FORMATION GRATUITE / NIVEAU DÉBUTANT

Hacking éthique
Les bases sur Linux

MANUEL PROFESSIONNEL / ÉDITION 2

Comprendre les mécanismes. Réaliser des ateliers bornés. Produire des preuves et des corrections expliquées.

12 modules	**60 questions de chapitre**
40 questions finales	**Laboratoire et projet**

Unité de formation Mashaah LLC, Delaware.
Support original en français, 6 octobre 2026.

Parcours sur viselareussite.com

Socle débutant préparant les niveaux intermédiaire et avancé. Les exercices et l’entreprise Atelier Aurore sont fictifs.

Sommaire

Lire, pratiquer, progresser 3

Syllabus et critères d’évaluation 5

Autorisation, déontologie et périmètre de test 6

Construire un laboratoire reproductible 10

Comprendre réseau, ports, DNS et échanges HTTP 15

Authentification, autorisation et intégrité des preuves 21

Lire une entrée non fiable et corriger son traitement 27

Inventorier, vérifier et prioriser une vulnérabilité 31

De la preuve à la correction et au contre-test 35

Linux : terminal, utilisateurs et permissions 39

Linux : services, sockets et frontière réseau 42

Linux : journaux, horodatage et analyse limitée 45

Linux : correctifs, confinement et restauration 48

Projet final : audit limité d’Atelier Aurore 51

Travaux pratiques et dossier final 54

Révision et préparation de la suite 58

Glossaire des bases 59

Références et provenance 63

Lire, pratiquer, progresser

Les neuf formations Windows, Linux et macOS se suivent en ligne : leçons, vidéos, exercices, QCM de chapitre et examen final. Le PDF est un support complémentaire facultatif. Les ateliers pratiques demandent un laboratoire local personnel. Les niveaux intermédiaire et avancé sont disponibles dans des parcours distincts.

Mode d’emploi du manuel

Ce manuel constitue le socle débutant du parcours gratuit VLR. Il associe les douze modules existants à des approfondissements, des exemples de commandes commentés et des exercices corrigés. Les durées du syllabus restent des repères, pas une garantie : lire les compléments, refaire un TP et construire un portfolio peut demander davantage de temps. La deuxième édition du manuel n’ajoute pas de QCM obligatoire et ne modifie pas votre progression déjà enregistrée.

Lisez les objectifs d’un module, étudiez le mécanisme, préparez l’environnement puis réalisez le TP. Notez votre prédiction avant la commande et votre résultat après. Consultez le corrigé après votre essai ; ne copiez pas une capture du manuel dans vos preuves personnelles. Passez ensuite le QCM du chapitre sur le site. La réussite demande quatre bonnes réponses sur cinq ; le QCM final s’ouvre après les douze chapitres validés et exige 32 réponses correctes sur 40.

Les exemples locaux utilisent des identités et données fictives. Les captures de l’application pédagogique sont réelles ; elles ne représentent pas des captures natives Windows ou Linux. Les commandes de ces deux systèmes ont été rapprochées de leurs documentations, sans exécution native revendiquée. Le diplôme de réussite interne VLR valide le parcours de QCM ; il n’atteste ni un audit professionnel supervisé ni un diplôme universitaire reconnu.

Diagnostic avant de commencer

Répondez sans outil : qu’est-ce qu’un fichier, un processus, un chemin relatif, une adresse IP et une permission ? Puis expliquez pourquoi l’adresse publique d’une entreprise ne vous autorise pas à la tester. Ce diagnostic ne compte pas pour la certification. Si vous ne pouvez pas expliquer un terme, reprenez le module laboratoire et le glossaire avant son premier atelier.

Exercice : le serveur est arrêté, mais le navigateur conserve une page ouverte. Peut-on conclure que le service fonctionne ? Corrigé : non ; le contenu visible peut avoir été chargé auparavant. Il faut une nouvelle requête vers health et son résultat, en notant l’heure. Cette question introduit la différence entre ce que vous voyez, ce que vous mesurez et ce que vous en déduisez.

Trois niveaux, une progression de compétences

Niveau	Finalité pédagogique	Travaux attendus
Bases, disponible	Comprendre les mécanismes et produire une preuve limitée	Laboratoire local, matrice d’accès, rapport et contre-tests
Intermédiaire, disponible en ligne	Planifier une évaluation plus structurée dans un laboratoire dédié	Application de test, réseau isolé, revue de code et remédiation
Avancé, disponible en ligne	Analyser des scénarios complexes et argumenter les limites	Projet spécialisé, validation croisée et restitution approfondie

Syllabus et critères d’évaluation

Volume indicatif : lectures, ateliers et projet personnel ; la durée des vidéos est distincte.

Prérequis

• Savoir installer et utiliser un logiciel depuis sa source officielle

• Pouvoir créer un dossier et ouvrir un terminal sur Linux

• Disposer de Python 3 et d’un navigateur récent

• Aucune expérience préalable de test d’intrusion requise

Compétences visées

• Définir un mandat et ses conditions d’arrêt

• Construire un laboratoire local reproductible

• Comprendre les échanges réseau et les contrôles d’accès

• Observer fichiers, processus et journaux sur Linux

• Confirmer un défaut pédagogique et écarter un faux positif

• Proposer une correction et un contre-test

• Rédiger un rapport reproductible avec des preuves proportionnées

Module	Compétence / sujet	Budget
01	Autorisation, déontologie et périmètre de test	4 h
02	Construire un laboratoire reproductible	4 h
03	Comprendre réseau, ports, DNS et échanges HTTP	4 h
04	Authentification, autorisation et intégrité des preuves	4 h
05	Lire une entrée non fiable et corriger son traitement	4 h
06	Inventorier, vérifier et prioriser une vulnérabilité	4 h
07	De la preuve à la correction et au contre-test	4 h
08	Linux : terminal, utilisateurs et permissions	4 h
09	Linux : services, sockets et frontière réseau	4 h
10	Linux : journaux, horodatage et analyse limitée	4 h
11	Linux : correctifs, confinement et restauration	4 h
12	Projet final : audit limité d’Atelier Aurore	16 h

Les 60 heures sont un budget indicatif : modules 1 à 11, chacun 1 h de cours, 2 h de TP et 1 h de révision ; projet 12 h et préparation finale 4 h. Les compléments et les reprises peuvent augmenter le temps réel. La durée des vidéos est distincte.

Quatre réponses correctes sur cinq sont requises à chacun des douze QCM. Le QCM final demande 32/40. Le projet est autoévalué. Aucun jury, crédit ECTS, contrôle d’identité supervisé ni reconnaissance universitaire du diplôme interne VLR n’est revendiqué.

MODULE 01 / BASES LINUX

Autorisation, déontologie et périmètre de test

À l’issue du module : Délimiter une mission ; Distinguer accès et autorisation ; Écrire une condition d’arrêt

Un test de sécurité sert à vérifier une hypothèse sur un système et à proposer une correction. La permission est une donnée technique du travail : elle désigne le propriétaire habilité, les actifs concernés, les horaires, les méthodes admises et les situations où il faut s’arrêter. Posséder un compte sur un service ne donne pas le droit de tester tous ses utilisateurs. Une adresse accessible sur Internet ne constitue pas une autorisation. Dans ce parcours, les essais portent sur un laboratoire local construit pour l’apprentissage et sur des données fictives.

