDebugPrivilege → LSASS**, **service misconfiguration** (`sc qc`, weak permissions `icacls` on services → `snake`/`SharpUp service-exe`), **unquoted service paths**, **AlwaysInstallElevated**, **DLL hijacking** (search order), **AppLocker/WDAC bypass**, **NTFS ADS abuse**, **Print Spooler** (Potato), **Windows Defender exclusions abuse**, **WinRM**, **AlwaysInstalled** COM hijack.
    - **WINDOWS post-exploitation**: Mimikatz (comprenderlo, no solo ejecutarlo) — `lsadump::hashes`, `sekurlsa::logonpasswords`, `kerberos::golden`, `kerberos::silver`, `crypto::capi`, `lsass::memssp`; **Impacket** (`secretsdump`, `psexec`, `smbexec`, `wmiexec`, `atexec`, `mimikatz` module, `ntlmrelayx`, `GetUserSPNs`, `Kerbrute`, `Get-NetUserSPNs`); **NetExec** (`netexec smb lab.local -u user -p pass`, `netexec ldap lab.local -k`, `netexec winrm -u -p -x` (shell), `-s` (smbexec), `--pth`, `--asrep`, `--mimikatz`); **Coercer** (coerción de autenticación a SMB/RPC/WinRM), **PetitPotam** (coerción a EFSRPC), **Noprint/MultiPwn**; **SharpHound/BloodHound CE** (gratuito) para mapear AD.
- **Meterpreter y Metasploit** (gratis):
    - Módulos, payloads, staging, `msfvenom` (o mejor `msfvenom` desde Kali), **Meterpreter** (`getuid`, `hashdump`, `upload`, `download`, `migrate`, `pivot`, `shell`), **Cobalt Strike** (NO es gratis; en su lugar usa **Sliver** (gratuito, `https://github.com/sliverhq/sliver`) o **Mythic** (`https://github.com/whitehatx/mythic`) — aprende C2 sin piratear)).
    - Aprende a **leer y adaptar módulos**, no solo a ejecutar `use exploit/multi/handler`.
- **Hash cracking**: `john`, `hashcat -m 0/1000/2100/ ...`, formatos de hash (`hashid`, `hashcat --example-hashes`, `hash-identifier`), wordlists/máscaras de `hashcat`, reglas, `hashcat --potfile`, crear tu propio "lab" de cracking (crea un Windows local con 3-4 hashes NTLM y atácalos).

1.2.6 Fundamentos de blue team (en paralelo, ~20 h)

- **Modelo de detección**: MITRE ATT&CK como lenguaje común.
- **Telemetría Windows**: Sysmon, Event IDs (4624/4625/4688/4697/7045/4720/4728/4732/4768/4769/4776/5140/5145), **Windows Security log**, Process creation, PowerShell ScriptBlock (4104), AMSI (4103), `Microsoft-Windows-Sysmon/Operational`.
- **Telemetría Linux**: `auditd`/`auditctl`, `syslog`, `journald`, **osquery** (muy usado en empresas: `https://osquery.io`), `audit.rules` personalizadas, `falco` (Falcón), `tetragon`.

- **SIEM y logging**: qué es un log útil, **normalización y parseo**, `CFEAT` (Common Log Format), **Sigma rules** (`https://github.com/SigmaHQ/sigma`) — escribe 15 reglas, **YARA rules** (`https://github.com/VirusTotal/yara`) — escribe 5.
- **Threat hunting**: **Atomic Red Team** (`https://github.com/redcanaryco/atomic-red-team`) — ejecuta 20 técnicas atómicas y detecta el rastro forense (ejecutar `T1059.001` y encontrar el log). **SANS Reading Room whitepapers** sobre hunting.
- **Incident response básico**: **NIST SP 800-61** (fases: preparación, detección y análisis, contención, borrado y recuperación, post-mortem), **PICERL** (Preparación, Identificación, Contención, Erradicación, Recuperación, Lecciones), cadena de custodia, diagramas de **kill chain**, **MITRE Engage**.
- **Referencias de calidad**: `https://dfir.report`, **Velociraptor** (`https://github.com/Velocidex/velociraptor`), **Splunk BOTS datasets** (`https://www.splunk.com/en_us/training`), **Blue Team Labs Online (BTLO)** (`https://blueteamlabs.online`).

1.2.7 Forense básico (en paralelo, ~15 h)

- **Conceptos**: imagen forense (dd, FTK Imager, KAPE), cadena de custodia, hashing de evidencia, **ADTs** (Abstract Data Types), sistemas de ficheros forenses (NTFS internals, journaling, MFT, $LogFile, $UsnJrnl, ADS, Alternate Data Streams, slack space), **MFT parsing**, **artefactos de registro** (UserAssist, RecentDocs, Shimcache, Amcache, BAM/DAM, Prefetch, LNK, Jump Lists, **Shimcache**).
- **Herramientas**: **Autopsy** (gratis), **Sleuth Kit**, **Volatility 3** (`https://github.com/volatilityfoundation/volatility3` — **imprescindible**), `tsk_recover`, `sleuthkit`, **KAPE** (triaje), **Plaso/log2timeline** (timeline), **Timesketch**, `bulk_extractor`, `foremost` (carving), `bulk_extractor`, `hashdeep`, `scalpel`/`foremost`, `Autopsy + Volatility` workflow.
- **Práctica**: descarga una imagen forense de Digital Corpora (`https://digitalcorpora.org`) o del NIST CFReDS (`https://cfreds.nist.gov`), haz un triaje completo, y extrae: usuarios, archivos borrados, navegador (historial/descargas), USB artifacts, registry hives, prefetch, timeline superpuesto. Escribe un informe de 3 páginas.
- **Memoria forense**: Volatility 3 con el plugin `windows.info`, `windows.pstree`, `windows.netscan`, `windows.filescan`, `windows.registry.printkey`, `windows.cmdline`, `windows.malfind`, `windows.dumpfiles`, `windows.thrdscan`, `windows.handles`.

1.3 EJE DEFENSIVO — SOC Analyst junior

Si tu inclinación es blue team/IR (muy buena elección de mercado), refuerza:

- **Fase 1.2.6** (blue team) con el doble de profundidad.
- **Plataformas**: **Wazuh** (instala en Ubuntu Server o Docker en tu host), **Elastic Security** (KQL, dashboards), **Splunk** (SPL básico: `index=`, `stats`, `eval`, `dedup`, `timechart`, `transaction`, `join`, `lookup`), **TheHive + Cortex** (open source, para gestión de incidentes), **MISP** (inteligencia de amenazas), **Shuffle** (SOAR OSS), **Velociraptor** (DFIR OS).
- **Certificaciones gratuitas**: **Fortinet NSE 1/2** (gratis), **Palo Alto Cybersecurity Academy 10x01/10x11** (gratis), **Check Point CC** (gratis), **Microsoft SC-200 (Security Operations**

Analyst) — voucher de examen gratis vía Microsoft Learn, **Google Cloud Security** path en Coursera (audit), **Splunk BOTS** datasets para práctica, **LetsDefend** (SOC analyst learning path con free tier), **Blue Team Labs Online**.

1.4 Certificaciones de la Fase 1

CERTIFICACIÓN	COSTE	CUÁNDO
PortSwigger Web Security Academy (no es cert, es Demonstrate)	Gratis	Al acabar el bloque web
TryHackMe Jr Penetration Tester path (Jr. Penetration Tester, con tarjeta)	Gratis/Suite	Al acabar la fase
eJPT (eLearnSecurity Junior Penetration Tester) — HTB exclusive, muy buen reputation/precio	~$100-250	Al acabar la fase
Google Cybersecurity Professional Certificate	~$49/mes, 4-6 meses	En paralelo
CompTIA Security+ (SY0-701)	~$400-500 (con vouchers, ~$150)	Opción si quieres "papeles"
eCPT (INE) o **CPTS (Hack The Box)**	$210-400	bordering para Fase 2

1.5 PROYECTOS OBLIGATORIOS DE LA FASE 1

#	PROYECTO	ENTREGABLE	HORAS
P1.1	**60 labs de PortSwigger Academy**	Captura + writeup de cada uno en tu repo (mínimo los "Intermediate")	60 h
P1.2	**Pentest completo de DVWA + Juice Shop**	Informe PDF de 15 páginas con CVSS, evidencias, remediación	10 h
P1.3	**2 máquinas Hack The Box (retired: Basic/Medium)**	Writeup completo por máquina (los writeups están en `https://0x00ffsec.com` o en tu repo)	12 h
P1.4	**Blue box: Windows con AD + 20 técnicas atómicas**	20 detecciones (Sigma/YARA) y una detección Atomic Red Team por técnica ATT&CK	15 h
P1.5	**Informe forense (imagen NIST/CFDReDS)**	Informe de 8 páginas: acquisition, triage, timeline, artifacts, IOC	10 h
P1.6	**Script propio de pentest** (wrapper de nmap o crawler) funcional y documentado	En tu repo	8 h
P1.7	**Pipeline SIEM local (Wazuh o Elastic)**	1 VM Ubuntu con Wazuh, 3 dashboards y 10 alertas funcionando	10 h

1.6 ✅ CHECKLIST "He terminado la Fase 1 cuando..."

- [ ] He hecho un pentest completo de una aplicacion web y he reportado 10+ vulnerabilidades con informe PDF real.
- [ ] He enumerado una red /24 completa y he comprometido las dos maquinas Windows/Linux.
- [ ] He NTT gotten root/admin en un Windows y en un Linux **sin usar un exploit de escalate de la Fase 2** (es decir, con técnicas de Fase 1).
- [ ] He resuelto **mínimo 4 máquinas de Hack The Box** (2 Windows, 2 Linux) y escrito writeups.

- [ ] He resuelto 60+ labs de PortSwigger Academy hasta nivel intermedio.
- [ ] Escribo un informe con CVSS correcto y remediación accionable.
- [ ] Tengo 20 reglas Sigma y 5 YARA que he testeado (detectan de verdad).
- [ ] He ejecutado 20+ técnicas de Atomic Red Team y he encontrado la evidencia.
- [ ] He hecho un triaje forense de una imagen real y sé explicar la cadena de custodia.
- [ ] He cracking de al menos 20 hashes reales (de un lab) con john/hashcat.
- [ ] Tengo una certificación de Fase 1 obtenida (eJPT o similar).
- [ ] Duración real: **~300 h**.

FASE 2 — INTERMEDIO (Mid-level Pentester / SOC Tier 2)

Duración: 5-7 meses (semanas 31-58) · 400-500 horas · 2-4 años de experiencia

Aquí es donde te separas del "script kiddie". Aparecen **AD profundo**, **web avanzado**, **exploits propios**, **cloud**, **detección real**.

2.1 Objetivos medibles

1. Comprometer un entorno Active Directory multi-DC y obtener **Domain Admin** (Kerberoast, AS-REP roast, DCSync, ADCS, delegation, NTDS.dit, PTH, coercion).
2. Escribir un **exploit propio** funcional (buffer overflow → shell; idealmente un bypass de mitigación o un CVE con writeup).
3. Realizar un **pentest de cloud** (AWS/Azure) que comprometa un entorno vía IAM, metadata o función serverless.
4. Configurar un **pipeline de detección** end-to-end (Sigma → SIEM) y demostrar que detecta 15 técnicas de Atomic Red Team.
5. Realizar un **incident response** completo con adquisición de imagen, timeline, IOC extraction y post-mortem.
6. Escribir un **malware analysis report** de una muestra real (Windows PE) con IOCs y YARA.

