Ethical Hacking Starter Lab

De zéro à tes premiers labs :
méthodologie, outils et pratique encadrée

KadoHackLabs

Édition 2026 — v1.0

Usage strictement éthique et légal — voir la règle absolue

Bienvenue dans le Lab

Ce guide n'est pas un cours théorique. C'est un atelier. Tu vas installer un environnement de test, lancer tes premières commandes, observer des résultats réels, et comprendre pourquoi chaque chose fonctionne.

À la fin, tu auras un portfolio de labs que tu peux montrer, et une méthodologie que tu peux réutiliser sur n'importe quelle cible autorisée.

> Rejoins la communauté KadoHackLabs pour poser tes questions et recevoir les nouveaux labs : t.me/KadoTech_IA_Cyber

À qui s'adresse ce guide

- Débutant complet en cybersécurité qui ne sait pas par où commencer.
- Étudiant ou junior en reconversion qui veut de la pratique, pas des slides.
- Technicien réseau ou support qui veut monter en compétence sur la sécurité.
- Toute personne qui veut comprendre l'offensive pour mieux défendre.

Ce que tu vas savoir faire

- Monter un laboratoire virtuel isolé et légal (VirtualBox + VMs vulnérables).
- Cartographier un réseau et découvrir des hôtes avec Nmap.
- Identifier services, versions et systèmes d'exploitation exposés.
- Capturer et analyser du trafic réseau avec Wireshark et tcpdump.
- Exploiter une faille web volontairement vulnérable (injection SQL, XSS).
- Mener une reconnaissance locale et une escalade de privilèges sous Linux.
- Rédiger un rapport d'audit court, structuré et professionnel.

RÈGLE ABSOLUE — À LIRE AVANT TOUT

Tu ne testes QUE : (1) tes propres machines, (2) des machines virtuelles que tu as installées toi-même, (3) des plateformes d'entraînement légales (TryHackMe, Hack The Box, VulnHub, OWASP Juice Shop), (4) des cibles pour lesquelles tu disposes d'une autorisation écrite explicite.

Scanner, tester ou exploiter une machine qui ne t'appartient pas et sans autorisation est un délit dans la quasi-totalité des pays — au Bénin comme en France, aux États-Unis ou ailleurs. Aucune exception, aucune « juste pour voir ». Ce guide te donne les compétences ; à toi d'en faire un usage éthique et légal.

Module 1 — Monter ton laboratoire

Avant de toucher à un outil offensif, tu dois avoir un terrain de jeu isolé. C'est la fondation de tout le reste. Un lab bien construit, c'est la garantie que tu peux tout casser sans conséquence.

1.1 — Pourquoi un environnement isolé

Une machine virtuelle vulnérable est, par définition, remplie de failles exploitables. Si elle est connectée à ton réseau domestique ou à Internet, n'importe qui peut la prendre et rebondir vers tes autres machines. Le lab doit donc être isolé : réseau « host-only » ou « internal network », jamais en pont avec ton réseau réel.

1.2 — Les outils à installer

Outil	Rôle	Où
VirtualBox	Hyperviseur (fait tourner les VMs)	virtualbox.org
VMware Workstation Player	Alternative à VirtualBox	vmware.com
Kali Linux	Ta machine d'attaque	kali.org
Metasploitable 2	Cible volontairement vulnérable	sourceforge.net
OWASP Juice Shop	Cible web moderne (Node.js)	owasp.org
DVWA	Application web vulnérable didactique	dvwa.co.uk

1.3 — Installer Kali Linux

Kali est la distribution de référence pour les tests d'intrusion. Tu peux l'utiliser en machine virtuelle ou en live USB. Pour ce guide, prends la version VM « VirtualBox » officielle.

1. Télécharge l'image VirtualBox de Kali sur kali.org/get-kali.
2. Importe le fichier .ova dans VirtualBox : Fichier → Importer un appareil virtuel.
3. Alloue au moins 2 CPU, 4 Go de RAM et 40 Go de disque.
4. Démarre la VM. Identifiants par défaut : kali / kali.
5. Change immédiatement le mot de passe : passwd.
6. Mets à jour : sudo apt update && sudo apt full-upgrade -y.