Avant de lancer un outil, définissez la question. « Une utilisatrice peut-elle consulter le document fictif d’une autre ? » est une question testable. « Trouver tout ce qui ne va pas » ne fixe ni l’étendue ni le critère de réussite. Associez chaque question à une preuve minimale. Lire la réponse d’un document de démonstration suffit pour établir un défaut de contrôle d’accès ; collecter une base entière n’améliore pas cette preuve. Cette sobriété facilite l’analyse et limite les conséquences d’une erreur.

La disponibilité fait partie de la sécurité. Même un outil connu peut produire trop de requêtes, remplir un journal ou perturber une application fragile. Le laboratoire utilise des essais manuels bornés et un service limité à l’adresse de boucle locale. L’arrêt d’un test n’est pas un échec lorsque sa condition d’arrêt est atteinte. Notez l’événement, conservez uniquement les éléments nécessaires et demandez une clarification avant toute extension du périmètre.

Séparez une observation d’une conclusion. Une page de connexion existe : c’est un fait. Le système est bien protégé : cette conclusion exige davantage de tests. Un code 403 n’établit pas, à lui seul, que toutes les routes contrôlent leurs permissions. À l’inverse, une réponse inhabituelle ne prouve pas une compromission. Le rapport doit rendre visible ce que vous avez vérifié, ce que vous supposez et ce que vous n’avez pas étudié.

Le projet utilise une entreprise fictive, Atelier Aurore, et deux identités de laboratoire, Alice et Bob. Aucun nom de client, mot de passe réel, adresse privée ou donnée médicale n’est nécessaire. Les règles juridiques et contractuelles varient selon le pays : ce parcours enseigne une méthode d’autorisation et ne remplace pas l’examen du mandat applicable. Vous ne testez pas un employeur, une école, un voisin ou une plateforme externe pour réaliser un devoir.

Définir les objectifs de sécurité

La confidentialité limite la divulgation, l’intégrité protège la cohérence et la traçabilité des informations, et la disponibilité permet un usage attendu dans les conditions prévues. Un défaut peut toucher plusieurs objectifs. Dans le laboratoire, la réponse du document de Bob à Alice affecte la confidentialité. Une modification non autorisée aurait aussi un effet sur l’intégrité. Un serveur arrêté ne démontre pas un contrôle d’accès fiable : il rend le service indisponible. Pour chaque question de test, nommez l’objectif concerné et le résultat attendu avant de choisir un outil.

Menace, vulnérabilité, impact et risque

Une menace décrit une possibilité d'événement défavorable ; une vulnérabilité décrit une faiblesse exploitable dans certaines conditions ; l'impact décrit ses conséquences. Le risque dépend notamment de l'exposition et du contexte. Un audit n'a pas besoin de simuler toutes les menaces pour produire une information utile. Il doit annoncer sa couverture et les critères de décision. Dans le projet, les actifs sont fictifs et la conséquence est limitée au document de démonstration. Vous apprenez à raisonner sans présenter ce cas comme une fuite réelle.

Conduire une mission, même dans un laboratoire

Un mandat utilisable distingue le propriétaire, la personne qui autorise, l'exécutant et le destinataire du rapport. Une adresse techniquement accessible n'est pas une autorisation. Pour Atelier Aurore, le propriétaire fictif autorise seulement le service local 127.0.0.1:8876 et les fichiers distribués. Le périmètre ne comprend ni les comptes personnels de l'apprenant, ni son réseau familial, ni le site public de Vise la Réussite. La formation vous donne un terrain de travail ; elle ne vous autorise pas à tester une organisation réelle. Une demande ultérieure se traite comme une nouvelle mission.

Écrivez les méthodes permises avant de choisir un logiciel : demandes HTTP prévues, lecture de fichiers du TP, comparaison d'empreintes et traitement du journal fictif. Ajoutez une limite de fréquence et une règle d'arrêt si une requête quitte la boucle locale, si un résultat contient des données réelles ou si un service inconnu est touché. Le contrat d'un audit réel dépend de la mission et des juridictions concernées ; notre formulaire pédagogique ne remplace pas ce contrat. Le débutant apprend d'abord à reconnaître cette limite.

Une menace décrit ce qui pourrait se produire ; un risque relie une situation, un impact et des conditions. Dans notre exemple, la lecture par Alice du document de Bob met en jeu la confidentialité. L'arrêt complet du laboratoire relève de la disponibilité. Une modification incorrecte d'un document relève de l'intégrité. Ces termes servent à préciser le besoin, sans transformer chaque erreur en incident grave. Un actif fictif sans conséquence réelle permet d'apprendre le raisonnement, pas de mesurer le risque financier d'une entreprise.

Avant le premier essai, créez un registre : objectif, hypothèse, route, identité fictive, résultat attendu, résultat observé et décision. La même matrice vous aidera à relire le rapport. Une opération qui n'aide pas à répondre à une hypothèse précise est probablement prématurée. Au terme du TP, vous devez pouvoir expliquer pourquoi chaque requête a été nécessaire, où sa preuve est conservée et ce qui reste inconnu. Cette discipline différencie une mission reproductible d'une succession de commandes.

Du besoin métier à une hypothèse vérifiable

Le hacking éthique mobilise des connaissances techniques pour évaluer un système dans un périmètre autorisé. Son résultat est une décision étayée : quelle faiblesse est confirmée, dans quelles conditions, avec quelle correction ? Il se distingue d'une simple utilisation d'outils. Un inventaire n'est pas encore un audit, une alerte n'est pas encore une vulnérabilité confirmée, et une vulnérabilité ne prouve pas qu'un incident a eu lieu.

Dans le cas fictif Atelier Aurore, les documents 101 et 102 appartiennent respectivement à Alice et Bob. Le besoin métier est que chaque personne voie son document sans accéder à celui de l'autre. L'hypothèse est « la route corrigée refuse Alice lorsqu'elle demande 102 ». Le critère est une réponse 403 sans contenu du document. Le contre-test vérifie qu'Alice peut encore obtenir 101. Une mesure qui bloque tous les documents échoue sur le fonctionnement attendu, même si elle empêche aussi l'accès indu.

Séparez trois rôles : le commanditaire fixe le besoin, l'auditeur collecte des observations et le propriétaire du service décide des corrections. Dans une vraie organisation, ces rôles peuvent être répartis entre plusieurs équipes. Le débutant doit savoir à qui adresser une question de périmètre et qui autorise une modification. Il ne

reçoit pas un droit de changement implicite parce qu'il a identifié une faiblesse.

Règles d'engagement utilisables

Avant un atelier, écrivez les actifs inclus, les exclusions, les identités disponibles, les actions autorisées, la fenêtre de travail, le destinataire des preuves et les conditions d'arrêt. Définissez aussi la procédure de retour à l'état initial. Une permission d'observation ne couvre pas nécessairement une modification, une collecte massive ou un test de charge. Dans ce manuel, aucune activité ne nécessite ces deux dernières méthodes.

Exercice de décision : une capture contient une adresse personnelle en plus de la preuve utile. Conservez une copie originale dans votre dossier privé si sa conservation est autorisée, puis produisez une copie expurgée pour le rapport. N'effacez pas silencieusement une partie d'une preuve : mentionnez le masquage, sa raison et les informations retirées. Si les données n'auraient pas dû être présentes, arrêtez l'essai et demandez la conduite à tenir.