2.2 Bloque A — Active Directory profundo (la habilidad #1 de 2025)

Teoria y practica (semanas 31-38, ~80 h)

- **Objetos y atributos**: users, computers, groups (nested), OUs, GPOs, ACLs de AD, `objectClass`, `userAccountControl` (account disabled, password not required, etc.), `msDS-KeyCredentialLink` (passwordless).
- **Kerberos a fondo**:
  - KDC, TGT, TGS, AS-REQ/AS-REP, TGS-REQ/TGS-REP, **AS-REP roasting** (usuarios sin `DONT_REQUIRE_PREAUTH`), **Kerberoasting** (SPN de cualquier usuario, crack offline del TGS-REP RC4/AES), **Kerberoasting en longevidad** (cifrados a largo plazo), **Overpass-the-hash** (shadow creds), **Pass-the-ticket**, **TGS-REQ with delegation**.
  - **Kerberoasting sin crack**: hashcat `m 13100/19600/19700`, `GetUserSPNs`, `Rubeus kerberoast`, `netexec ldap lab.local --kerberoasting`.
  - **Kerberos con AES**: SHA-1 vs AES256, RC4 legacy. Configurar `msDS-SupportedEncryptionTypes` y probar.
- **NTLM y Pass-the-Hash**:
  - NTLMv1/v2, **NTLM relay** (con `ntlmrelayx` + `Responder`/`ExchangeRelay`), **relay de HTTP→SMB** (Exchange), `SealingKey`, **pbkdf2/gsecretsdump** con NTDS.dit.
  - **PtH**: `psexec.py`, `smbexec`, `wmiexec`, `impacket-psexec`, `netexec --pth`, **Rubeus ptH**, `sekurlsa::pth`.

- **DCSync** (robo de credenciales del DC): `impacket-secretsdump -dc-ip DC01 'domain/localadmin:pass'`, `netexec ldap lab.local -u user -p pass -dcsync -dc-ip DC01`, **ACL** `DS-Replication-Get-Changes` **y** `Replicating Directory Changes (All)` sobre la RootDSE, y variantes (`MS-DRSR` abuse). Es un clásico en RAG (Rogue admin like DCSync).
- **Coerción** (usada por tipose de ataques): `PetitPotam` (EFSRPC), `Coercer` (SMB/RPC/WinRM), **ExchangeRelayX** (Exchange→AD, o el nuevo **Exchange → NTLM → ADCS**).
- **Escalación de privilegios en AD**:
  - **ACL abuse**: `GenericWrite` sobre un usuario, `WriteDacl`, `GenericAll` sobre un Domain Object, **GPO abuse** (Editor del GPO), `WriteSPN` (Shadow credentials), `Owns` / `OwnsComputer`, `DCSync` ACLs, `ChildOfObject` con `GenericWrite`, `AddKeyCredentialLink`.
  - **Escalación vía Group Policy**: crear GPO malicioso, `gpedit` en el DC, Startup Script, `Computer Configuration` scripts.
  - **Escalación vía delegation**: **unconstrained delegation** (pivote a DC), **constrained delegation** (`msDS-AllowedToDelegateTo`, **Rubeus::deltatoa** / Kerberoasting de un servicio), **resource-based constrained delegation (RBCD)**, `msDS-AllowedToActOnBehalfOfOtherIdentity`.
  - **Escalación vía ADCS** (ESC1-ESC8, uno de los Primitive Attacks más críticos):
    - **ESC1**: plantilla vulnerable + enrollment habilitado para el grupo equivocado + clave de emision debil que permite falsificarla, **ESC2**: configuracion de plantilla que permiteESC any, **ESC3**: flags de plantilla, **ESC4**: permiso de lectura de CA sin restriccion, **ESC5**: plantilla vulnerable, **ESC6**: edicion de plantilla sin control de acceso, **ESC7**: template de DC, **ESC8**: enrolamiento sin verificacion de approbation.
    - Herramientas: **Certipy-AD** (`https://github.com/ly4k/Certipy`), **PKCAgent**, **ForgeCert**, **PyWhisker**, **impacket** (`get-adcs`).
    - Impacto: **DCSync desde cualquier usuario no privilegiado** a Domain Admin. Es probablemente la vía de escalación AD más usada hoy.
  - **Escalación vía Entra Connect / Sync / Passwordless** (cloud) — en Fase 3.
  - **Escalación vía Windows Server**: `SeImpersonatePrivilege`, `SeBackupPrivilege`, `SeTakeOwnershipPrivilege`, `SeRestorePrivilege`, **EfsRpc-Way** (msfssrv service), `NTFS` (vulnerable EFS paths).
- **BloodHound CE** (gratuito) como herramienta principal de mapeo:
  - Instalar BloodHound CE (ya viene con su GUI desde 2023), importar datos con SharpHound (`SharpHound.exe BloodHound -o out.zip`), técnicas de **colectores**: `Session-Enumeration`, `Group Membership`, `ACL`, `RDP`, `DCOM`, `Trusts`, `SAM`.
  - **Custom Queries**: enumerar shortcuts hacia `Domain Admin`, `ADCS` abuse, `MCSAdmin`, `Sessions to Windows Servers`, `Enumerate all sessions`, `Find all principals that can write to a given GPO`. Colección de consultas: `https://github.com/Orange-Cyberdefense/AD-Miner`, `https://github.com/BloodHoundAD/BloodHound/wiki/Useful-Queries`, `https://github.com/ZeroLegion/SharpHoundRules`.
  - **AD Miner** (gratis) genera un informe HTML completo de tus rutas de ataque: `https://github.com/Orange-Cyberdefense/AD-Miner`

- **Azure AD / Entra ID (intro)**: Azure AD Connect (sync agent abused), **Token theft** (ROADtools), `Az`, `AADInternals`, Conditional Access bypass, MFA, Application permissions, OAuth App consent phishing, `m365spray` (**Microsoft no tiene MFA por defecto → esto es crítico**), Enumeration via `MicroBurst`, `SAML` (Golden SAML, **SAML2-Fail 2025**).
- **Ejercicios de referencia**: **GOAD** (Orange Cyberdefense, gratuito) — `"https://github.com/Orange-Cyberdefense/GOAD"`, `GOAD-Light` (8-10 GB RAM → **perfecto para tu equipo**), `GOAD-3tiers` (12-15 GB), `GOAD-TheEmpire`, `GOAD-RastaLabs`, `AD-Cloud` (Azure), `HackAD` (Snap Labs, free tier), `Seaside labs` (`https://github.com/sajadalabs/SeasideLabs`).
- **Recursos adicionales de calidad**: `https://ired.team` (S4vitar's AD attack notes, **excelente**), `https://adsecurity.org`, `https://www.ired.team/blog/`, `https://github.com/Orange-Cyberdefense/GOAD`, **Orange Cyberdefense research blog**, **SANS whitepapers AD** (`https://www.sans.org/white-papers/`), **Trimarc** (blog sobre defensa AD).
- **Blue team AD**: monitorizar con Sysmon/Windows logs: `4624` (logons), `4662` (DS-Replication), `4742` (cambio de computer), `5136` (object modification), `5137` (object value set), `4741` (computer created), `4720` (user created), `4728` (group added), **deteccion de ADCS abuse** (cert issuance, `Enroll` events, `MS-CADFS`), **deteccion de Kerberoasting** (4769 con RC4), **deteccion de PtH** (4624 tipo 3 + 4768 con RC4), **deteccion de DCSync** (4662 desde cuentas de servicio), **deteccion de LDAP queries sospechosas** (5140/5156 con filtros y SamAccountName). Escribe **15 reglas** para esto.

2.3 Bloque B — Web avanzado y Exploit Development (semanas 36-45, ~80 h)

Web avanzado

- **Source code review** (SAST): leer código en PHP/Python/Node/Java, identificar **vulnerabilidades por patrón** (fuzzing de inputs, concatenación SQL, `exec`/`eval`/`system`, deserialización, path traversal en `file_get_contents`/`include`, IDOR por falta de authz check, mass assignment, business logic, race conditions en pagos, insecure direct object reference en GraphQL).
- **GraphQL**: introspección, IDOR, batching attacks, authorization bypass, `__schema`, query complexity, field-level authz.
- **OAuth/OIDC 2024-2025**: authorization code + **PKCE** obligatorio, **client_id redirect_uri validation**, **redirect_uri abuse / open redirect**, **mix-up attacks**, **OIDC con** `id_token` **validation flaws**, **nonce**, `prompt=none` con sesión existente. Herramientas: `oauth2-proxy`, `Burp AuthFlow`, `AuthLab`, `oauthX`, `evil-client`, `SAML Raider` (postura).
- **Deserialización**: Java (`ysoserial`, `Marshalsec`), PHP (`phpggc`, `Phar deserialization`), Python (`pickle`), .NET (`ysoserial.net`, `gadget` chains, `BinaryFormatter`), Node (`unserialize`, `--` `node-serialize`). Concepto de **gadget chains**.
- **Race conditions y business logic**: hacer 2 requests simultaneos a un cupon, una compra, o una transferencia, y "ganar" dos veces. Practicar con labs de PortSwigger y CTFs.
- **Prototype pollution** (Node): `__proto__`, `constructor.prototype`, gadgets hacia RCE.
- **Cache attacks**: web cache poisoning, web cache deception, CDN quirks, `Host` header attacks, **HTTP request smuggling** a fondo (h2c, TE.TE, desync, `Transfer-Encoding` obfuscation) y su

impacto en el cache.

- **Server-side template / expression language injection**: OGNL, SpEL, MVEL, Freemarker, Jinja2 SSTI.
- **JWT attacks a fondo**: `none` algorithm, HS/RS confusion, kid injection, jwk/x5u injection, weak secrets, `jwt_tool`, `hashcat -m 16500`.

Exploit Development (30-40 h)

- **Exploits de memoria a fondo**: stack buffer overflow, format string, use-after-free, integer overflow, heap exploitation (unlink, fastbin, tcache, house of force), ROP, SROP, ret2csu, **bypass de PIE/ASLR/RELRO/Canary/NX**, shellcode (y por qué falla hoy), **slab**, **GOT overwrite**.
- **Exploits de Windows**: **stack cookie** bypass, **SEH** chains, **DEP/ASLR bypass**, **BadGet** (Egg-hunting), **Unicorn Engine** (emulación de `ntdll!LdrLoadDll`), `NtCreateThreadEx` shellcode, **Windows pwn (pwntools + WSL + Windows)**, `WARP` de fuzzing (`winafl`, `WinAFL` — fuzzing de apps Windows), `libfuzzer` / `AFL++` con `wine`, `boofuzz` para protocolos de red.
- **Práctica seria**: **pwn.college** (gratis, curso completo de "Binary Exploitation" de ASU, como 500 labs de dificultad progresiva). **Exploit Education**, **Narnia**, **pwnable challenges** de OverTheWire, **pwnchallenge**.
- **Fuzzing**: concepts, AFL++, libFuzzer, coverage-guided fuzzing, `radamsa` (mutation), `boofuzz` (network protocol), `Peach` ;ARGET. Practica fuzzing un parser (ej., un parser de JSON/URL personalizado) y encuentra un crash.

2.4 Bloque C — Cloud Security (semanas 40-48, ~60 h)

- **Conceptos cloud**: modelos de responsabilidad compartida, IAM en profundidad, **principio de menor privilegio en la nube**, multi-tenancy,Regions/AZs, **CloudTrail / Azure Monitor**, **billing as an attack surface**, serverless, PaaS, containers, **Kubernetes** (pod security, RBAC, network policies, service accounts, secrets, `kubectl` exposure, `container escape`, `hostPath` mounts).
- **AWS**: IAM policies, roles, trust policies, `AssumeRole`, **Metadata Service SSRF** (IMDSv1 vs IMDSv2, 169.254.169.254), S3 bucket misconfigurations (ACL, public access block, policy), EC2 security groups, `PublicIp`, **IAM privilege escalation** (`iam:PassRole` + `ec2:RunInstances`, `lambda:CreateFunction` + `lambda:InvokeFunction`, `iam:CreatePolicyVersion`), `AssumeRole` chains, CloudFormation/StackSet abuse, **AWS Organizations SCP bypass**, **ECR**, **Bedrock/RAG** poisoning. Herramientas: `pwncloud` (`https://github.com/RhinoSecurityLabs/pwncloud`), `cloudsplaining`, `ScoutSuite`, `Pacu`, `prowler`, `CloudGoat` (`https://github.com/WithSecureLabs/cloudgoat`), `Verrus` (`https://github.com/salesforce/verrus`), `roadtools`.
- **Azure/M365**: Entra ID (Azure AD) permisos, **OAuth consent phishing**, **Azure Functions / App Service**, **Azure Storage (SAS tokens)**, **Key Vault**, **Managed Identity abuse**, **Azure Subdomain Takeover**, `ROADtools` (`https://github.com/GhostPack/ROADtools`), `AADInternals`, `MicroBurst` (`https://github.com/NetSPI/MicroBurst`), `SharpHound` para Entra (`Get-AzureADUser`), `AzureHound`, **Cloud_Red_Team** (AWS/Azure TTP execution: `https://github.com/salesforce/Cloud-Red-Team`), **Stratus Red Team** (Azure TTP: `https://github.com/redcanaryco/stratus-red-team`).