1.4 — Configurer le réseau isolé

C'est l'étape que les débutants sautent, et c'est celle qui compte le plus. Tu vas créer un réseau privé entre tes VMs, sans accès à ton réseau réel.

1. Dans VirtualBox : Outils → Réseau → créer un réseau « interne ».
2. Sur chaque VM : Configuration → Réseau → Adaptateur 2 → « Réseau interne » (intnet).
3. Démarre Kali et la cible. Note leurs adresses IP avec ip a.
4. Vérifie qu'elles se voient : ping <ip_de_l_autre>.
5. Vérifie qu'elles ne sortent PAS sur Internet : ping 1.1.1.1 doit échouer.

> Si le ping vers 1.1.1.1 fonctionne, ton isolation est ratée. Reprends l'étape 2 et retire tout adaptateur en mode « pont » (bridged).

1.5 — Premier contact : reconnaître son environnement

Avant d'attaquer quoi que ce soit, tu dois savoir où tu es. Sur Kali, ouvre un terminal et lance :

```
ip a                  # tes interfaces et adresses
ip route              # ta table de routage, la passerelle
cat /etc/hosts        # résolution locale
cat /etc/resolv.conf  # serveurs DNS utilisés
whoami && id          # ton utilisateur et ses groupes
```

Comprendre ces quatre commandes te servira dans tous les labs suivants. Note l'adresse de ton interface réseau : c'est ton point d'ancrage.

Module 2 — Reconnaissance réseau avec Nmap

Nmap est l'outil le plus important de ta boîte à outils. Il répond à une question simple : « qui est là, et qu'est-ce qui est ouvert ? ». Maîtriser Nmap, c'est maîtriser la première phase de toute évaluation de sécurité.

2.1 — Découverte d'hôtes : qui est sur le réseau ?

Le « ping sweep » envoie une sonde à chaque adresse d'une plage et liste celles qui répondent. C'est ta toute première commande face à un réseau inconnu.

```
nmap -sn 192.168.1.0/24
```

L'option -sn (ex -sP) désactive le scan de ports : tu veux juste savoir qui est vivant. Sur ton lab, tu dois voir ton Kali et ta cible. Rien d'autre.

2.2 — Le scan SYN : rapide et discret

Le scan SYN (ou « half-open ») est le scan par défaut de Nmap. Il n'achève jamais la connexion TCP, ce qui le rend rapide et moins visible dans certains journaux.

```
nmap -sS 192.168.1.10
nmap -sS -p 1-1000 192.168.1.10
nmap -sS -p- 192.168.1.10        # les 65535 ports (long)
```

Sous le capot : Nmap envoie un paquet SYN. Si la cible répond SYN/ACK, le port est ouvert ; si elle répond RST, il est fermé ; sans réponse, il est filtré. Nmap envoie ensuite un RST pour couper proprement.

2.3 — Les six états d'un port

État	Signification
open	Un service écoute et accepte les connexions.
closed	Aucun service, mais le port est joignable (il répond).
filtered	Un pare-feu bloque la sonde. Nmap ne peut pas conclure.
unfiltered	Le port répond mais Nmap ne sait pas s'il est ouvert ou fermé.
open|filtered	Ambigu : impossible de trancher (fréquent en scan UDP).
closed|filtered	Ambigu également (rare).

2.4 — Détecter les services et les versions

Savoir qu'un port est ouvert ne suffit pas. Tu veux savoir QUEL logiciel tourne et QUELLE version — car c'est la version qui détermine la faille.

```
nmap -sV 192.168.1.10
nmap -sV --version-intensity 9 192.168.1.10
```

La sortie ressemble à : 22/tcp open ssh OpenSSH 7.2p2 Ubuntu. Cette précision est ce qui te permet de chercher une vulnérabilité connue pour cette version précise.