Corrigé : on réduit la diffusion et on documente la transformation. Un rapport partageable n'est pas l'archive brute. La qualité d'un audit dépend aussi de la manière dont ses preuves sont protégées. Le niveau intermédiaire approfondira la préparation d'une mission multi-actifs ; les règles de périmètre restent nécessaires à tous les niveaux.

Méthode pas à pas

Préparez une fiche comprenant objectif, propriétaire du laboratoire, adresse 127.0.0.1, port 8876, données fictives autorisées, méthodes admises, heure de début, limite de requêtes et condition d'arrêt. Vérifiez le périmètre avant chaque TP. Utilisez la fiche de mandat du dossier laboratoire. Si une destination redirige vers un service externe, ne suivez pas la redirection pour le test. Terminez par un journal daté des opérations réellement effectuées.

Votre exercice

Complétez le mandat d'Atelier Aurore. Classez quatre demandes : tester le service local, tester une adresse trouvée sur un moteur de recherche, modifier un fichier de TP et lancer un test sur le poste d'un collègue. Expliquez votre décision et les informations manquantes.

Cas entièrement fictif

Cas fictif : un responsable dit « vérifie notre sécurité » et transmet une URL sans document de périmètre. L'élève doit reformuler la mission et commencer uniquement par le laboratoire pédagogique local déjà autorisé.

Corrigé raisonné

Le laboratoire local est admis avec son périmètre défini. L'adresse découverte publiquement et le poste du collègue restent hors périmètre. Le fichier de TP appartient au bac à sable du cours et peut être modifié selon la consigne. Une URL donnée oralement ne couvre pas les prestataires, les infrastructures partagées ou les données des utilisateurs. La bonne réponse est une demande de précisions, pas une découverte intrusive. Le mandat doit également indiquer comment transmettre les constats sans rendre publiques les preuves.

MODULE 01 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Quel élément fixe les cibles autorisées ?

A. Le mandat explicite

B. La seule accessibilité technique

C. Le nom du logiciel utilisé

2. Quelle situation impose un arrêt du TP ?

A. Un refus 403 attendu

B. Une empreinte de fichier différente après votre modification

C. La découverte de données réelles étrangères au TP

3. La confidentialité concerne principalement…

A. La taille du fichier seul

B. Les accès aux informations

C. Le temps de chargement seul

4. Quel document rend une hypothèse vérifiable ?

A. Une matrice attendu/observé

B. Une liste de logos

C. Un nombre de commandes sans contexte

5. Quelle cible appartient au TP fourni ?

A. Le site public VLR

B. Tout équipement sur le Wi-Fi

C. Le service local 127.0.0.1:8876

MODULE 02 / BASES LINUX

Construire un laboratoire reproductible

À l’issue du module : Documenter un environnement ; Vérifier une isolation ; Restaurer un état connu

Un laboratoire doit rendre une erreur réversible. Créez un dossier consacré au cours, puis un sous-dossier par TP. Les programmes fournis n’ouvrent aucun accès vers une cible distante : le serveur écoute sur 127.0.0.1 et sert uniquement ses propres routes pédagogiques. Ne remplacez pas cette adresse par 0.0.0.0. Le serveur de démonstration n’est pas destiné à être publié, installé sur un serveur d’entreprise ou conservé actif après l’exercice.

Un compte utilisateur ordinaire suffit aux exercices Python. Les ateliers d’administration propres à chaque OS se font dans une machine de laboratoire dont vous contrôlez les réglages. Une machine virtuelle peut faciliter les essais de permissions et les restaurations. Sur un Mac à processeur Apple, une image Linux ARM64 n’est pas équivalente à une image x86_64 ; les images Windows doivent correspondre à l’architecture et aux droits de licence disponibles. L’installation d’une VM n’est donc pas une condition silencieuse : on vérifie sa faisabilité avant de promettre un parcours de commandes identique.

Distinguez isolation réseau et simple commodité. Le réseau NAT d’une VM permet généralement des sorties ; il ne garantit pas l’absence de communication extérieure. Pour un exercice local, la boucle locale suffit. Pour une future machine cible séparée, un réseau interne sans accès sortant et des règles explicites seront nécessaires. Le niveau débutant ne demande aucune cible exploitable sur Internet, aucun transfert de port et aucune désactivation de protection du poste principal.

Fixez une version du support, de l’interpréteur et de l’application. Un relevé de version aide à comprendre pourquoi un résultat diffère. Il ne faut pas recopier une sortie attendue comme si elle avait été observée. Dans les fiches de résultat, utilisez les mentions « exécuté », « non exécuté » ou « non applicable ». Les captures doivent venir de votre essai et afficher des données fictives. Une capture de documentation est utile, mais ne prouve pas que votre système a été testé.

Le dossier de preuves conserve le mandat, la liste des outils, le résultat des tests et les fichiers avant/après correction. Une empreinte SHA-256 peut montrer qu’un fichier a changé entre deux observations. Elle ne garantit ni l’origine du fichier ni son innocuité. Conservez la trace du script utilisé pour obtenir l’empreinte et l’heure de collecte. Fermez le service par Ctrl+C en fin d’atelier et vérifiez que la page ne répond plus.

Les bases de Python nécessaires au TP

Un script contient des instructions, des fonctions et des valeurs. Une fonction reçoit des paramètres et produit un résultat ; une liste conserve plusieurs éléments ; un dictionnaire associe des clés à des valeurs. Dans tp_sql.py, demo crée la base en mémoire, fragile construit une commande à partir d’une chaîne et corrected fournit le paramètre séparément. Une exception signale un problème à traiter ; elle ne prouve pas une faille. Lisez d’abord la fonction avant de la modifier. Travaillez sur une copie, changez une seule hypothèse, relancez les mêmes cas et comparez les résultats. Vous n’avez pas besoin d’écrire un outil offensif pour comprendre la différence entre donnée et instruction.

Une machine attaquante n’est pas une identité de métier

Kali ou une autre distribution spécialisée peut regrouper des outils. Son installation ne donne aucune permission et n’est pas un prérequis à ces fondamentaux. Le cours Linux utilise un environnement généraliste de TP. Le parcours Windows n’est pas une simulation Linux dans une fenêtre, et le parcours macOS conserve ses outils et protections propres. L’objectif est de comprendre les décisions et les preuves sur chaque OS. Les outils plus spécialisés seront introduits avec un besoin précis au niveau intermédiaire.

Télécharger les fichiers de travaux pratiques

Téléchargez et décompressez l’archive du laboratoire. Elle contient les scripts Python, le guide, les données fictives, le modèle de mandat et le modèle de rapport. Lisez GUIDE-TP.md avant de lancer laboratoire.py. Le serveur écoute exclusivement sur 127.0.0.1:8876.

Préparation reproductible et résolution des erreurs

Python 3 exécute le laboratoire avec sa bibliothèque standard. Vérifiez sa version dans votre terminal avant de commencer. Le nom de la commande dépend de l’installation : py -3 ou python sous Windows, python3 sous Linux et macOS. Un message indiquant que la commande est inconnue signale une installation ou un chemin à vérifier, pas une raison de désactiver la protection de l’OS. Utilisez la documentation officielle de Python et consignez la version réellement installée. Les scripts n’exigent ni Kali Linux ni un abonnement commercial.

Décompressez l’archive dans un dossier utilisateur consacré à la formation. Gardez les noms des fichiers et leur encodage UTF-8. Le fichier atelier.html doit rester à côté de laboratoire.py. Si l’interface manque mais que health répond, comparez d’abord le contenu de l’archive et le chemin affiché dans le terminal. Si le navigateur annonce un refus de connexion, vérifiez que le serveur tourne, que l’URL contient le bon port et que l’adresse reste 127.0.0.1. Changez une seule hypothèse à la fois et notez le résultat.