- **Kubernetes**: `K8S`, `kube-hunter` (`https://github.com/aquasecurity/kube-hunter`), `kics` (`https://github.com/Checkmarx/KICS`), Kubuntu/CTF, metad8er, DVGA, `Kubi` — Configmaps/Secrets, RBAC, Network Policy, Container escape (`/proc`, docker socket), **K8S API server anonymous auth**.
- **Práctica obligatoria**:

1. Levanta **CloudGoat** o **Verrus** (vulnerable cloud lab) en AWS (con cuenta gratuita tier, $100 de crédito nuevo o existing). Si no puedes con AWS, usa **Azure for Students** ($100) o **Google Cloud Free Tier** ($300 + $100/mes para los primeros meses).
2. Levanta un clúster **Kubernetes** vulnerable (Kubespray en VMs, minikube o `kind` con labs CTF como `https://github.com/chrismccann/...` o `KubeHound`) y compromételo.
3. Escribe un informe de "pentest cloud" sobre CloudGoat con los paths que encontraste y el impacto (IAM privilege escalation, exfiltration de secrets).
4. Configura **CloudTrail** logging y una detección de `AssumeRole` anómalo (relevante para Blue Team).

2.5 Bloque D — Malware Analysis avanzado y Forense (~50 h)

- **Malware Analysis estático** (complementa lo básico de Fase 1):
  - **PE format**: DOS header, PE header, sections (.text, .rdata, .data, .rsrc, .reloc), imports/exports (IAT/EAT), TLS callbacks, resources, Rich header, overlay, entropy analysis, packing detection (UPX, Themida, VMProtect).
  - **ELF format**: structure, `.text`, PLT/GOT, `.init_array`, dynamic linker, shared libraries.
  - **Análisis de imports**: ¿qué capacidades sugiere? (crypto → ransomware, `CreateRemoteThread` → injection, `InternetOpen` → C2, `RegSetValue` → persistencia, `CryptEncrypt` → ransom).
  - **Herramientas**: **Detect It Easy (DiE)**, **PE-bear**, **PEStudio**, **capa** (Mandiant, automática, con **capa-rules**), **FLOSS** (strings dinámicos de .NET), **strings**, **ExifTool**, **YARA** para triage, **Rizin/Cutter** (`https://cutter.re` — alternativa libre de IDA, con Rizin).
  - **.NET malware**: **dnSpy** (decompilar), **ILSpy** (ILSpyCmd en CLI), **de4dot** (deobfuscador), **ConfuserEx** (deobfuscar), **dnlib**. Aprende a leer código .NET decompilado.
  - **Escribe tu propio "malware" inocuo**: un programa en C (o Python compilado con PyInstaller) que simula comportamiento malicioso, y analizalo tu mismo. Entiende el lado ofensivo de la ingenieria inversa.
- **Malware Analysis dinámico**:
  - **Sandboxing**: ANY.RUN (free tier), Triage, **CAPE** (`https://github.com/kevoreilly/CAPE`) auto-hospedado,joe Sandbox (pagado, opt). Analiza muestras de **MalwareBazaar** (`https://bazaar.abuse.ch`).
  - **Debugging**: **x64dbg** (breakpoints, patching, dumping, ESP, EIP, ROP gadgets, ScyllaHide para anti-anti-debug), **WinDbg** (Microsoft, `!analyze -v`, `!threads`, `!peb`, `lm`, `bp`, `g`, `ed` para modificar registros/memoria).
  - **Sysinternals en accion**: **Procmon** (registro de llamadas a la API de archivos y registro), **ProcDump** (`procdump -ma`), **Autoruns**, **Process Explorer**, **Psexec** para post-explotacion.

    - **Anti-análisis**: detección de VM/sandbox (MAC address, `GetSystemTime` checks, checking for VMware tools, `SbieDll.dll`, `vmtoolsd.exe`, checking de disco/tamaño de RAM/CPU), ofuscación, empaquetado, anti-debug (IsDebuggerPresent, NtQueryInformationProcess), timing checks, code injection, **process hollowing**, **DLL injection** (`VirtualAllocEx` + `WriteProcessMemory` + `CreateRemoteThread`), **reflective DLL injection**, **process of "living off the land"**.
- **Familias para estudiar** (muestras gratuitas de MalwareBazaar, analysis con writeups públicos):
    - **Ransomware**: **Qilin**, **Akira**, **LockBit 3.0**, **Black Basta/BianLian**, **Royal**, **Play**, **Medusa**, **Hunters International**, **8Base**, **Vice Society**, **Cl0p** (Cl0p^k3ysROT13). Lee sus **extortion notes, kill chains, victimologies** en `https://dfir.report`.
    - **Stealers**: **RedLine**, **Raccoon**, **Lumma**, **Vidar**, **StealC**, **Amadey**.
    - **Loaders/RATs**: **Qakbot** (**QBot/Pinkslipbot**), **IcedID**, **Emotet**, **AsyncRAT**, **Remcos**, **njRAT**, **AgentTesla**, **Cobalt Strike** (usado masivamente por crimeware, y como precio, understanding it es clave).
    - **Threat actor toolset**: **SUNBURST**, **SolarWinds Orion** (CVE-2020-12882, supply chain), **Cobalt Strike Beacon**, **PlugX**, **ShadowPad**, **Winnti**, **Gootloader** (2024, C#), **Latrodectus**.
    - **Casos de estudio obligatorios (lectura + writeup)**: **MOVEit** (CVE-2023-34362, Cl0p, MOVEit SQLi + LFI chaining), **Storm-2460 / Storm-2572** (Petya 2024, Quick create, ejecución de PowerShell, uso de ADS, `esentutl` y `reg save`), **xz-utils** (CVE-2024-3094, liblzma/`sshd` backdoor, supply chain, SSHD, iodine), **FortiOS** (CVE-2024-47575 "FortiMemory" — `fns daemon`), **Ivanti Connect Secure** (CVE-2024-21887 —.command injection + CVE-2023-46805 auth bypass), **Check Point** (CVE-2024-XXXX), **Cisco ASA** (CVE-2024-20353, CVE-2026-20406), **Snowflake** (UNC5537, credenciales robadas en infostealers, tokens, rotación), **MFA bombing / AiTM / ClickFix** (tendencias 2024-2025), **Termite/ToolShell** (SharePoint, CVE-2025-5528 + CVE-2025-53770/53771), **Apache Tomcat** (CVE-2025-24813), **CVE-2025-32433** (Erlang/OTP SSH RCE). Para cada uno: root cause, TTP ATT&CK, detección (Sigma rule), remediación.
- **Escribe tu propio informe de análisis**: toma una muestra de **Emotet**, **Qakbot** o **AgentTesla** de MalwareBazaar, y produce un informe de 5 páginas con: hash, comportamiento estático, IOCs, comportamiento dinámico, detección YARA/Sigma, conclusión.

2.6 Bloque E — Blue Team avanzado / Detection Engineering (~50 h)

- **Frameworks de detección**: **Sigma** (escritura y conversión a SIEM queries: `sigma cli` + `pySigma` con backends `sigma-cli`), **YARA** (escritura de reglas, `yara-python`, `YARA-X`), **ATT&CK mapping** (`Atomic Red Team` ↔ `Sigma` ↔ `MITRE Navigator`).
- **Sigmahq** (`https://github.com/SigmaHQ/sigma`): aprende a escribir reglas con `detection`, `condition`, `falsepositives`, `level`, `logsource`. Contribuye a la colección.
- **MITRE Caldera** (`https://github.com/mitre/caldera`): simula ataques reales para testear tus detecciones (operacionespurple team automatizadas). Configura un agente Windows,unnel y ejecuta 10 operaciones.
- **Atomic Red Team** (ejecuta 300+ técnicas): úsalo **para testear tus detecciones**. Automatiza con Caldera o con `Invoke-AtomicRedTeam`.
- **Splunk BOTS** (datasets Boss of the SOC v1/v2/v3): resuelve 3-5 imágenes de datasets, esto es un estándar de facto para BEC/IR. Ejercita tu capacidad de hunting.

- **KQL** (para Microsoft Sentinel / Defender): operators, joins, `datatable`, `summarize`, `make_series`, `bin`, `extract`, `parse`, `lookup`, `union`. Escribe 20 queries de hunting: Shadow Copies, LSASS access, NTLM anomalies, PowerShell encoded commands, unusual parent-child process, SUID-like behavior, Scheduled Task creation, 4662, service creation 7045, Kerberoasting 4769, remote services creation 4697.
- **Sigma to Sentinel**: `https://github.com/SentinelOne/SentinelOne-Queries`, Microsoft's `https://github.com/Azure/Azure-Sentinel` (deprecada, pero útil), `SecurityCopilot`.
- **SOAR**: **TheHive + Cortex + MISP** (todo open source), **Shuffle** (SOAR OSS), **Dead Man's Snitch** (monitorización), **Pomerium**, etc.
- **Detección de Ransomware**: crea una detección end-to-end de **Shadow Copy deletion** (`vssadmin delete shadows`, `wmic shadowcopy delete`, `bcdedit /set recoveryenabled No`, `wbadmin delete catalog`) → esta es una regla "king". Practica con simulaciones (Atomic Red Team).
- **Blue team lab**: levanta un **Windows Server AD**, ejecuta 25 técnicas Atomic, y **detecta al menos 20 de ellas con tus reglas** — este es tu proyecto estrella de Fase 2.
- **Referencias**: `https://dfir.report`, `https://www.splunk.com/en_us/training`, `https://blueteamlabs.online`, `https://cyberdefenders.com`, `https://letsdefend.io`, `https://redcanary.com/threat-detection-hub` (descarga de Sigma/YARA packs), `https://github.com/SigmaHQ`, `https://github.com/elastic/detection-rules`, `https://github.com/splunk/security_content`.

2.7 Bloque F — Reverse Engineering (30-40 h)

- **Conceptos de RE**: flujo de análisis (triage → estático → dinámico → profundo), architecture x86/x64 (registros, flags, calling conventions `cdecl/stdcall/fastcall`, pila, segmentos, `x86-64` (RIP-relative, registros extendidos), [sistema operativo y compilers basics]).
- **Herramientas**: **Ghidra** (NSA; **gratis** y la opción profesional hoy), **IDA Free** (gratis, con limitations), **Cutter/Rizin** (libre), **x64dbg** (Windows), **lldb**, **Binary Ninja** (no gratis), **radare2**.
- **Ejercicio 1**: invertir un binario de 32 bits (Linux) de un CTF (pwn.college) y resolverlo.
- **Ejercicio 2**: analizar un crackme en Ghidra y explicar el algoritmo de validación.
- **Ejercicio 3**: analizar un `.exe` de malware real de MalwareBazaar en Ghidra + x64dbg, identificar la C2 (URL/IP/dominio) y escribir una regla YARA.
- **Ejercicio 4**: **.NET reversing** — descompilar un malware .NET de CryptoLocker/AgentTesla con dnSpy, identificarOfuscación, extraer configuración.
- **Ejercicio 5**: **Linux malware analysis** — analizar un ELF (keylogger, rootkit) en Ghidra, identificar syscalls.
- **Ejercicio 6**: **Unpack malware** (UPX y un packer custom) — desempaquetar y analizar el payload final.
- **Referencia gratis**: **OpenSecurityTraining2** (x86-64 Assembly, gratis, muy completo), **Practical Reverse Engineering** (Dennis Andriesse, `https://github.com/denisandriesse/practical-re`), **"Reverse Engineering for Beginners"** (Yurish, gratis en PDF), **pwn.college** (módulo de "Reverse Engineering" y "Binary Exploitation"), **cutter.re tutorials**.
- **Lenguaje**: **Assembly x86-64** (usa `https://www.ostraining.org` y el libro **"Intel® 64 and IA-32 Architectures Software Developer's Manual"** (gratis en Intel). Necesitarás también **C bajo**

WinAPI/Linux para los ejercicios.