2.5 — Détecter le système d'exploitation

```
nmap -O 192.168.1.10
nmap -O --osscan-guess 192.168.1.10   # plus agressif
```

La détection d'OS compare des empreintes de la pile TCP/IP. Elle n'est jamais fiable à 100 % : Nmap donne une probabilité. Traite-la comme un indice, pas comme une vérité.

2.6 — Le scan UDP

Beaucoup de services critiques tournent en UDP (DNS sur 53, SNMP sur 161, DHCP sur 67-68). Le scan UDP est lent, car UDP n'a pas de poignée de main : l'absence de réponse est ambiguë.

```
sudo nmap -sU --top-ports 50 192.168.1.10
```

2.7 — Scripts NSE : automatiser la détection

Le Nmap Scripting Engine (NSE) exécute des scripts Lua pour aller plus loin : inventaire, détection de vulnérabilités, découverte de partages.

```
nmap --script default 192.168.1.10
nmap --script vuln 192.168.1.10
nmap --script smb-enum-shares -p 445 192.168.1.10
nmap --script http-title -p 80,443 192.168.1.10
```

> Le script 'vuln' est bruyant : il peut faire planter un service fragile. À n'utiliser que sur des cibles d'entraînement, jamais en production.

2.8 — Le LAB : cartographier la cible

Objectif : produire une cartographie complète de ta VM vulnérable.

1. Trouve la cible : nmap -sn <plage_de_ton_lab>
2. Scan complet des ports : nmap -p- -T4 <cible>
3. Détection services et OS : nmap -sV -O -p <ports_ouverts> <cible>
4. Scripts de découverte : nmap --script default -p <ports_ouverts> <cible>
5. Note chaque port, chaque service, chaque version dans un tableau.

Livrable : un tableau à trois colonnes (Port / Service / Version). C'est la matière première de tout le reste du guide.

Module 3 — Analyse de paquets avec Wireshark

Dans un réseau, les journaux mentent, les utilisateurs se trompent, mais les paquets ne mentent jamais. Wireshark te donne accès à la vérité brute : ce qui circule réellement sur le fil.

3.1 — Capturer le bon trafic

Wireshark capture tout ce que voit ton interface. Sur un réseau commuté moderne, tu ne verras que ton propre trafic (et le broadcast) — c'est normal et sain.

```
sudo wireshark            # lancer en root pour capturer
sudo tcpdump -i eth0 -w capture.pcap   # capture en ligne de commande
sudo tcpdump -i eth0 -nn port 80       # HTTP seul
```

Le format .pcap (ou .pcapng) est universel : tu peux capturer avec tcpdump sur un serveur et analyser avec Wireshark sur ton poste.

3.2 — Les filtres d'affichage qui servent vraiment

Il y a deux familles de filtres : les filtres de capture (BPF, appliqués avant la capture) et les filtres d'affichage (appliqués après). Voici ceux que tu utiliseras tous les jours.

Filtre d'affichage	Ce qu'il montre
ip.addr == 192.168.1.10	Tout le trafic vers/depuis cette IP
tcp.port == 443	Le trafic HTTPS
http.request	Uniquement les requêtes HTTP
dns	Tout le trafic DNS
tcp.analysis.flags	Problèmes TCP : pertes, retransmissions
tcp.flags.syn == 1 && tcp.flags.ack == 0	Les tentatives de connexion (SYN)
http.response.code >= 400	Les erreurs HTTP

3.3 — Le handshake TCP, vu de tes yeux

Ouvre une capture, filtre sur un flux et cherche la connexion en trois temps. C'est la respiration de tout Internet.

- SYN : le client demande « on se parle ? » (flag SYN).

- SYN/ACK : le serveur répond « ok, et toi ? » (flags SYN + ACK).
- ACK : le client confirme « parfait » (flag ACK).

Filtre utile pour isoler ces paquets : tcp.flags.syn == 1. Puis clic droit → Follow → TCP Stream pour voir la conversation complète en clair.