Le message « adresse déjà utilisée » signifie qu’un autre processus écoute déjà sur ce port. Il ne désigne pas nécessairement une attaque. Identifiez le processus avec le module propre à votre OS et comparez son rôle. N’arrêtez pas un programme inconnu pour libérer le port. Fermez votre ancien serveur si vous avez conservé une fenêtre de TP ouverte. Une autre option consiste à reporter le TP et à documenter l’obstacle ; le cours ne demande pas de modifier les services de la machine principale.

Une machine virtuelle permet ensuite de séparer des essais d’administration de votre poste principal. Elle consomme des ressources et ne dispense pas d’un périmètre : un mode réseau bridgé peut exposer davantage qu’une boucle locale. Pour cette base, le service local suffit. Si vous choisissez une VM, documentez l’hôte, l’invité, le type de réseau et le point de restauration, puis utilisez des comptes fictifs. L’interconnexion de plusieurs VM et les scénarios d’attaque complets seront traités au niveau intermédiaire.

Ce qu’il faut comprendre avant le premier terminal

Le système d’exploitation organise les processus, la mémoire, les fichiers et les périphériques. Un processus est une instance d’un programme en cours d’exécution ; un fichier de programme n’est pas ce processus. Un service peut démarrer sans fenêtre visible. Un terminal est une interface ; le shell interprète ce que vous y saisissez. PowerShell manipule principalement des objets, tandis qu’un pipeline de shell Unix transmet généralement du texte ou des octets.

Un chemin relatif dépend du dossier courant. Un chemin absolu désigne un emplacement depuis la racine du système ou du volume. Les espaces dans un nom exigent une syntaxe adaptée ; les guillemets ne transforment pas une chaîne en commande sûre dans tous les contextes. N’exécutez pas une ligne copiée sans identifier son

interpréteur, ses arguments, les fichiers qu'elle modifie et sa destination réseau.

Utilisez un compte ordinaire pour le laboratoire Python. Le préfixe sudo ou une console élevée change les conséquences d'une erreur. Un message « accès refusé » mérite d'abord une analyse de la ressource et des permissions, plutôt qu'une élévation automatique. Les versions, éditions et distributions peuvent modifier la disponibilité d'une commande ; notez le message réel si une étape est impossible.

Préparer une structure de preuves

Créez un dossier indépendant de vos documents personnels. Organisez-le en mandat, notes, preuves originales, copies expurgées et rapport. N'utilisez aucune donnée de votre employeur. Conservez la date, le fuseau, la version Python et la version du manuel dans la fiche de TP. Les exercices fournis sont des programmes lisibles : ouvrez leurs sources avant de les lancer.

Un instantané de machine virtuelle facilite le retour à un état, mais ne constitue pas à lui seul une sauvegarde indépendante. Le mode NAT ne rend pas une VM entièrement isolée ; il autorise souvent des communications sortantes. Pour ce parcours, le service reste sur la boucle locale. Une VM est utile pour les futurs essais d'administration ; elle n'est pas nécessaire aux exercices Python distribués.

Python : comprendre une petite automatisation

Une variable associe un nom à une valeur. Une liste conserve une suite de valeurs ; un dictionnaire associe des clés à des valeurs. Une fonction permet de nommer une opération, de recevoir des paramètres et de rendre un résultat. Un bloc try/except traite une erreur prévue ; masquer toute exception rend l'analyse moins fiable. JSON transporte des données structurées, pas un programme à exécuter.

```
import json
from pathlib import Path

fiche = {"tp": "PREUVE-01", "donnees": "fictives"}
destination = Path("fiche-exemple.json")
destination.write_text(
    json.dumps(fiche, ensure_ascii=False, indent=2),
    encoding="utf-8"
)
relue = json.loads(destination.read_text(encoding="utf-8"))
assert relue["donnees"] == "fictives"
print(relue["tp"])
```

Cet exemple crée ou remplace seulement fiche-exemple.json dans le dossier courant. Exécutez-le dans un dossier de TP vide. Résultat attendu : PREUVE-01 et un petit fichier JSON. Le résultat dépend du droit d'écriture ; un échec ne prouve pas que Python est mal installé. En cas d'erreur, relevez le chemin courant et le message sans publier votre arborescence personnelle.

Méthode pas à pas

Vérifiez Python 3 avec python --version, python3 --version ou py -3 --version selon votre installation. Copiez le dossier du laboratoire dans votre espace de TP. Démarrez laboratoire.py sans droits administrateur ; il annonce l'adresse locale. Ouvrez uniquement http://127.0.0.1:8876. Créez votre fiche d'environnement, réalisez un test, arrêtez le service et comparez le résultat. Si le port est occupé, identifiez d'abord votre ancien processus ; n'arrêtez pas un service inconnu.

Votre exercice

Produisez une fiche d’environnement avec OS, architecture, Python, version du cours, port, dossier et procédure de retour à l’état initial. Démontrez que l’arrêt du serveur rend la page indisponible.

Cas entièrement fictif

Cas fictif : deux apprenants obtiennent des commandes de lancement différentes, l’un avec py -3 et l’autre avec python3. Tous deux doivent prouver qu’ils exécutent la même version du laboratoire, sans conclure que l’autre a échoué.

Corrigé raisonné

La fiche explique la différence de lanceur et indique la version observée. Le serveur est lié à la boucle locale et aucun port du routeur n’est ouvert. La capture après arrêt n’est pas remplacée par une capture du premier essai. Une VM est proposée pour les changements d’administration, mais les TP applicatifs restent réalisables sans modifier les protections du poste. Le dossier de preuves ne contient aucun secret personnel.

MODULE 02 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Pourquoi le serveur écoute-t-il sur 127.0.0.1 ?

A. Pour limiter l’écoute à la boucle locale

B. Pour remplacer une authentification

C. Pour rendre le service public

2. Le port est déjà occupé. Quel premier geste convient ?

A. Arrêter tous les processus Python

B. Désactiver le pare-feu

C. Identifier le processus avant toute action

3. Quel logiciel suffit au laboratoire distribué ?

A. Un serveur de production

B. Python 3 et sa bibliothèque standard

C. Un abonnement à un scanner commercial

4. Health répond mais la page manque. Que vérifier ?

A. La présence et le chemin de atelier.html

B. Le chiffrement complet du disque

C. Les comptes du site public

5. Quel relevé facilite la reproduction ?

A. Le seul prénom de l’élève

B. Le nom de son employeur réel

C. OS, version Python, version du cours et commande

MODULE 03 / BASES LINUX

Comprendre réseau, ports, DNS et échanges HTTP

À l’issue du module : Lire une requête HTTP ; Relier adresse, port et processus ; Écarter un diagnostic réseau prématuré

Une adresse IP identifie une interface dans un contexte réseau ; un port identifie le point d’écoute d’un service. TCP fournit un transport connecté, tandis qu’HTTP décrit la demande et la réponse d’une application. Le port 8876 de notre laboratoire a été choisi pour l’exercice. Un numéro de port ne prouve pas quel logiciel y répond : il faut observer le service et son propriétaire. Un port fermé n’est pas une preuve d’absence de toute vulnérabilité.