2.8 PROYECTOS OBLIGATORIOS DE LA FASE 2

#	PROYECTO	ENTREGABLE	HORAS
P2.1	**Comprometer GOAD completo hasta Domain Admin**	Writeup de 3000+ palabras con BloodHound graph, método (Kerberoast→ADCS→DCSync), diagramas, detección para cada paso (Sigma)	30 h
P2.2	**AD CS (ESC1-ESC8) lab**	Monta un CA vulnerable en tu AD, explota ESC1 con Certipy, y detecta la emisión de certificadosinks	15 h
P2.3	**Exploit propio funcional**	Buffer overflow → shell en un binario sin PIE + una cadena ROP. Writeup estilo CVE	20 h
P2.4	**Informe de malware analysis**	Informe de 8-10 páginas de una muestra real (Qakbot/AgentTesla/Emotet): estático, dinámico, IOCs, Sigma, YARA	15 h
P2.5	**Pentest de cloud (CloudGoat o Verrus)**	Informe de 10 páginas: IAM privesc, exfiltración, detección, remediación	15 h
P2.6	**Blue team: 25 técnicas Atomic, 20 detectadas**	Dashboard + writeup de detección con las reglas Sigma y el dashboard de Splunk/Elastic	20 h
P2.7	**EC2 Kubernetes pentest**	Compromiso de un clúster K8s (kube-hunter → container escape → secrets)	12 h
P2.8	**Full IR report**	Adquirir imagen de un DC comprometido, timeline, Volatility 3 rootkit hunt, informe final	15 h

2.9 Certificaciones de la Fase 2

CERTIFICACIÓN	COSTE	NOTA
CPTS (Hack The Box)	~$210	**Excelente relacion calidad/precio**; cubre red, web, AD, cloud.
eCPT (INE)	~$399	
eJPT -> eCPT/ePPTP	~$300	Ruta INE clásica.
CompTIA PenTest+ (PT0-002) o **CySA+ (CySA-002)**	~$350-500	Con vouchers Microsoft/Cisco, bajan a $150.
Blue Team Level 3 (BTL3-EXT/BTL3-INT)	~$700-900 (pack)	Caro pero es **el** estandar de deteccion. Alternativa: el contenido de BTL3 vale la pena.
Google Cloud / AWS Cloud Security specialty (opcional)	~$300 (exam)	Barato si usas free tiers para los labs.
CRTO/CRTPP (Practical Networks)	~$300	Detección y purple team.
BTL3 / CySA+ / SANS SEC555 (Alternativas)		El SANS **SEC555** es el mejor curso de blue team; **sec573** (Threat Intelligence/PSIRT) y **SEC450** (Windows IR) son excelentes. SANS ofrece **becas** (SANS scholarship) y acceso a la biblioteca.

2.10 ✅ CHECKLIST "He terminado la Fase 2 cuando..."

- [ ] He obtenido **Domain Admin** en un AD multi-DC **sin usar la GUI**, mediante técnicas de redteam documentadas (Kerberoast, ADCS, DCSync, PTH, Coerción).
- [ ] He mapeado un AD completo con BloodHound y he interpretado los shortest paths a DA.

- [ ] He escrito un **exploit propio** (no copiado de un tutorial) que consigue ejecución de código.
- [ ] He expresado un vuln de un servicio real (con writeup) o he reportado un CVE a un proyecto real.
- [ ] He hecho un **pentest de cloud** real (AWS/Azure) y he escalado de low-priv a control de la cuenta.
- [ ] He pentesteado un cluster Kubernetes y escalado privilegios.
- [ ] He analizado malware real (.exe y .NET) y he extraído C2, IOCs, y una regla YARA que detecta la familia.
- [ ] He **creado** 30+ reglas Sigma, 10 YARA, 20 queries KQL/SPL, y he probado que **disparan de verdad** contra Atomic Red Team.
- [ ] He hecho un pentest **end-to-end** de una máquina HTB de dificultad "Medium"/"Hard" y lo he documentado.
- [ ] He hecho 60+ labs de PortSwigger Academy, incluyendo los "hard" de smuggling/deserialización/business logic.
- [ ] Tengo certification de Fase 2 (CPTS, eCPT, CySA+).
- [ ] Duración real: **~450 h** (más de 6 meses a 20 h/semana).

FASE 3 — AVANZADO (Senior Security Engineer / Red Team Operator / Detection Engineer)

Duración: 8-12 meses (año 2) · 700-900 horas · 4-6 años de experiencia

Aquí decides tu **especialización**. No intentes ser experto en 5 cosas a la vez. Las tres salidas reales con mejor mercado:

RUTA	PERFIL	DEEP DIVE PRINCIPAL
A. Red Team Operator	Red team, pentesting avanzado, C2, exploit dev, AD/cloud/offensive	C2, detección bypass, detección de EDR, movement
B. Detection/Blue Team Lead	Detection engineering, DFIR, threat hunting, malware, IR	Detección, DFIR, hunting, respuesta
C. Cloud/AppSec/AppSec Research	Cloud security engineering, AppSec, secure coding, threat modeling	IaC, CI/CD security, threat modeling, API security

La regla de los Experts: elige UNA ruta y profundiza 6-9 meses seguidos. La combinación de 2 (p.ej. red + cloud) también vale; 5 no.

3.1 Ruta A — RED TEAM OPERATOR (semanas 1-26 del Año 2)

3.1.1 Dominio de Active Directory a nivel ofensivo (100 h)

- **Pass-the-Hash variants**: shadow credentials (`mimikatz lsadump::shadow /format:list`, `secretsdump.py`), **Shadow Credentials for both EFS and DPAPI**, `Rubeus::pth`, delegation abuse, ticket manipulation.
- **Kerberos avanzado**: **Kerberoasting en masse** (`Rubeus kerberoast /user: ...`, `netexec ldap ... --kerberoasting /user:*`), **AS-REP roast en masse** (`Rubeus asrep /domain:`, `netexec ldap ... --asrep /user:*`), **Kerberoasting con AES-256 vs RC4**, **Overpass-the-hash** (shadow creds + RBCD), **Unconstrained/Constrained delegation attacks** (`Rubeus delegate /user: ... /keytab: ...`, `msDS-AllowedToDelegateTo`), **RBCD** (`addShadowCredential`, `Rubeus::deltatoa`), **DNS delegation**, **CVE-2022-4139 / CVE-2023-4428** (Notepad++ / 3D Viewer 2023, printer driver, hotpatch, UserTarget).
- **C2 de AD**: **Sliver** (Go, MITRE, modular, gratuito, maduro), **Mythic** (Cobalt Strike-free C2, OSS), **Havoc** (C++, GitHub), **Brute Ratel C4** (no OSS, solo compra o de licencia), **Adapt** (CS post-ex), **pwncat** (bind/reverse shellsLongres).
- **Pivoting a fondo**: **ligolo-ng** (TUN-based pivoting, preferible a chisel), **chisel**, **SOCKS5 con `ssh -D`**, **proxychains** con `msfvenom` reverse, **Pivoting en AD** (moverse a través de workstations, `netsh interface portproxy`, `SharpSocks`), **RDP tunneling**, **WinRM tunneling** (`Enter-PSSession` sobre portproxy).
- **Bypass de controles de Windows**: **AMSI bypass** (`patch amsi.dll`, `amsiContext` hooking, `AmsiScanBuffer` patching), **ETW patching** (patching `ntdll!EtwEventWrite`), **unhooking** de Sysmon (Sysmon DLL unhooking, Windows 11 AMSI kernel patches), **direct syscalls** (`NtAllocateVirtualMemory` from `ntdll` via `syswhispers`), **PPL bypass** (`PPLKiller` / `PPL-`

`Dumping`), **Windows 11/10 CFG bypass**, **ETW-agnostic injection**, **Peb/TEB unhooking**, **Process injection variants** (`CreateRemoteThread`, `NtCreateThreadEx`, `QueueUserAPC`, `SetWindowsHookEx`, manual mapping con `NtMapViewOfSection`).

- **Esteganografía y tunneling**: DNS tunneling (`iodine`, `dnscat2`, `dnscat2` reverse via `iodine`), **HTTP/HTTPS C2 over legit channels**, `icmp` tunneling (`ptunnel-ng`), **TCP-over-HTTP** (`chisel`, `ngrok`, `frp`), **domain fronting** (aunque en 2025 está muy mitigado, entiéndelo), **TXT record tunneling** en C2.
- **Lateral movement**: **PsExec** (ofuscated variant), **WinRM**, **WMI**, **DCOM** (`Mimikatz LSADump::dcsync`, `Sharp-WMI`, `nc` reverse over **DCOM port 135/49152-65535**), **MSSQL lateral** (`xp_cmdshell`, `OPENROWSET` / `linked servers`), **WinRM**, **Remote Registry**, **RDP (token theft, `mstsc /restrictedAdmin`, session hijacking)**, **Pass-the-Ticket**, **RDP cache poisoning**, **NTDS exploitation**.

3.1.2 Explotación y research (100 h)

- **Publicación de vulnerabilidades**: aprende **CVE disclosure workflow** responsibly: reportar, coordinar con el proveedor, CVSS scoring, escribir un **PoC público** (CVE).
- **Bug Bounty serio**: empieza en plataformas con scope claro: **HackerOne**, **Bugcrowd**, **Intigriti**, **YesWeHack**, **Open Bug Bounty**, **Google VRP**, **Meta**; aprende a hacer recon a escala de dominios grandes, enumerar programas con mayor severidad.
- **Vulnerability research**: **fuzzing con AFL++ + libFuzzer**, lectura de código fuente (C/C++/Rust/Go) buscando memory corruption, y **recherche** en primitivas (integer overflow, type confusion, TOCTOU, race conditions, UAF en allocator, off-by-one).
- **Fuzzing targets**: fuzz de una app open source real (ej. `libpng`, `libxml2`, `json-parser`, un driver, un juego).
- **Assembly y shellcode advanced**: **Windows shellcode** (`NtAllocateVirtualMemory` + `RtlCreateUserThread`), **Linux shellcode** (execve, `mdup`...), **Egg-hunting**, **AV evasion** polymorphic / metamorphic, **Staged shellcode (Cobalt Strike style)**.
- **CVE-2024-2025 que puedes estudiar como caso**: **xz-utils** (supply chain), **CVE-2024-3094**, **V8 / Chrome** sandbox escapes, **FortiOS SSL VPN**, **Ivanti** (chains: auth bypass + command injection), **Cisco ASA** (CVE-2024-20353, webvpn), **FortiManager** (CVE-2024-47575), **Chrome V8 (CVE-2024-XXXX) type confusion**, **Linux kernel netfilter** (CVE-2024-XXXX), **F5 BIG-IP**.