3.4 — Repérer les problèmes réseau

Les fameux « ça rame » se voient noir sur blanc dans Wireshark. Cherche les retransmissions, les doublons d'ACK et les fenêtres à zéro.

Symptôme (champ Expert)	Cause probable
Retransmission	Perte de paquet, congestion, lien instable
Duplicate ACK	Paquet perdu détecté par le récepteur
Zero Window	Le récepteur est saturé, il ne peut plus recevoir
TCP Previous segment not captured	Capture incomplète ou perte réelle

> Menu Analyse → Expert Information : Wireshark classe lui-même les anomalies par gravité. C'est ton point de départ pour un diagnostic.

3.5 — Le LAB : diagnostic d'un problème réseau

Objectif : prouver, capture à l'appui, qu'un problème vient du réseau et non de l'application.

1. Lance une capture sur ton interface pendant un transfert de fichier.
2. Arrête la capture. Filtre sur le flux concerné.
3. Cherche les retransmissions (tcp.analysis.retransmission).
4. Mesure le temps entre le premier SYN et le dernier ACK.
5. Ouvre Analyse → Expert Information et note les 3 alertes principales.
6. Conclusion : le problème est-il réseau (retransmissions) ou applicatif ?

Module 4 — Les failles web fondamentales

La majorité des attaques réelles visent des applications web. Ce module s'attaque aux deux failles les plus pédagogiques : l'injection SQL et le XSS. Tout se passe sur des applications faites POUR être vulnérables.

4.1 — Installer les cibles légales

Deux applications d'entraînement, maintenues et légales :

- DVWA (Damn Vulnerable Web Application) : PHP/MySQL, réglage de difficulté, parfait pour comprendre les bases.
- OWASP Juice Shop : application moderne Node.js, plus proche du réel, avec un tableau de bord de progression.

```
# Juice Shop via Docker (le plus simple)
docker run --rm -p 3000:3000 bkimminich/juice-shop
# puis ouvre http://localhost:3000
```

> Ces applications sont conçues pour être vulnérables. Ne déploie JAMAIS ce type d'application sur un serveur exposé à Internet.

4.2 — Injection SQL : comprendre avant d'exploiter

Une injection SQL arrive quand une entrée utilisateur est concaténée dans une requête sans être neutralisée. L'attaquant peut alors modifier la logique de la requête.

Requête normale attendue par l'application :

```
SELECT * FROM users WHERE username = 'admin' AND password = 'secret';
```

Si l'utilisateur saisit admin' -- dans le champ, la requête devient :

```
SELECT * FROM users WHERE username = 'admin' -- ' AND password = 'secret';
```

Le -- commente la fin : la vérification du mot de passe disparaît. C'est le principe de base, à comprendre puis à reproduire sur DVWA.

4.3 — XSS : quand ton navigateur exécute le code de l'autre

Le Cross-Site Scripting injecte du JavaScript qui s'exécute dans le navigateur de la victime. Trois familles :

Type	Où le script est stocké	Exemple
Reflété	Nulle part (dans la réponse)	Paramètre d'URL renvoyé tel quel
Stocké	En base de données	Commentaire de forum persistant
DOM	Côté client uniquement	JavaScript qui écrit innerHTML

Test minimal sur une cible d'entraînement, dans un champ de recherche ou de commentaire :

```
<script>alert('XSS')</script>
<img src=x onerror=alert('XSS')>
```

Si une boîte d'alerte s'affiche, l'entrée n'est pas échappée. Le vrai danger n'est pas l'alerte, mais le vol de session ou le détournement d'action.

4.4 — Comment on se défend (le réflexe à garder)

- Requêtes préparées / paramétrées : la seule vraie parade à l'injection SQL.
- Échappement contextuel en sortie : la parade au XSS.
- Content-Security-Policy : réduit fortement l'impact d'un XSS.
- Principe du moindre privilège pour le compte base de données de l'application.

> Un bon testeur comprend toujours la défense. Savoir corriger une faille vaut plus que savoir l'exploiter.