127.0.0.1 désigne la boucle locale IPv4. Le nom localhost peut aussi résoudre vers la boucle locale IPv6. Si un client essaie ::1 alors que le service écoute seulement en IPv4, il peut obtenir un échec qui ressemble à un arrêt du serveur. Le TP utilise explicitement 127.0.0.1 pour réduire cette ambiguïté. DNS traduit un nom en informations d’adressage ; il n’accorde pas de droit d’accès et ne garantit pas la confiance accordée au destinataire.

Une requête HTTP comporte une méthode, un chemin, des en-têtes et parfois un corps. Une réponse comporte un statut, des en-têtes et un contenu. GET demande une représentation ; POST peut soumettre une opération. Les statuts 200, 401, 403 et 404 expriment des résultats différents, mais leur signification dépend aussi de l’implémentation. Une application peut choisir de répondre 404 pour ne pas révéler l’existence d’un objet. On examine donc le comportement attendu plutôt que de noter seulement le chiffre.

HTTPS ajoute TLS au transport. Il protège la communication quand la validation du certificat et du nom est correcte. Il ne corrige pas une erreur de contrôle d’accès côté serveur. Le laboratoire HTTP est volontairement limité à la machine locale et utilise des données fictives ; on ne le présente pas comme un modèle de déploiement public. Ne désactivez pas la validation TLS pour faire fonctionner une requête réelle dans un devoir.

La capture réseau exige de maîtriser ce qui est enregistré. Au niveau débutant, les outils de développement du navigateur suffisent pour examiner les requêtes du laboratoire. Ils évitent de collecter le trafic du Wi-Fi ou d’autres applications. L’onglet Réseau montre le chemin, le statut et les en-têtes ; il peut aussi afficher un jeton de session. Même fictif, ce jeton est masqué dans le rapport si sa valeur n’apporte rien à la preuve. Une nouvelle requête doit rester dans le mandat local.

Découverte ciblée avec Nmap, atelier facultatif

Si Nmap est installé depuis sa source officielle dans votre environnement de TP, la commande nmap -sT -Pn -p8876 127.0.0.1 vérifie le seul port prévu du laboratoire. -sT utilise une connexion TCP ; -Pn évite de dépendre d’une découverte préalable d’hôte ; -p fixe le port. N’élargissez pas la cible ou les ports pour ce devoir. Comparez le résultat pendant et après l’arrêt du serveur. Un état open décrit une écoute au moment du test ; il ne valide pas l’application. Le client Python fourni réalise déjà la vérification fonctionnelle. Cet atelier est facultatif et sa non-exécution doit être indiquée si l’outil n’est pas disponible.

Observer avec Wireshark ou un proxy passif

Le navigateur reste l'outil obligatoire. Dans une VM ou un poste dédié au TP, Wireshark peut illustrer l'échange de boucle locale avec un filtre limité à TCP port 8876. Le nom et les permissions de l'interface diffèrent selon l'OS. La capture ne doit pas englober le Wi-Fi ni les communications d'autres applications. ZAP peut aussi observer passivement les seules requêtes du laboratoire dans un profil de navigateur distinct. N'activez aucun scan automatique sur un domaine tiers. Une alerte passive est un indice à examiner, pas une preuve complète. Un outil non disponible ne doit pas être contourné en réduisant les protections du poste.

Outils de réseau : observer avant d'interpréter

Nmap constitue un outil facultatif de découverte. Dans ce TP, une commande limitée est nmap -sT -p 8876 127.0.0.1 : elle demande une connexion TCP sur le seul port du laboratoire. Le résultat open indique qu'une connexion a été acceptée au moment de l'essai ; il ne prouve ni la version du logiciel, ni une vulnérabilité. Exécutez-la seulement si vous avez installé Nmap depuis sa source officielle et consignez sa version. Le TP ne demande ni balayage de sous-réseau, ni script de recherche de failles, ni contournement du pare-feu. Comparez le résultat avec health et avec la lecture du processus.

Wireshark aide à observer des paquets, tandis que les outils de développement du navigateur décrivent les échanges applicatifs. Une capture de boucle locale demande une interface adaptée et, selon l'installation, des droits ou composants spécifiques. Préférez l'onglet Réseau pour ce premier exercice. Si vous disposez déjà d'un environnement de capture de TP, sélectionnez uniquement la boucle locale et filtrez tcp.port == 8876. Arrêtez la capture après quelques requêtes. Le filtre d'affichage réduit ce que vous voyez ; il ne garantit pas que le fichier de capture ne contient pas d'autres paquets.

Une adresse peut être privée sans appartenir à votre laboratoire. Une plage 192.168.x.x peut contenir une imprimante, un téléphone ou une machine professionnelle. Ne la confondez pas avec 127.0.0.1. IPv6 dispose aussi d'une boucle locale, ::1, mais notre serveur de référence écoute en IPv4. Une absence de réponse sur un autre protocole ou une autre interface doit être décrite avec précision. DNS, routage, TCP, TLS et HTTP sont des couches différentes : une panne de résolution n'est pas une panne de permission applicative.

Dans la preuve, relevez méthode, chemin, statut, type de contenu et identité fictive. Ne publiez pas un export complet de votre navigateur personnel. Vous pouvez joindre une capture annotée limitée à la requête de TP et au contenu utile. Si vous mesurez un temps, indiquez le nombre d'essais, la machine et ce qui est mesuré. Une réponse locale rapide ne prédit pas les performances d'un site hébergé. Votre conclusion doit rester proportionnée à l'observation.

Adresses, préfixes et frontières

IPv4 représente une adresse sur 32 bits, souvent écrite en quatre nombres décimaux ; IPv6 utilise 128 bits. Le préfixe précise combien de bits définissent le réseau. Dans l'exemple documentaire 192.0.2.0/24, il reste huit bits pour les adresses, soit 256 combinaisons. La plage utilisable dépend du type de réseau : sur un sous-réseau IPv4 classique /24, l'adresse de réseau et l'adresse de diffusion ne sont généralement pas attribuées aux hôtes. N'appliquez pas cette règle à tous les préfixes ni à IPv6.

Une adresse privée n'est pas une autorisation de test. Elle peut désigner un autre poste, une infrastructure d'entreprise ou un équipement familial. 127.0.0.1 est l'adresse IPv4 de boucle locale utilisée ici ; ::1 est son équivalent IPv6, mais le serveur fourni écoute en IPv4. Un refus sur ::1 ne démontre donc pas un dysfonctionnement du laboratoire. localhost peut se résoudre différemment selon l'environnement : employez l'adresse explicite du TP.

Un port permet de distinguer des points de communication au sein d'une adresse. Une écoute 127.0.0.1:8876 concerne la machine locale. Une écoute 0.0.0.0:8876 couvre les interfaces IPv4 disponibles ; ce n'est pas la configuration autorisée ici. L'accessibilité effective dépend aussi des règles réseau et du pare-feu. L'existence d'une écoute ne suffit pas à conclure à une exposition Internet.

Calcul de sous-réseau sans connexion

Le module Python ipaddress réalise des calculs locaux. Il ne contacte aucun hôte. Les adresses ci-dessous servent à la documentation : aucun balayage de réseau n'est demandé.

```
import ipaddress

reseau = ipaddress.ip_network("192.0.2.0/28")
print(reseau.num_addresses)
print(reseau.network_address)
print(reseau.broadcast_address)
print(ipaddress.ip_address("192.0.2.10") in reseau)
print(ipaddress.ip_address("192.0.2.20") in reseau)
```

Résultats attendus : 16, 192.0.2.0, 192.0.2.15, True, False. Un /28 conserve quatre bits variables : 2 puissance 4 donne 16 adresses. L'adresse .10 appartient à cette plage et .20 n'y appartient pas. Exercice : remplacez /28 par /29 et prédisez les résultats avant d'exécuter. Corrigé : huit adresses, dernière adresse .7 ; .10 et .20 sont hors de cette plage.