3.2 Ruta B — DETECTION ENGINEERING & DFIR LEAD (semanas 1-26 del Año 2)

3.2.1 Detección a escala (100 h)

- **Ciencia de la detección**: qué es una **falsa positiva**, **falso negativo**, **cobertura de ATT&CK**, **modelo deaguaje** (ATT&CK coverage, gaps), **Threat-informed defense**.
- **Escribir reglas de calidad**: 200+ reglas Sigma, 30+ YARA, 50+ KQL, 30+ SPL, todas con `falsepositives`, `level`, `references`; miden tu cobertura con **MITRE ATT&CK Navigator** y **MITRE Engage**.
- **Convertir y desplegar**: `sigma cli` + `pySigma` con pipelines a Sentinel, Splunk, Elastic, Chronicle, Loki, Carbon Black; constructo de reglas en un pipeline CI (unit tests de reglas, lints, validaciones, deployment automatizado, versionado en git, peer-review).

- **Modelos de deteccion ML**: **Sigma + anomalias**, machine learning-based detection (Isolation Forest, autoencoders) para anomalias de red y DNS (aplicado con `river` o scikit-learn). Base: "Anomaly Detection in Cybersecurity" y los blogs de SANS/MITRE sobre ML para deteccion.
- **Hunting a escala**: **Threat hunting hypotheses** (basado en ATT&CK), framework **"Hypothesis-driven hunting"** de "The Hunting Handbook" (Robert M. Lee, gratis: `https://www.happyhacking.co/hunting-handbook`), **SigmaHQ/Elastic detection rules** (`https://github.com/elastic/detection-rules`), **Splunk ESCU (Security Content Update)** (`https://security.splunk.com`).
- **Purple teaming a escala**: **MITRE Caldera** (operaciones automáticas), **Atomic Red Team** (300+ técnicas), **VECTR** (tracking de purple team, `https://github.com/SecurityRiskManagement/VECTR`), **Stratus Red Team** (cloud), **Prowler** (compliance), **AttackForge**.
- **Purple team emulando ataques completos**: conduce simulacros que replican **technique reales** de actor (ej. **UNC5537** Snowflake, **Storm-2460** Petya, **Black Basta** con QKD, **Scattered Spider** con técnicas de Help Desk), y verifica qué detecta tu stack.

3.2.2 DFIR profundo (100 h)

- **Windows forensics avanzado**: NTFS artifacts completos (`$MFT`, `$UsnJrnl:$J`, `$LogFile`, ADS, `$I30`, `$ObjId`, `$Quarantine`), **Shimcache / Amcache / Prefetch / BAM-DAM / UserAssist / LNK / Jump Lists / RecentDocs / MUISIC**, **LNK/LNK2 forense** (Zeek, LinkParser), **registry artifacts** (USB, TypedPaths, Shellbags, MountedDevices, RunMRU, **Uninstall keys**), **Event logs** (evtx parsing, `chains`, `Windows Event Log` forense), **EFS**, **Prefetch+Superfetch**.
- **Memory Forensics avanzado**: **Volatility 3 plugins** (`windows.memmap`, `windows.vadinfo`, `windows.ldrmodules`, `windows.malfind`, `windows.vadyarascan`, `windows.cachedfile`, `windows.vadinfo`, `windows.sharptcl`), **Windows memory dumps** (crashdump, `hiberfil.sys`, `pagefile.sys`, `WinPmem` — ya integrado en Volatility 3), **Dridex/Emotet memory** (inyección de proceso, hollowing, APC injection), **Volatility 3 plugins para rootkits** (`windows.modmap`, `windows.ssidtable`, `windows.driverscan`, `windows.symscan`).
- **Network Forensics**: pcap analysis, `Zeek` (antes Bro) en profundidad (`conn.log`, `dns.log`, `http.log`, `ssl.log`, `notice.log`, `files.log`, `x509.log`, `weird.log`), **pcap forense** (truncamiento, tunneling, `tshark` para extract), **JA3/JA4 fingerprinting**, **HTTPTunneling detection**, **DNS tunneling detection** (entropía de subdominios, longitud, volumen), **C2 traffic detection** (Beacon patterns, jitter, **JA3S** clustering).
- **Timeline y reporting**: **Plaso/log2timeline** (super-timeline), **Timesketch** (colaborativo), **KAPE** para adquisición masiva, **Sleuth Kit**, `log2timeline.py`, correlation de fuentes (EDR, firewall, proxy, email, AD), informe de IR ejecutivo + técnico, **indicadores de compromiso (IOCs) vs indicadores de comportamiento (IOBs)**, **MITRE ATT&CK en el informe**.
- **Malware analysis a nivel de research**: reverse engineering profundo (ya visto en 2.7), **C2 protocol reverse engineering**, **config extraction** (Cobalt Strike config, stealer config, ransomware config), **YARA para familias**, **reglas de detección de alto valor**.
- **IR real**: response a ransomware, BEC, cloud compromise, insider threat, supply chain compromise. **Escribe 3 playbooks de IR** (ransomware, cloud, BEC) con las acciones de cada fase.

3.3 Ruta C — CLOUD SECURITY / AppSec (semanas 1-26 del Año 2)

- **AppSec**: **threat modeling** (STRIDE, attack trees, data-flow diagrams), **secure SDLC**, **SAST/DAST/IAST/SCA**, **OWASP ASVS**, **SAMM (Software Assurance Maturity Model)**, **API security** (OWASP API Top 10, GraphQL, `swagger` parsing, `k6` / `ffuf` para API fuzzing, `apitest`), **supply chain security** (SBOM, SLSA, Sigstore, Dependabot, `osv-scanner`, fuzzing de dependencias), **IaC security** (Terraform, Kubernetes, `checkov`, `terrascan`, `kics`, `tfsec`), **CI/CD security** (pipeline attacks, secret leakage en logs, Actions abuse, artifact tampering).
- **Cloud avanzado**: AWS (IAM deep, STS, Lambda@Edge, CloudFront, KMS, IAM Access Analyzer, SCP, Organizations, control plane), Azure (Entra ID deep, PIM, Managed Identity, Azure AD Graph), GCP (IAM, service accounts, org policy), multi-cloud security posture, CSPM (Cloud Security Posture Management), CNAPP (Cloud-Native Application Protection Platform), CASB.
- **Cloud detection**: `CloudTrail` + Athena + CloudWatch, **CloudTrail Lake**, `GuardDuty`, `Security Hub`, `Defender for Cloud`, Azure Sentinel (Defender for Cloud), GCP Security Command Center, SentinelOne/Palo Alto cloud, `Zeek` en VPC flow logs, `Suricata` en EKS, MITRE ATT&CK for Cloud, **Cloud Security Alliance (CSA)**, **Cloud Threat Alliance**.

3.4 PROYECTOS OBLIGATORIOS DE LA FASE 3

#	PROYECTO	ENTREGABLE	HORAS
P3.1	**Simulación red team completa (purple team)**	Plan de 2 semanas contra tu AD + cloud lab; ejecutas con Sliver, mides la detección, escribes informe de "misses"	60 h
P3.2	**200 reglas Sigma/ YARA desplegadas y probadas**	Repositorioorio, CI con pySigma, dashboard de cobertura ATT&CK	40 h
P3.3	**Investigación y CVE**	Find + reportar una vulnerabilidad real (CVSSv4, PoC) a un proyecto open source	60 h
P3.4	**Serie de malware research**	Análisis de 10 muestras de malware nuevas, informe de 50+ páginas con YARA para 3 familias	40 h
P3.5	**Publicación técnica**	Escribe 2 blogs técnicos (data de phishing, bypass de EDR, o DFIR) publicados en Medium/HackerOne/0x00ffsec, o un whitepaper interno de SANS	30 h
P3.6	**Infraestructura de detección completa**	Pipeline `Sigma → SIEM` en GitHub Actions, Wazuh/Elastic/Splank dashboard con 30+ visualizaciones, hunting queries	30 h
P3.7	**Lead de un purple team de 3 días**	Simula un ataque realista (no solo Atomic), mide la detección, escribe el "debrief"	25 h

3.5 Certificaciones de la Fase 3 (las que importan de verdad)

CERTIFICACIÓN	COSTE	POR QUÉ
OSCP (OffSec)	~$1599 + 30 días de lab	El estándar de facto de redteam. Si vas a ruta A, es obligatorio.
OSWE (OffSec)	~$1600	Web apps. Ruta C.
OSIR / **OSEP** (OffSec)	~$1800-2200	**Experto.** Alto prestige, muy difícil. Considera la beca.
CRTO (ZeroPointSecurity)	~$500	Detección + redteam de windows/linux, muy práctico.
OSEP (beca)	Beca (~75 %)	La beca de Offensive Security reduce el coste a ~$500. Postula.

CERTIFICACIÓN	COSTE	POR QUÉ
CISSP / **CISM** / **CISA**	~$700-1000	No son técnicos, pero **valen puertas en empresas**. CISSP requiere 5 años de experiencia.
CISM / CISA		Para IR/GRC/leadership.
GSEC / GCIH (SANS)	~$4000-9000	SANS GIAC es el estándar de blue/IR; **muy caro**, pero con **SANS scholarship** o el "SANS CyberDefenders" gratuito puedes empezar.
Google Professional Certificate: "Advanced Data Analytics"? no.		Los "Specializations" de Google en Security Engineering son útiles.
CISA (ISC2)	~$600	Buen para-manager de seguridad.
Azure/AWS security specialty	~$300	Si tu ruta es cloud.
SANS SEC660 (Advanced IR & Threat Hunting) / SEC599 (Defeating the adversary)	$5000+	El contenido vale su dinero, aunque el precio duele. Alternativa: **BTL3**, **CySA+**, o el **"Applied Network Defense"** material de QOMPL.

3.6 ✅ CHECKLIST "He terminado la Fase 3 cuando..."

- [ ] He elegido **una** ruta y llevo 6 meses sin abandonarla.
- [ ] He **liderado** (o ejecutado como senior) un engagement completo (red team, DFIR o cloud) de principio a fin, con informe executive.
- [ ] He reportado al menos 1 **CVE** con CVSSv4, o 3+ CVEs en plataformas de bug bounty.
- [ ] He publicado 2+ blogs técnicos o playbooks.
- [ ] Mi pipeline de detección está en CI/CD, con >200 reglas, >80% de cobertura ATT&CK de mi negocio.
- [ ] He escrito un **whitepaper** (o un curso) que otra persona ha/vndusado.
- [ ] Tengo una certificación de Fase 3 obtenida (OSCP, CRTO, CISA, o la de cloud/security de tu ruta).
- [ ] He mentorado a al menos a 1 junior (incluso informalmente).
- [ ] Duración real: **~800 h**.

FASE 4 — SENIOR / EXPERTO (Security Engineer Senior / Lead / Researcher)

Duración: 12-18 meses · 1000+ horas · 6+ años de experiencia

Aquí ya no se trata de "aprender cosas nuevas" sino de **sintetizar, liderar, investigar y_REMIR** (investigar y enseñar).

4.1 Objetivos medibles

1. **Liderar** un equipo o un engagement complejo (con múltiples escaladores, timing, reporting a C-level).
2. **Diseñar** la arquitectura de seguridad de una organización (o de un pipeline de detección) desde cero.
3. **Investigar y publicar** investigación original (CVEs, conference talks, papers, herramientas OSS).
4. **Ser referente** técnico en al menos una niche (p.ej. "la persona de AD CS de mi sector", "la que escribe sobre forensic de cloud en español").
5. **Diseñar y mentorar**: un programa de formación interno, un Purple Team plan anual, un threat model de un producto.

4.2 Áreas donde profundizar (elige 2-3)

A. Arquitectura de seguridad y riesgo

- **Secure design**: threat modeling (STRIDE, LINDDUN, attack trees), security control design,Zero Trust Architecture (NIST SP 800-207), **security-by-design** en SDLC, **AppSec**: OWASP ASVS + SAMM.
- **Seguridad cloud a escala**: landing zones (AWS/Azure), CSPM/CNAPP/CNDP, IAM a escala, Data Perimeter, DevSecOps pipeline, IaC security a escala, Policy-as-Code (OPA/Rego).
- **Modelos de riesgo**: FAIR, factor analysis, quantitative risk; **Frameworks**: ISO 27001 (implementación práctica: ISMS), NIST CSF 2.0 (maturity model), CIS Controls v8, NIS2 (obligatorio en muchas empresas europeas en 2024-2025), DORA (resiliencia financiera en UE), **Ensayo REGUAM**: SGSI, ENS RD 2022/37, ISO 27001/27701, SOC 2.
- **Seguridad de la cadena de suministro**: SBOM (SPDX, CycloneDX), SLSA, **NIST SSDF**, TIBER/Coordinated Vulnerability Disclosure, software supply chain risk management.