4.5 — Le LAB : injecter et documenter

1. Lance Juice Shop ou DVWA et connecte-toi.
2. Reproduis l'injection SQL du point 4.2 sur la page de connexion.
3. Reproduis un XSS reflété dans le champ de recherche.
4. Note pour chaque faille : l'entrée utilisée, la réponse, et l'impact possible.
5. Écris la correction recommandée pour chacune.

Livrable : deux fiches « faille / preuve / impact / correction ». C'est exactement le format d'un rapport d'audit professionnel.

Module 5 — Linux : reconnaissance et escalade de privilèges

Une fois entré sur une machine, la question devient : « jusqu'où puis-je aller ? ». Ce module t'apprend la méthodologie d'escalade de privilèges, toujours sur une VM d'entraînement.

5.1 — Où regarder en arrivant

Sur une cible d'entraînement, une fois un accès obtenu, tu commences par l'inventaire. Ces commandes ne modifient rien : elles observent.

```
whoami && id                  # qui suis-je, quels groupes
uname -a                      # version du noyau
cat /etc/os-release           # distribution
sudo -l                       # que puis-je lancer en root ?
find / -perm -4000 -type f 2>/dev/null   # binaires SUID
crontab -l; ls -la /etc/cron*  # tâches planifiées
ss -tulpn                     # services en écoute
```

5.2 — Les pistes classiques d'escalade

Piste	Pourquoi c'est dangereux
Binaire SUID	S'exécute avec les droits du propriétaire (souvent root)
sudo mal configuré	Permet de lancer un outil en root sans mot de passe
Tâche cron modifiable	Un script exécuté par root que tu peux modifier
Mot de passe en clair	Fichier de config lisible contenant un secret
Noyau obsolète	Vulnérabilité locale connue et exploitable

5.3 — La méthodologie, pas le hasard

1. Énumère : qui je suis, quel système, quels droits.
2. Cherche les écarts : SUID, sudo, cron, capabilities, fichiers inscriptibles.
3. Croise avec une base de connaissances (GTFOBins, référence des SUID).
4. Teste UNE piste à la fois, sur ta VM, et note le résultat.
5. Documente ce qui a marché ET ce qui a échoué : les échecs sont instructifs.

> L'escalade n'est pas une liste de recettes magiques. C'est une méthode : énumérer, repérer l'écart, comprendre pourquoi c'est un écart, l'exploiter proprement, documenter.

5.4 — Le LAB : de l'utilisateur au root

Sur Metasploitable 2 (ou toute VM VulnHub prévue à cet effet) :

1. Connecte-toi avec un compte non privilégié fourni par la VM.
2. Lance l'inventaire complet du point 5.1 et note chaque sortie.
3. Identifie au moins un binaire SUID inhabituel.
4. Vérifie dans GTFOBins s'il est exploitable.
5. Obtiens un shell root, puis vérifie avec id.
6. Note la commande exacte qui a fonctionné : c'est ta preuve.

Livrable : la chaîne complète, de l'accès initial à root, écrite pas à pas.

Module 6 — Rédiger un rapport d'audit

Un test sans rapport n'existe pas. Le rapport est le livrable : c'est ce que le client lit, paie, et utilise pour corriger. Un bon technicien qui écrit mal perd contre un moyen technicien qui écrit bien.

6.1 — La structure standard

Section	Contenu
Résumé exécutif	2-3 paragraphes, langage non technique, pour la direction
Périmètre	Ce qui a été testé, quand, et les limites
Méthodologie	Approche suivie, outils utilisés
Constats	Chaque faille : preuve, impact, criticité, correction
Priorisation	Ce qu'il faut corriger d'abord, et pourquoi
Annexes	Captures, sorties de commandes, journaux

6.2 — Décrire un constat correctement

Chaque constat suit la même trame. C'est ce qui rend un rapport crédible.