TCP, UDP, DNS et TLS : des rôles différents

TCP fournit un flux ordonné avec des mécanismes de retransmission. UDP transmet des datagrammes sans offrir ces mêmes garanties au niveau du protocole. Leur comportement ne se résume pas à « sécurisé » et « dangereux ». La sécurité dépend de l'application, de l'authentification, des protections et du contexte. Certaines versions de HTTP utilisent des transports différents : ne déduisez pas la pile complète du seul nom du protocole.

DNS associe notamment des noms à des informations telles que des adresses. Une résolution correcte ne garantit pas l'intégrité de l'application. TLS protège la communication et participe à l'authentification du serveur lorsque le certificat et son nom sont correctement vérifiés. Il ne corrige ni une permission applicative absente ni une requête SQL fragile. Le laboratoire HTTP sans TLS reste local et ne contient aucun secret réel ; ce choix n'est pas un modèle de production.

Lire une conversation HTTP

Une requête présente une méthode, une cible, des en-têtes et parfois un corps. Une réponse comporte un statut, des en-têtes et éventuellement un corps. GET vise la récupération d'une représentation ; POST est utilisé pour transmettre une donnée à traiter selon l'application. Les cookies servent notamment à transporter un identifiant de session. Un en-tête envoyé par le client est une entrée non fiable tant que le serveur n'a pas établi sa validité.

Observation	Interprétation prudente	Vérification suivante
200 sur health	Cette route répond à cet instant	Examiner séparément la permission
401 sur un document	L'identité requise n'est pas reconnue	Tester une identité autorisée
403 pour Alice vers 102	Ce cas d'accès est refusé	Tester Alice vers 101

Observation	Interprétation prudente	Vérification suivante
404 sur une route	La ressource n’est pas présentée	Vérifier l’URL et le périmètre
500	Le service signale une erreur interne	Conserver l’heure et lire le journal local

Les applications peuvent volontairement choisir un 404 pour ne pas révéler une ressource ; un statut seul ne décrit pas toute leur politique. Dans les outils de développement du navigateur, relevez aussi le contenu renvoyé. Ne copiez jamais de cookie réel dans le rapport. Une capture filtrée sur le laboratoire est suffisante.

Atelier de diagnostic progressif

Si la page ne s’ouvre pas, vérifiez d’abord que le serveur a démarré dans la fenêtre attendue, puis l’adresse et le port. Examinez seulement l’écoute du port 8876. Si la page s’ouvre mais pas health, relevez la requête et le statut. Si une identité échoue, comparez la matrice d’autorisation. Ne désactivez pas le pare-feu global pour « essayer » : cette opération ne localise pas la cause.

Livrable : cinq observations distinctes, leur hypothèse et une étape suivante. Corrigé : l’absence d’écoute après Ctrl+C explique le refus de connexion ; un 403 applicatif signifie que la connexion et la requête ont atteint le serveur. Ces deux observations se situent à des couches différentes. Cette distinction prépare l’analyse réseau du niveau intermédiaire.

Schéma de lecture du laboratoire



Schéma original VLR : trajet de la requête locale ; la réponse revient au client.

Méthode pas à pas

Ouvrez les outils de développement du navigateur et leur onglet Réseau. Chargez le laboratoire, puis la route /api/health. Relevez URL, méthode, statut, Content-Type et temps de réponse. Comparez la page HTML et la réponse JSON. N’activez pas un enregistrement de trafic général. Avec le client fourni client_tp.py, confirmez la même réponse sur le seul port 8876 de 127.0.0.1. Aucun balayage de plage d’adresses n’est demandé.

Votre exercice

Dessinez le trajet navigateur → adresse locale → port → application → réponse. Annotez une réponse 200 et un refus 403 du laboratoire. Expliquez ce que chaque observation établit et ce qu’elle ne prouve pas.

Cas entièrement fictif

Cas fictif : Alice voit une erreur de connexion après avoir saisi localhost au lieu de l’adresse indiquée. Elle conclut à tort que le pare-feu bloque tous les accès.

Corrigé raisonné

Le diagnostic commence par l’adresse réellement utilisée et l’état du processus de laboratoire. Un échec IPv6 peut expliquer la différence sans changer le pare-feu. La route health répond en JSON et sa disponibilité ne prouve pas que les documents sont bien protégés. HTTP est acceptable ici uniquement pour ce service de boucle locale fictif. Le rapport distingue le temps de réponse d’un service déjà démarré du premier lancement et ne généralise pas une mesure unique.

MODULE 03 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Un port TCP ouvert prouve…

A. Qu’une connexion a été acceptée au moment du test

B. Une vulnérabilité exploitable certaine

C. La version exacte de l’application

2. HTTPS corrige-t-il le défaut d’autorisation du TP ?

A. Oui, systématiquement

B. Seulement si le port est 443

C. Non, transport et autorisation sont distincts

3. Quelle cible Nmap respecte notre TP ?

A. Le site d’un concurrent

B. 127.0.0.1 sur le seul port 8876

C. Tout le sous-réseau de l’entreprise

4. Pourquoi localhost et 127.0.0.1 peuvent-ils différer ?

A. localhost peut résoudre vers IPv6

B. 127.0.0.1 nécessite Internet

C. localhost est toujours une IP distante

5. Que faut-il éviter dans la preuve réseau ?

A. Le statut du TP

B. La méthode HTTP du TP

C. Un export global du navigateur personnel

MODULE 04 / BASES LINUX

Authentification, autorisation et intégrité des preuves

À l’issue du module : Distinguer identité et permission ; Vérifier une matrice d’accès ; Utiliser une empreinte à bon escient

L’authentification établit une identité selon un mécanisme choisi. L’autorisation décide ensuite ce que cette identité peut faire sur une ressource. Une session valide n’est donc pas une permission universelle. Le contrôle doit porter sur le document demandé, l’action et l’utilisateur. Dans notre application fictive, Alice possède le document 101 et Bob le document 102. Le numéro du document n’est pas un secret et ne peut pas remplacer une permission serveur.

Le laboratoire dispose de deux routes explicites. /api/demo-vulnerable/document/102 montre une erreur pédagogique : elle renvoie le document fictif de Bob à une requête représentant Alice. /api/demo-corrected/document/102 effectue la comparaison du propriétaire et refuse la demande. Ces routes ne représentent pas un logiciel réel à attaquer. La première est uniquement un contre-exemple local pour comprendre une référence directe à un objet insuffisamment contrôlée.

Un identifiant aléatoire peut réduire la facilité de deviner un objet, mais il ne constitue pas un contrôle d’accès. Une permission fiable vérifie le propriétaire ou le rôle à chaque requête. Il faut tester un accès autorisé et un accès interdit : corriger une faille en refusant toutes les opérations rendrait l’application inutilisable. La matrice du TP comprend Alice vers 101, Alice vers 102, Bob vers 102 et visiteur vers un document.