B. Investigación de seguridad (vuln research, exploit dev de alto nivel)

- **Research original**: lectura de CVEs y source, fuzzing avanzado, symbolic/concolic execution (**angr**, **KLEE**, **Triton**, **Manticore**), **static analysis** (SAST escalado), **symbolic execution** de binarios reales, finding de 0-days en software widespread (in Responsible Disclosure).
- **Publicar**: en conferences (**Black Hat**, **DEF CON**, **BSides**, **HITB**, **Nullcon**, **SteelCityCon**) o como technical writeup en **HackerOne**, **Google Project Zero style** blog, **SANS Internet Storm Center (DFIR)**, **Mitre ATT&CK** para nuevas técnicas, **CVE** en proyectos reales.

- **Herramientas propias OSS**: escribe una herramienta (scanner, fuzzer, detector, plugin de Burp, framework C2) y publícala en GitHub. Contribuye a proyectos existentes (BloodHound, Sliver, Sigma, Nmap, Impacket, Ghidra).
- **Ingeniería inversa de alto nivel**: reversing de JVM/.NET/Go binaries, **GoReSym** (Go binaries), **protocol reverse engineering** (Telegram, WhatsApp, Steam, Discord bots), **driver/kernel RE**, **UEFI/BIOS reverse engineering** (básico).

C. Detection Engineering / DFIR Leadership

- **Arquitectura de detección**: designing a detection program, **métricas de cobertura**, **evaluación de calidad de reglas** (unit testing, deduplication, tuning), **plataformas de detección** (Sigma → SIEM, YARA → EDR, sigma-YARA pipeline, detection-as-code), **fuzzing de detecciones** (escritura de tests adversarios).
- **DFIR leadership**: response playbook, digital forensics leadership, litigation support, **Expert witness** (basic), threat intelligence programs, **CTI** (cyber threat intel): **MISP**, **OASIS STIX/TAXII**, **ATT&CK-based intel**, **curated feeds** (abuse.ch, MISP, OTX), **reporting to ISACs** (CISA, NCSC, INCIBE/CERTS).
- **Malware research leadership**: track emerging families, publish YARA, coordinate with CERTs, produce threat reports (public **APT reports**, e.g. de Mandiant/CrowdStrike/ESET), **reputation**.

D. Red Team leadership

- **Operaciones de red team a escala**: **planning, ROE, rules of engagement, deconfliction, OPSEC, infraestructura** (**C2 domains, redirectors, proxies, "burner" infra, OPSEC-safe infra**), **Adversary Emulation** (**Atomic Red Team, MITRE Caldera, VECTR**), **Full-scope y phased** engagements, **attack surface management** en Fetch/Pentera.
- **Infrastructure de ataque OPSEC-segura**: **C2 domains rotativos, redirects, "dead drop" resolvers, TLS fingerprint mimicry (JA3/JA3S), traffic shaping, timing jitter** — replica los patrones de Cobalt Strike/Mythic/Sliver, para que tu tráfico sea indistinguible del real (esto es **defensa contra detection**; en un engagement autorizado es perfecto).
- **Purple team con el equipo de defensa**: escribir los **attack plans** y **los detection tests**; **Threat-informed defense**.
- **Métricas de red team**: **Atomic / MITRE coverage de detección**, **emulación de actores** (**Emulate APT29 / APT28 / FIN7 / Lazarus / Sandworm / Scattered Spider** con "adversary profiles" y Atomic sets, "Atomic Red Team - Enterprise" tiene sets listos para simular actores específicos).

4.3 Certificaciones de la Fase 4

- **CISSP** / **CISA** / **CISM** (líder/GRC, requieren 5 años exp)
- **OSEP** (OffSec) — el pináculo ofensivo. Becas disponibles.
- **GCIH / GCFA / GCFE / GNFA** (SANS GIAC) — el estándar DFIR/leadership.
- **CISO** (si te orientas a dirección), **CompTIA** CISSP**/**Security+**.
- **Masters/MBA** en cyber (opcional, poco ROI técnico).
- **CISSP** (el "gold standard" de management), **CISM** (gestión de riesgos), **CRISC** (riesgo).
- En paralelo, **publicaciones y)** speaking: **DEF CON**, **BSides**, **OWASP AppSec**, **SANS Summit**.

4.4 PROYECTOS DE LA FASE 4

#	PROYECTO	ENTREGABLE	HORAS
P4.1	**Diseña e implementa** la estrategia de seguridad (o de detección) de una organización completa	Documento de arquitectura de 60+ páginas + implementación funcionando	100 h
P4.2	**Programa de mentoring**: mentoriza a 2-3 juniors	2-3 personas que llegan a nivel junior/intermedio	80 h
P4.3	**Research original**: 1-2 CVEs o una herramienta OSS significativa	CVE + blog técnico + código público	150 h
P4.4	**Lead de un engagement** grande (multi-dominio, multi-técnica)	Informe de 50+ páginas con C-level exec summary	80 h
P4.5	**Threat model** de un producto real	Diagrama DFD + STRIDE + mitigaciones	40 h
P4.6	**Publicación en una conference** (BSides/DEF CON/AppSec) o 2 whitepapers en SANS DFIR	Talk aceptado + slides	80 h
P4.7	**Construye un home lab Private Offensives en un host Windows 11 con 32-48 GB de RAM**	Multi-DC AD, cloud, Kubernetes, SIEM, EDR — todo integrado	60 h

4.5 ✅ CHECKLIST "He terminado la Fase 4 (Senior) cuando..."

- [ ] He **liderado** un engagement/programa de seguridad completo (del scope al board report).
- [ ] He **diseñado desde cero** un pipeline de detección o una arquitectura de seguridad (con un diagram funcional, bien documentado).
- [ ] He **publicado** investigación original: al menos 1 CVE, o 2 blogs técnicos, o 1 paper, o 1 talk.
- [ ] He construido y mantengo **una herramienta OSS** con usuarios.
- [ ] Tengo mentorados a 2-3 juniors que hanDennis empate nivel.
- [ ] Mi github/publicaciones funcionan como **portfolio de referencia** (empresas me buscan a mí, no al revés).
- [ ] Tengo CISSP/CISA/OSEP/GCFA o equivalente.
- [ ] He dado al menos 1 workshop/tutorial/tutorial. Duracion real: **~1000 h** (muchas de las cuales son trabajo real, no estudio).

PARTE FINAL

5.1 Principled de la Regla 70/30

CANTIDAD	% DEL TIEMPO DE ESTUDIO
Teoría (cursos, vídeos, lecturas)	**30 %**
Práctica (labs, CTFs, writeups)	**50 %**
Escritura y documentación (writeups, informes, notas)	**20 %**
→ Lo que esto produce	Un profesional empleable

5.2 Métricas para medir tu progreso (dashboard personal)

Lleva un spreadsheet con estas columnas y actualízalas semanalmente:

- **Horas acumuladas** (por fase y total)
- **Certificaciones** (con fecha, y "cuántos de los 3 escenarios reales de cada cert he hecho")
- **CTFs/labs resueltos** (por plataforma y dificultad)
- **Writeups publicados** (cantidad, calidad)
- **Habilidades** (1-5 por dominio: red, web, AD, cloud, RE, blue, forense, OSINT, social)
- **Empleos/entrevistas**: DATE + de qué setrataba
- **CVE/BB/reports**: count

5.3 La advertencia más importante

El mercado **no** premia "saber muchos comandos". Premia:

1. **SaberDiagnosticar** (pensar como atacante, entender la lógica del negocio y por qué se rompe).
2. **SaberComunicar** (informes claros, writeups, hablar con ingeniería y con negocio).
3. **SaberMantenerse actualizado** (leer reports, seguir CTF, escribir tus propias herramientas).
4. **Saber profundo en 1-2 dominios** (más que breadth superficial en 5).

5.4 Resumen visual del roadmap

CODE0

PARTE A — Plan de 52 semanas (año 1) a 15-20 h/semana

A.1 Rutina semanal tipo (la plantilla que tienes que repetir)

DÍA	BLOQUE	DURACIÓN	ACTIVIDAD	TIPO
Lunes	17:30-19:30	2 h	**Teoría + práctica guiada** del tema de la semana (curso/libro/PaperSwigger)	30 %
Martes	17:30-19:30	2 h	**Lab práctico** (TryHackMe room, HTB machine, PortSwigger lab, pwn.college)	50 %
Miércoles	17:30-19:30	2 h	**Lab práctico** (continuación) o **lectura de código/fuente**	50 %
Jueves	17:30-19:30	2 h	**Proyecto de la fase** (escribir el script, montar el lab, redactar)	50 %
Viernes	17:30-18:30	1 h	**Writeup + documentación**: escribe lo que hiciste los 4 días anteriores	20 %
Sábado	10:00-13:00	3 h	**Bloque largo**: tema nuevo + ejercicio práctico profundo	mixto
Domingo	17:00-18:00	1 h	**Actualización técnica**: newsletter, informe nuevo, 1 writeup de CTF, notas de ATT&CK	mixto
	TOTAL	**13-15 h**		

Para llegar a 20 h: añade un segundo bloque de 2 h el martes o miércoles por la tarde, o un bloque de 2-3 h el domingo por la mañana. Nunca sobre 4 h seguidas sin descanso.

Reglas de la rutina:

1. **Nunca**_more de 2 sesiones consecutivas sin escribir algo. Lo que no se escribe, se olvida.
2. **Siempre** termina la semana con un writeup publicado (aunque sea en tu repo privado).
3. **Snapshots antes de cada lab.** Nunca entres en una máquina sin snapshot.
4. **Sesgo hacia la práctica:** si te pasas 2 horas leyendo sin tocar nada, es un día perdido.