- Titre : court et factuel (« Injection SQL sur la page de connexion »).
- Criticité : Critique / Élevée / Moyenne / Faible, avec justification.
- Preuve : la commande, la capture, la réponse observée.
- Impact : ce que l'attaquant peut faire concrètement.
- Correction : la mesure précise, pas « il faut sécuriser ».

6.3 — La règle d'or de la rédaction

Écris pour trois lecteurs en même temps : le dirigeant (qui veut l'impact), le développeur (qui veut la correction), et l'auditeur futur (qui veut la preuve). Le résumé exécutif parle au premier, les constats aux deux autres.

6.4 — Le LAB final : ton premier rapport

1. Reprends les livrables des modules 2 à 5.
2. Rédige un rapport de 5 à 8 pages avec la structure du point 6.1.
3. Écris un résumé exécutif de 200 mots maximum, sans jargon.
4. Classe les constats par criticité.

5. Relis en te demandant : « un non-technique comprend-il l'enjeu ? »

> Ce rapport est ton premier vrai portfolio. Garde-le : c'est exactement ce qu'un employeur ou un client veut voir.

Annexes

A. Aide-mémoire Nmap

```
nmap -sn <plage>            # découverte d'hôtes
nmap -sS <cible>            # scan SYN (défaut)
nmap -sT <cible>            # scan connect (sans root)
nmap -sU --top-ports 50     # scan UDP
nmap -sV <cible>            # versions de services
nmap -O <cible>             # détection d'OS
nmap -p- <cible>            # tous les ports
nmap -T4 <cible>            # plus rapide (agressif)
nmap --script default       # scripts par défaut
nmap -oA sortie <cible>     # exporter en 3 formats
```

B. Aide-mémoire Wireshark

```
ip.addr == <ip>                 # filtrer par IP
tcp.port == <port>              # filtrer par port
http.request                    # requêtes HTTP
dns                             # trafic DNS
tcp.analysis.flags              # anomalies TCP
tcp.analysis.retransmission     # retransmissions
tcp.flags.syn == 1              # paquets SYN
Analyse -> Expert Information   # vue des anomalies
```

C. Aide-mémoire Linux (énumération)

```
id                          # identité et groupes
sudo -l                     # droits sudo
find / -perm -4000 -type f  # binaires SUID
getcap -r / 2>/dev/null     # capabilities
crontab -l                  # tâches planifiées
ss -tulpn                   # services en écoute
uname -a                    # version du noyau
```

D. Plateformes d'entraînement légales

- TryHackMe — parcours guidés, idéal pour débuter.
- Hack The Box — machines à rooter, progression réelle.
- VulnHub — images de VMs vulnérables à télécharger.
- OWASP Juice Shop — application web moderne.

- DVWA — application web didactique.
- PortSwigger Web Security Academy — labs web gratuits et excellents.

E. Rappel légal (à ne jamais oublier)

Toutes les techniques de ce guide doivent être appliquées exclusivement à des systèmes que tu possèdes ou pour lesquels tu as une autorisation écrite. Le cadre professionnel s'appelle la « règle d'engagement » : périmètre, dates, cibles, actions autorisées, contacts. Sans ce document, il n'y a pas de test d'intrusion — il y a une infraction.

KadoHackLabs promeut exclusivement la sécurité défensive et les tests éthiques autorisés. Fais-en un usage responsable.

Et maintenant ?

Tu as un laboratoire, des outils, une méthodologie et un rapport. Ce qui fait la différence maintenant, c'est la répétition : une machine par semaine sur TryHackMe ou Hack The Box, et tu progresses plus vite qu'en lisant dix livres.

- Refais chaque lab en moins de temps, de mémoire.
- Prends une nouvelle machine chaque semaine.
- Note systématiquement ce qui a marché et ce qui a échoué.
- Partage tes notes : enseigner est la meilleure façon d'apprendre.

Bon lab. Reste curieux, reste éthique.

— KadoHackLabs

> Tu as fini ? Rejoins la communauté pour la suite du parcours et les nouveaux labs : t.me/KadoTech_IA_Cyber