Une empreinte cryptographique est un autre outil. SHA-256 permet de comparer des octets avec une probabilité de collision négligeable dans cet usage courant. Le hachage n’est pas du chiffrement : on ne récupère pas le texte initial à partir de l’empreinte comme avec une clé de déchiffrement. Pour les mots de passe, une fonction rapide de hachage seule est inadaptée ; il faut un mécanisme de dérivation résistant aux essais massifs et correctement paramétré. Le cours n’enseigne aucune récupération de mots de passe réels.

La preuve d’un défaut d’autorisation contient le rôle utilisé, la ressource fictive, le résultat attendu, le résultat constaté et la correction. Elle ne contient pas une collection de documents d’autres personnes. Le TP d’empreinte porte sur preuve.txt dans votre dossier de travail. Après une modification, comparez les deux valeurs et notez la version. Un changement attendu d’empreinte prouve le changement des octets, pas une intention malveillante. Cette nuance sera essentielle au projet final.

Sessions, cookies et frontières du navigateur

Une application réelle peut conserver une session serveur et un cookie de référence. HttpOnly réduit l’accès au cookie par JavaScript, Secure limite sa transmission aux connexions sécurisées et SameSite influence les envois entre sites. Ces attributs répondent à des risques différents et ne remplacent pas une permission sur un document. Un jeton de session doit rester hors des captures publiques. Le header X-Lab-User de notre laboratoire est volontairement contrôlé par l’élève : il simplifie le scénario et ne doit jamais être décrit comme une authentification de production.

CSRF, origine et opérations qui changent un état

Une opération qui change un état doit vérifier son contexte et ses permissions. La protection contre les requêtes forgées dépend notamment du mécanisme de session, de la vérification d'origine et des jetons adaptés. Un refus d'origine peut compléter une défense, mais ne donne pas un droit à un utilisateur connecté. Le TP actuel effectue des lectures GET sur des données fictives ; il ne démontre pas une protection CSRF complète. Dans le rapport, indiquez ce point hors couverture plutôt que d'annoncer une sécurité générale de session.

Sessions, permissions et tests de frontière

L'authentification établit une identité selon un mécanisme ; l'autorisation décide ensuite si cette identité peut réaliser une action sur un objet. Une session associe des demandes successives à un état côté application. Un cookie de session réel doit être traité comme un secret. Dans notre laboratoire, l'en-tête X-Lab-User sert seulement à choisir Alice ou Bob. Il est modifiable par l'élève et ne démontre donc pas la sécurité d'une authentification. Le TP isole volontairement le contrôle d'accès pour rendre le défaut visible.

Construisez les cas positifs et négatifs avant d'essayer la route. Le cas positif Alice → 101 protège l'usage légitime. Le cas négatif Alice → 102 vérifie la séparation des documents. Bob → 102 doit continuer à fonctionner. L'absence d'identité reçoit un refus distinct. Si vous ne testez que le refus, vous pouvez prendre une application qui refuse tout pour une correction réussie. L'analyse d'un contrôle d'accès exige aussi une fonction qui reste utilisable par le propriétaire.

Un identifiant difficile à deviner réduit certaines découvertes, mais ne remplace pas la décision de permission. Un bouton masqué dans l'interface ne protège pas une route que le serveur accepte encore. La correction pédagogique compare le propriétaire attendu à l'identité supposée connue. Dans une application réelle, cette identité doit provenir du mécanisme d'authentification du serveur, et les droits doivent être évalués pour chaque action. Ne réutilisez pas notre en-tête de démonstration comme architecture de connexion.

Cookies, jetons et clés API ont des usages différents ; leurs noms ne suffisent pas à garantir leur sécurité. Pour une première revue, demandez où le secret est créé, où il est conservé, combien de temps il reste utilisable et comment il est révoqué. Un secret placé dans une URL peut se retrouver dans des journaux. Décrivez les mécanismes sans collecter de secrets réels. Le renouvellement, l'expiration et les essais de permissions plus complexes appartiennent à des laboratoires dédiés ultérieurs.

Hachage, chiffrement et signature

Le chiffrement vise la confidentialité et dépend d'une clé permettant l'opération inverse dans le modèle prévu. Un hachage calcule une empreinte à partir d'octets ; il ne permet pas de « déchiffrer » le fichier. Une signature numérique associe un mécanisme cryptographique à une clé privée et une vérification par clé publique ; son interprétation dépend aussi de la confiance accordée à cette clé. Aucun de ces mécanismes ne remplace une permission sur une ressource.

Pour vérifier une copie de preuve, comparez son empreinte SHA-256 avec une référence acquise dans des conditions connues. Si un fichier malveillant et son empreinte sont fournis par la même source compromise, leur cohérence n'établit pas la confiance. Une empreinte sans origine, heure ni chemin est une information incomplète. Elle ne constitue pas à elle seule une chaîne de conservation.

Pourquoi un mot de passe n’est pas une empreinte de fichier

SHA-256 est adapté à l’exercice d’intégrité, mais n’est pas une recommandation de stockage direct des mots de passe. Un mécanisme de dérivation adapté aux mots de passe introduit un coût de calcul et un sel propre à chaque enregistrement. OWASP présente notamment Argon2id dans sa documentation. Les paramètres dépendent du système et évoluent : ce manuel n’impose pas une constante universelle ni une implémentation artisanale.

Le sel n’a pas à être secret ; il évite notamment que des mots de passe identiques produisent le même enregistrement lorsque le reste du mécanisme est comparable. Un éventuel secret supplémentaire, souvent appelé pepper, a une gestion différente et doit être protégé séparément. Pour une application réelle, privilégiez des bibliothèques éprouvées et un service d’authentification maintenu. Le cours ne demande ni de casser des mots de passe ni de collecter leurs empreintes.

Session et permission : dessiner la frontière

Une session permet au serveur de rattacher plusieurs requêtes à un contexte reconnu. Les propriétés Secure, HttpOnly et SameSite d’un cookie répondent à des risques différents ; aucune ne dispense de contrôler l’autorisation à chaque action. La déconnexion devrait invalider le contexte selon le modèle de l’application. Une interface qui masque un bouton ne protège pas la route appelée par ce bouton.

Exercice sur papier : ajoutez une colonne « identité établie » puis une colonne « propriétaire du document » à la matrice Alice/Bob. Corrigé : l’identité fournit le sujet de la décision ; la propriété et la politique fournissent son droit. Le serveur du TP suppose une identité à partir de X-Lab-User uniquement pour illustrer cette deuxième étape. Ne transposez pas cet en-tête à une vraie connexion.

Où le serveur décide-t-il ?

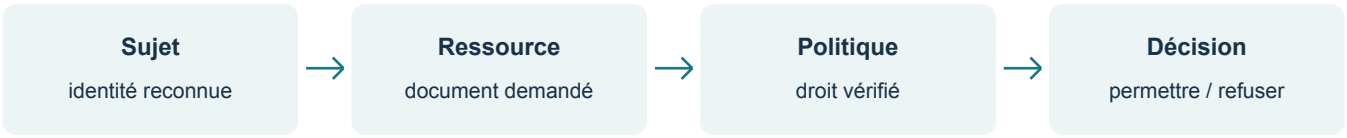


Schéma original VLR : le TP suppose une identité ; il illustre ensuite la permission.

Méthode pas à pas

À partir du navigateur du laboratoire, lancez le test d’accès au document 102 avec l’identité pédagogique Alice, puis le même test sur la route corrigée. Enregistrez seulement le statut et la référence fictive. Complétez la matrice de permission avec le client fourni. Créez preuve.txt, calculez son empreinte avec l’outil natif de votre OS, modifiez une ligne et recalculez. Gardez les deux preuves distinctes : accès à un objet et intégrité d’un fichier.