A.2 Calendario semana a semana (año 1)

TRIMESTRE 1 — FASE 0 (SEMANAS 1-12, ~150 H)

SEM	FOCO	ENTREGABLE
1	Historia de la seguridad, CIA, terminología. Empieza el repo de notas.	Tabla de 5 vulnerabilidades históricas
2	**Redes I**: OSI/TCP-IP, IP, subnetting, CIDR, ARP, Ethernet	20 problemas de subnetting resueltos
3	**Redes II**: TCP/UDP, puertos, DNS, DHCP, NAT, routing	Diagrama de tu red + `dig` / `nslookup` documentados
4	**Redes III**: HTTP/HTTPS/TLS, proxies, firewalls, VPN, IDS/IPS	Captura Wireshark de un TLS handshake exegética
5	**Montar lab de 3 VMs** y romper/abrir puertos	Lab funcionando documentado
6	**Linux I**: FHS, permisos, chmod/chown, SUID, usuarios	Bitácora de comandos

SEM	FOCO	ENTREGABLE
7	**Linux II**: grep/awk/sed/find, procesos, systemd, cron, SSH	40 ejercicios de shell resueltos
8	**Linux III**: bash scripting + OverTheWire Bandit 0-10	Bandit 0-10 con writeups
9	**Windows I**: NTFS, ADS, ACLs, procesos, servicios, registro	Tabla de permisos NTFS
10	**Windows II**: PowerShell (lo importante) + Event Viewer	Script PowerShell que exporta CSV y programa una tarea
11	**Sysmon** + instalación + captura de eventos	20 Event IDs catalogados
12	**Criptografía** + CryptoHack + revisión del trimestre	CryptoHack: Classical, Encoding, Hashes

TRIMESTRE 2 — FASE 1 PARTE A (SEMANAS 13-24, ~180 H)

SEM	FOCO	ENTREGABLE
13	**Web fundamentals** + HTML/JS mínimo + DevTools	Explicar un flujo HTTP completo
14	**OWASP Top 10** + PortSwigger Academy: SSRF, SSTI, OS command injection	8 labs
15	**SQLi** (la más importante) — intro + UNION + boolean/time blind	10 labs + 1 writeup largo
16	**SQLi avanzado** + NoSQL + second-order	8 labs + writeup
17	**XSS** — reflected, stored, DOM, blind	10 labs
18	**CSRF + Access Control/IDOR + Broken Auth**	10 labs
19	**Path Traversal / LFI / RFI / SSRF** + Information disclosure	10 labs
20	**Burp Suite a fondo** (Intruder, Repeater, Extensiones) + OWASP ZAP	3 workflows automatizados
21	**Nmap a fondo** + enumeración de SMB/LDAP/SNMP	Informe de enumeración de tu lab
22	**Cracking** (john + hashcat) + Hydration de credenciales en lab	20 hashes rotos
23	**Privesc Linux** (linpeas, SUID, cron, sudo, GTFOBins)	3 máquinas HTB Linux
24	**Privesc Windows** (winPEAS, token, UAC, servicios)	2 máquinas HTB Windows + informe PDF del trimestre

TRIMESTRE 3 — FASE 1 PARTE B (SEMANAS 25-30) + FASE 2 INICIO (SEMANAS 31-40, ~190 H)

SEM	FOCO	ENTREGABLE
25-26	**PortSwigger restante**: deserialización, smuggling, business logic, JWT, OAuth	20 labs
27-28	**Informe profesional de pentest** + **primer informe de pentest real** (HTB Pro Lab o una app DVWA)	Informe PDF de 20 páginas
29-30	**Repaso + primer AD lab** (montar DC) + BloodHound intro	AD funcionando + Grafo de BloodHound
31-32	**Kerberos/NTLM** a fondo: Kerberoasting, AS-REP roast, PtH	3 writeups de ataques Kerberos
33-34	**DCSync + ACL abuse + GPO abuse + delegation**	Commit de Domain Admin documentado
35-36	**ADCS (ESC1-8)** + Certipy + detección	Writeup de ADCS

SEM	FOCO	ENTREGABLE
37-38	**GOAD completo** (levantar y comprometer)	Writeup de 3000+ palabras
39-40	**Blue team: Sigma, Atomic Red Team, Wazuh**	15 reglas Sigma probadas

TRIMESTRE 4 — FASE 2 (SEMANAS 41-52, ~180 H)

SEM	FOCO	ENTREGABLE
41-42	**Exploit dev**: stack overflow, ROP, pwntools	Exploit propio funcional
43-44	**Malware analysis estático** (PE/ELF, DiE, capa, FLOSS, .NET)	Informe de malware estático
45-46	**Malware analysis dinámico** (x64dbg, WinDbg, sandbox)	Informe de malware dinámico
47-48	**Cloud security** (AWS/Azure) + CloudGoat	Informe de pentest cloud
49-50	**Kubernetes security** + kube-hunter	Compromiso de un cluster
51-52	**Forense avanzado** (Volatility, NTFS, timeline) + **KPIs + planificación año 2**	Informe forense completo + plan año 2

A.3 Plan de 4 años a 20 h/semana (1000+ h/año)

CODE0

A.4 Sistema de seguimiento (usa Notion, Obsidian o un simple Markdown)

Crea una base de datos con estas columnas y **actualízala cada domingo**:

CODE1

- **KPIs mensuales**: horas acumuladas, labs resueltos, writeups escritos, certificaciones.
- **Revisión trimestral**: ¿estoy en la fase que toca? ¿he hecho todos los entregables? ¿qué se me atasca?
- **Objetivo de ratio**: en 12 meses → **1.200-1.400 h**, **60+ labs/CTFs**, **50+ writeups**, **2-3 certificaciones**, **5+ proyectos completos**.

PARTE B — Catálogo de recursos gratuitos (con enlaces)

> Regla: **el 80 % de tu tiempo debe estar en recursos "hands-on"** (labs, plataformas, writeups). Los cursos teórico son solo el andamiaje.

B.1 Plataformas de práctica (tu terreno de juego)

PLATAFORMA	QUÉ ES	ENLACE	NOTAS
PortSwigger Web Security Academy	La mejor formación web gratuita del mundo. 17 temas, ~250 labs, writeups oficiales	`https://portswigger.net/web-security`	**EMPIEZA POR AQUÍ. Obligatorio.**
Hack The Box	Máquinas, retos, Pro Labs, Academy, CVE labs, CPTS exam	`https://www.hackthebox.com`	Requiere suscripción para Pro Labs, pero hay muchas máquinas free.
TryHackMe	Rutas guiadas, muy buena para el inicio, paths completos	`https://tryhackme.com`	Jr Pentester path, SOC L1 path.
picoCTF / picoGym	Retos de navegación, niveles Wakeup a through	`https://picoctf.org`	Ideal para principiantes.
OverTheWire	Wargames SSH classics	`https://overthewire.org/wargames/`	Bandit, Natas, Krypton, Advent of Code.
pwn.college	Curso de binary exploitation y reversing (gratuito, ASU)	`https://pwn.college`	El mejor para exploit dev.
CyberDefenders	Retos de forensics y blue team	`https://cyberdefenders.com`	Labs de DFIR.
Blue Team Labs Online (BTLO)	Retos de detección y DFIR	`https://blueteamlabs.online`	
LetsDefend	Path de SOC Analyst (hay free tier)	`https://letsdefend.io`	
Splunk BOTS	Datasets de SIEM para IR (Boss of the SOC)	`https://www.splunk.com/en_us/training`	
OffSec Proving Grounds (PG)	Máquinas de práctica (hay free tier y مُسلم)	`https://portal.attackdefense.com`	**GRATIS en 2024-2025**, muy parecido a la experience real.
CTFtime	Calendario de CTFs, equipos	`https://ctftime.org`	Registrate y participa en CTFs online.
VulnHub	Máquinas virtuales vulnerables para descargar	`https://www.vulnhub.com`	Perfecto para tu lab local.

B.2 Recursos de web / AppSec

- **OWASP Web Security Testing Guide (WSTG)**: `https://owasp.org/www-project-web-security-testing-guide/`

- **OWASP Cheat Sheet Series** (el mejor punto de referencia rápida): `https://cheatsheetseries.owasp.org/`
- **OWASP Top 10** (y **Top 10 for LLM Apps 2025**): `https://owasp.org/www-project-top-10/`
- **PortSwigger Research Blog** (los mejores writeups de web): `https://portswigger.net/research`
- **books.bugcrowd.com** (Web Security Handbook online gratis): `https://books.bugcrowd.com`
- **HackTricks** (compendio gigantesco de todo, gratis, muy útil de referencia): `https://book.hacktricks.wiki` · repo: `https://github.com/carlosrodicio/hacktricks`
- **PayloadsAllTheThings** (cheat sheets de payloads, de referencia diaria): `https://github.com/swisskyrepo/PayloadsAllTheThings`
- **0x00ffsec** (writeups completos de HTB, los mejores): `https://0x00ffsec.com`
- **The Hacker Recipes** (`https://thehacker.recipes`)
- **PentestMonkey** (tiene contenido gratuito visible aunque el curso sea de pago): `https://pentestmonkey.com`
- **SecLists** (wordlists): `https://github.com/danielmiessler/SecLists`

B.3 Recursos de red

- **Kurose & Ross Top-Down** (PDF gratis del autor): `https://gaia.cs.umass.edu/kurose_ross/index.php`
- **PacketLife Cheat Sheets**: `https://packetlife.net/campus/cheat-sheets/`
- **NetSecNotes** (diagramas): `https://www.netsecnotes.com`
- **Wireshark User Guide** (oficial): `https://www.wireshark.org/docs/`
- **Prof. Messer (Network+/Windows server)**: `https://www.professormesser.com`
- **NetworkChuck**: `https://www.youtube.com/@NetworkChuck`
- **David Bombal** (labs de red gratis): `https://www.youtube.com/@davidbombal`
- **Practical Networking Handbook** (gratis online): `https://www.practicalnetworking.com` (busca el PDF más reciente)

B.4 Recursos de Active Directory (AD)

- **ired.team** (notas de ataque a AD, de S4vitar — **EXCELENTE**): `https://ired.team`
- **GOAD (Orange Cyberdefense)**: `https://github.com/Orange-Cyberdefense/GOAD`
- **AD-Miner**: `https://github.com/Orange-Cyberdefense/AD-Miner`
- **Orange Cyberdefense research blog**: `https://www.orangecyberdefense.com/blog/`
- **BloodHound (SpecterOps)**: `https://github.com/SpecterOps/BloodHound` · guía: `https://bloodhound.specterops.io`
- **SharpHound Rules**: `https://github.com/ZeroLegion/SharpHoundRules`
- **BloodHound Useful Queries**: `https://github.com/BloodHoundAD/BloodHound/wiki/Useful-Queries`
- **Orange Cyberdefense: "Adversary Emulation with GOAD"**: parte de su blog
- **Ansible for AD (automation)**: `https://github.com/Orange-Cyberdefense/GOAD/tree/ansible-ad`

- **ADSecurity.org** (escrita por Timothy Jones, de Microsoft): `https://adsecurity.org`
- **SANS whitepapers sobre AD**: `https://www.sans.org/white-papers/`
- **certipy (ESC1-8)**: `https://github.com/ly4k/Certipy`
- **Rubeus**: `https://github.com/GhostPack/Rubeus`
- **CESP (Pivoting)**: `https://github.com/Orange-Cyberdefense/GOAD`
- **orange Cyberdefense GOAD writeups** y **S4vitar's Blog**: `https://s4vitar.github.io` (S4vitar también es el autor de ired.team)

B.5 Recursos de Exploit Development / Binary

- **pwn.college**: `https://pwn.college`
- **OpenSecurityTraining2** (gratis, x86-64): `https://www.ostraining.org`
- **Practical RE** (Dennis Andriesse, repo del libro): `https://github.com/denisandriesse/practical-re`
- **Exploit Education**: `https://exploit.education`
- **Irrelevant (Narnia, protostar, etc.)**: `https://irrelevant.ch`
- **Hacking: The Art of Exploitation** (el mejor libro de exploit dev, de pago pero imprescindible): `https://www.nostarch.com/hacking-the-art-of-exploitation-2nd-edition/`
- **Buffer Overflow Prep (NCC)**: `https://guyinatuxedo.github.io/`
- **Shellphish (repo legal)**: `https://github.com/shellphish/shellphish`
- **pwntools docs**: `https://docs.pwntools.com`

B.6 Recursos de malware analysis / reversing

- **Ghidra** (NSA, gratis, profesional): `https://github.com/NationalSecurityAgency/ghidra`
- **Cutter/Rizin** (libre): `https://cutter.re` · `https://rizin.re`
- **x64dbg**: `https://x64dbg.com`
- **capa** (Mandiant): `https://github.com/mandiant/capa`
- **FLOSS**: `https://github.com/mandiant/flare-floss`
- **Detect It Easy**: `https://github.com/horsicq/Detect-It-Easy`
- **PEStudio**: `https://winitor.com`
- **VirusTotal** (gratis para análisis): `https://www.virustotal.com`
- **MalwareBazaar** (samples gratis de malware): `https://bazaar.abuse.ch`
- **Triage** (sandbox gratis): `https://tria.ge`
- **ANY.RUN** (sandbox gratis con límites): `https://any.run`
- **CAPE Sandbox** (puedes auto-hospedar): `https://github.com/kevoreilly/CAPE`
- **Practical Malware Analysis** (paid pero imprescindible) — se consigue en bibliotecas o de segunda mano
- **Google SRE book** (gratis online, excelente): `https://sre.google/sre-book/table-of-contents/`
- **dfir.report** (reports de incidentes reales, con IOCs): `https://dfir.report`
- **Malware Analysis Principles and Practices** (no gratis, pero es la biblia)

- **Internet Storm Center (SANS)**: `https://isc.sans.edu`

B.7 Recursos de Blue Team / Detection

- **MITRE ATT&CK**: `https://attack.mitre.org` · Navigator: `https://mitre-attack.github.io/attack-navigator/`
- **MITRE D3FEND**: `https://d3fend.mitre.org`
- **MITRE Engage**: `https://engage.mitre.org`
- **Sigma**: `https://github.com/SigmaHQ/sigma`
- **YARA**: `https://github.com/VirusTotal/yara`
- **Atomic Red Team**: `https://github.com/redcanaryco/atomic-red-team`
- **MITRE Caldera**: `https://github.com/mitre/caldera`
- **Stratus Red Team** (cloud): `https://github.com/redcanaryco/stratus-red-team`
- **Cloud-Red-Team** (AWS): `https://github.com/salesforce/Cloud-Red-Team`
- **Elastic detection rules**: `https://github.com/elastic/detection-rules`
- **Splunk Security Content**: `https://security.splunk.com`
- **The Hunting Handbook** (Robert M. Lee, gratis): `https://www.happyhacking.co/hunting-handbook`
- **Wazuh**: `https://wazuh.com`
- **Velociraptor** (DFIR OS): `https://github.com/Velocidex/velociraptor`
- **TheHive** (gestión de incidentes OSS): `https://github.com/TheHive-Project`
- **MISP** (inteligencia de amenazas OSS): `https://github.com/MISP/MISP`
- **Shuffle** (SOAR OSS): `https://github.com/Shuffle/Shuffle`
- **SANS Reading Room** (whitepapers gratuitos de calidad): `https://www.sans.org/reading-room/`
- **Red Canary Threat Detection Hub** (descarga de packs Sigma/YARA): `https://redcanary.com/threat-detection-hub`
- **AlienVault OTX**: `https://otx.alienvault.com` (gratis)

B.8 Recursos de Forense

- **Sleuth Kit / Autopsy**: `https://www.sleuthkit.org` · `https://www.autopsy.com`
- **Volatility 3**: `https://github.com/volatilityfoundation/volatility3`
- **KAPE**: `https://github.com/KAPE/KAPE`
- **Cyber Triage**: `https://cybertriage.com`
- **Plaso/log2timeline**: `https://github.com/log2timeline/plaso`
- **Timesketch**: `https://github.com/google/timesketch`
- **Digital Corpora** (imágenes forenses): `https://digitalcorpora.org`
- **NIST CFReDS** (imágenes forenses del NIST): `https://cfreds.nist.gov`
- **Cellebrite** (tiene material público), **ForensicFocus**: `https://www.forensicfocus.com`
- **Velociraptor** (ya listado): `https://docs.velociraptor.app`

B.9 Recursos de Cloud Security

- **AWS Skill Builder** (gratis): `https://skillbuilder.aws`
- **Google Cloud Skills Boost** (free tier): `https://cloudskillsboost.google`
- **Microsoft Learn** (gratis): `https://learn.microsoft.com/es-es/training/`
- **CloudGoat** (AWS): `https://github.com/WithSecureLabs/cloudgoat`
- **Verrus** (AWS/Azure): `https://github.com/salesforce/verrus`
- **pwncloud** (AWS): `https://github.com/RhinoSecurityLabs/pwncloud`
- **Kubernetes Security** (documentación): `https://kubernetes.io/docs/concepts/security/`
- **kube-hunter**: `https://github.com/aquasecurity/kube-hunter`
- **KICS** (IaC scanning): `https://github.com/Checkmarx/KICS`
- **kube-bench** (CIS benchmarks): `https://github.com/aquasecurity/kube-bench`
- **Prowler** (cloud compliance): `https://github.com/prowler-cloud/prowler`
- **Awesome Cloud Security**: `https://github.com/irleks/awesome-cloud-security`
- **Kubernetes Academy / LFS258** (gratis): `https://training.linuxfoundation.com/training/`

B.10 Recursos de criptografía

- **CryptoHack** (interactivo, gratis): `https://cryptohack.org`
- **cryptopals**: `https://cryptopals.com`
- **Practical Cryptography** (Daniel Boneh & Pratik Prakash, gratis online): `https://practicalcryptography.com`
- **A Graduate Course in Applied Cryptography** (Dan Boneh, PDF gratis): `https://crypto.stanford.edu/~dabo/courses/OnlineCrypto/`
- **Coursera Cryptography I/II** (Dan Boneh, audit gratis): `https://www.coursera.org/learn/cryptography`
- **Modern Cryptography** (Kokubo, abierto): `https://book.masden.org` (busca el repo oficial)
- **PortSwigger "Encryption Attacks"** (labs): parte de la Web Security Academy
- **CryptoHack + CryptoPals** juntos cubren el 80 % de lo que necesitarás

B.11 Canales de YouTube recomendados

CANAL	IDIOMA	CONTENIDO
NetworkChuck	EN	Redes, hacking, Docker, Linux (energía alta)
John Hammond	EN	Malware, CTFs, pentest, threat actor reports
IppSec	EN	Walkthroughs de HTB (los mejores writeups en vídeo)
LiveOverflow	EN	Teoría de bajo nivel, cripto, exploiting
The Cyber Mentor	EN	Harness/eJPT, pentesting práctico, fundamentos ofensivos
Professor Messer	EN	Network+, Linux, Windows Server (fundamentos sólidos)
David Bombal	EN	Redes, Kali, labs practicales
HackerSploit	EN	Practical hacking, pentest práctico

CANAL	IDIOMA	CONTENIDO
0xFFSEC	EN	Writeups de HTB
PwnFunction	EN	Exploit dev, pwn, pwntools (excelente)
LowLevelLearning	EN	x86-64 assembly, exploit dev
S4vitar	ES	Android/Flutter security, OSINT, AD (el ired.team es suyo)
Demente0	ES	Explotación binaria, pwn, ROP (en español, muy bueno)
Machines Learning	EN	Con IA, research
Kevin Beato / Vergilius Project	EN	Ghidra, reversing
Simplify by simpleX	EN	Programación, pero útil para scripts de pentest

Canales en español de referencia (calidad variable, peroJX verify antes de studying):

- **S4vitar** (España) — el más serio y técnico
- **Demente0** (España) — binario/exploit dev
- **El Instructor / Juan Ignacio** (España) — hacking ético general
- **Ciberseguridad CCN-CERT** — videos de concienciación
- **INCIBE** — videos y guías en español

B.12 Certificaciones gratuitas y de bajo coste (recompiladas)

CERTIFICACIÓN	COSTE	CUÁNDO
Fortinet NSE 1 + NSE 2	100 % gratis	Fase 0-1
Cisco Introduction to Networking (Skills for All)	Gratis	Fase 0
Microsoft SC-900	Gratis (con Microsoft Learn)	Fase 0-1
Palo Alto Cybersecurity Academy 10x01/10x11	Gratis	Fase 1
Check Point CC	Gratis	Fase 1
AWS Cloud Practitioner	Gratis (exam voucher)	Fase 2
Google Cloud Digital Leader	Gratis (exam voucher)	Fase 2
Kubernetes Fundamentals (LFS258)	Gratis (Linux Foundation/edX)	Fase 2
Introduction to Linux (LFS158x)	Gratis (Linux Foundation/edX)	Fase 0
SANS CyberDefenders (labs)	Gratis	Fase 1-2
SANS Reading Room	Gratis	Siempre
Google Cybersecurity Professional Certificate	~$49/mes (7 meses)	Fase 0-1
eJPT (INE)	~$100-250	Fase 1
CPTS (Hack The Box)	~$210	Fase 1-2
eCPT (INE)	~$400	Fase 2
CompTIA Security+ / PenTest+ / CySA+	~$350-500 (vouchers bajan a $150)	Fase 1-2
CRTO (ZeroPointSecurity)	~$500	Fase 3
OSCP (OffSec)	~$1599 + 30 días	Fase 3
OSEP (OffSec)	~$2000 (becas: ~$500)	Fase 4

CERTIFICACIÓN	COSTE	CUÁNDO
CISSP / CISM	~$700-1000 (5 años exp)	Fase 4

B.13 Libros recomendados (los mínimos imprescindibles)

LIBRO	TEMA	GRATIS/PRECIO
Kurose & Ross — Top-Down Networking	Redes	Gratis (autor)
The Linux Command Line (Shotts)	Linux	Gratis (`linuxcommand.org`)
Linux From Scratch	Linux profundo	Gratis (`linuxfromscratch.org`)
Practical Cryptography (Boneh)	Cripto	Gratis online
The Web Application Hacker's Handbook (Stuttard & Pinto)	Web pentest	Libro (la web está por partes: usa PortSwigger Academy)
Hacking: The Art of Exploitation (Stevens & Aitel)	Exploit dev	Libro (esencial)
The Hacker's Handbook	General	Libro (clásico)
Red Team Field Manual (RTFM)	Red team ops	Libro (libre)
Blue Team Handbook: Incident Response Edition	Blue team / IR	**Gratis** (PDF, de DFIR Report)
Hunting Handbook (R. M. Lee)	Threat hunting	Gratis
Practical Packet Analysis (Wireshark)	Análisis de red	Libro
Malware Analysis: The Big Picture	Malware	**Gratis** (VirusTotal)
Secure Coding 101	General	Multiple (OWASP)
The Art of Computer Virus Research and Defense	Malware	Libro (esencial en malware)

B.14 Referencias de normativa / Compliance

- **CIS Benchmarks**: `https://www.cisecurity.org/benchmark`
- **CIS Controls**: `https://www.cisecurity.org/controls`
- **NIST Cybersecurity Framework 2.0**: `https://www.nist.gov/cyberframework`
- **ISO 27001/27701** (implementación, no necesitas el estándar)
- **ENS (España)**: `https://www.incibe.es/seguridad-ciber/ens`
- **NIS2**: `https://digital-strategy.ec.europa.eu`
- **DORA** (resiliencia financiera): `https://www.esma.europa.eu`
- **CSA (Cloud Security Alliance)**: `https://cloudsecurityalliance.org`

PARTE C — Cómo no abandonar (la parte que nadie cuenta)

1. **El enemigo real es el "learn-while-doing" mal gestionado:** Estate 5 horas leyendo sin tocar nada y te sientes productivo, pero no avanzas. **Solución:** la regla del 50/50 en la sesión.
2. **Es un maratón de 3 años.** La constancia gana a la intensidad. 2h/día > 12h el domingo.
3. **Las mesetas son normales.** Viene un día donde no entiendes Kerberos, o donde un lab te lleva 5 h. **La solución:** mira la solución/writeup, entiende por qué no lo viste, anótalo, sigue. No abandones la semana.
4. **No busques la perfección.** Entiende el 70 % de un tema, pasa al siguiente, vuelve a los 3 meses. El relleno se rellena.
5. **Documenta desde hoy.** Tu repo de notas es lo que te va a diferenciar en una entrevista.
6. **Busca comunidad:** en Twitter/X con #cybersecurity, en Discord de HTB/THM, en r/netsec, en r/ReverseEngineering, y en **foros**. Estudiar solo es más lento.
7. **Compromiso de entrevista:** cuando llegues a Fase 2, empieza a hacer entrevistas aunque no te sientas listo. La práctica de entrevistas es una habilidad propia.
8. **No te aferrantes a la certificación al principio.** Las certificaciones no importan nada hasta la Fase 2-3. Antes, conocimiento puro.
9. **Automatiza lo repetitivo:** cada tarea repetida (encender VMs, montar snapshot, bajar labs) debería ser un script. Empieza AHORA.
10. **Celebra los hitos:** cada 20 labs, cada 100 h, cada cert → haz algo que te guste. La sostenibilidad importa.

PARTE D — Resumen ejecutivo (para tener a mano)

CODE0