Votre exercice

Expliquez pourquoi une session connectée et un identifiant complexe ne suffisent pas. Fournissez la matrice attendue/observée et deux empreintes de votre fichier de TP. Proposez un contrôle qui conserve les accès légitimes.

Cas entièrement fictif

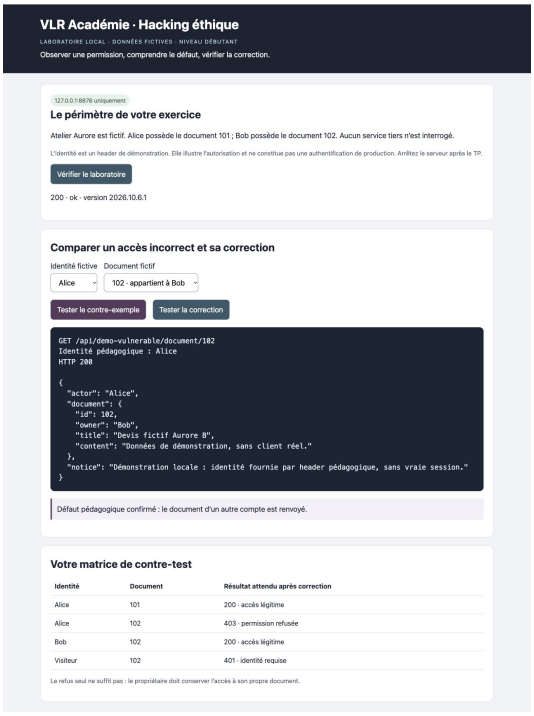
Cas fictif : Atelier Aurore utilise des numéros de documents difficiles à deviner. Une utilisatrice reçoit malgré tout la réponse d’un autre compte en changeant la référence dans le laboratoire.

Corrigé raisonné

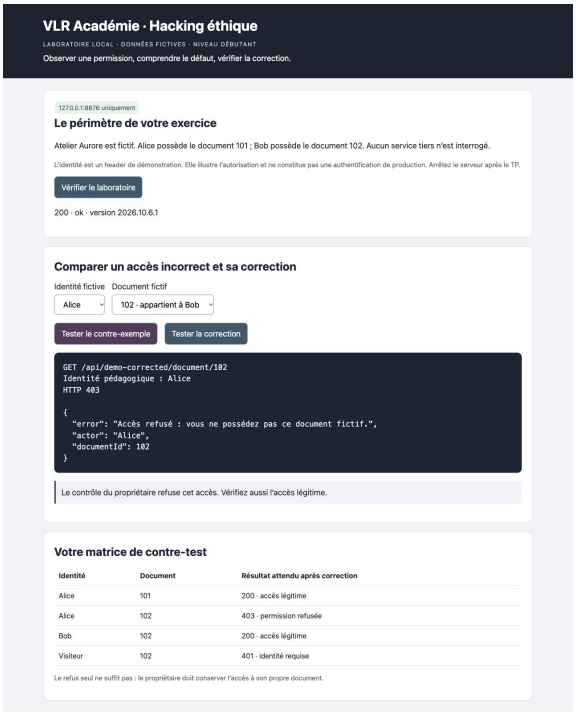
Le défaut se situe dans la vérification de permission, pas dans la longueur du numéro. La route corrigée autorise Alice vers 101 et refuse Alice vers 102 ; le visiteur n’a pas de droit. La matrice valide aussi Bob vers 102. L’empreinte du fichier change après la modification et ne donne aucun renseignement sur l’auteur. Le rapport décrit le mécanisme et le correctif sans publier de jeton réel ou de contenu sensible.

PREUVE ILLUSTRÉE / LABORATOIRE LOCAL

Comparer avant et après



Avant : Alice reçoit le document fictif de Bob dans la route de contre-exemple. Capture réelle du laboratoire local sur macOS, 6 octobre 2026.



Après : la route corrigée refuse cette demande avec HTTP 403. Capture réelle du laboratoire local sur macOS, 6 octobre 2026.

La capture du refus doit être complétée par les tests du propriétaire légitime. Ces images ne simulent pas une interface native Windows ou Linux.

MODULE 04 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Dans le TP, X-Lab-User est…

A. Une identité pédagogique modifiable

B. Une session de production sécurisée

C. Une signature cryptographique

2. Quel résultat est attendu pour Alice → 102 après correction ?

A. 200

B. Un accès à tous les documents

C. 403

3. Quel contrôle positif doit rester valide ?

A. Le serveur refuse tout

B. Bob → 102 reçoit 200

C. Alice reçoit tous les documents

4. Un identifiant aléatoire remplace-t-il la permission ?

A. Non

B. Oui dans tous les cas

C. Oui dès que le bouton est caché

5. Que fait l’authentification ?

A. Elle attribue automatiquement tous les droits

B. Elle chiffre toujours tous les fichiers

C. Elle établit une identité selon un mécanisme

MODULE 05 / BASES LINUX

Lire une entrée non fiable et corriger son traitement

À l'issue du module : Séparer données et instructions ; Corriger une requête locale ; Tester le comportement légitime

Une application reçoit des chaînes depuis un formulaire, une URL, un fichier ou une API. Leur origine ne garantit pas leur innocuité. Le traitement doit préciser le type attendu, la longueur, la structure et la façon d'utiliser la valeur. La validation ne dispense pas d'utiliser des interfaces sûres : une requête SQL doit recevoir ses paramètres par le mécanisme prévu, plutôt que par concaténation de texte.

Le TP fourni travaille dans une base SQLite en mémoire avec deux produits fictifs. Il compare une requête construite par concaténation et une requête paramétrée. Une entrée contenant une apostrophe suffit à montrer que texte de recherche et syntaxe SQL peuvent se mélanger. L'objectif n'est pas d'extraire des données d'un site : on lit le code local et on constate comment les deux fonctions traitent la même valeur. La correction maintient la recherche légitime et empêche l'entrée de changer la structure de la requête.

Une sortie HTML a ses propres contraintes. Une chaîne affichée avec textContent est traitée comme du texte. Une chaîne passée à innerHTML est interprétée comme du balisage. Le laboratoire compare le rendu d'une balise simple sans script, comme Bonjour. Il ne fournit pas de charge de vol de session, d'exfiltration ou d'appel externe. Le rendu différent montre le problème de contexte ; il ne constitue pas, à lui seul, un test complet de protection contre les scripts intersites.

Chaque contexte nécessite son traitement : contenu HTML, attribut, URL et code JavaScript ne sont pas interchangeables. Un assainissement appliqué au mauvais endroit peut créer une fausse assurance. Pour un affichage de texte, choisissez d'abord l'API qui n'interprète pas le contenu. Pour une URL, contrôlez les protocoles et les destinations admises. Pour un fichier, limitez l'emplacement et les droits. N'ajoutez pas une liste arbitraire de mots interdits en espérant couvrir tous les cas.

La validation d'un correctif comprend des entrées ordinaires, une entrée vide, une valeur longue et le contre-exemple observé. Si le correctif casse la recherche d'un nom contenant une apostrophe, il a supprimé une fonction utile. Le rapport doit dire exactement quelle fonction a été testée et quelles autres interfaces restent à vérifier. Il ne faut pas annoncer « plus aucune injection » après une seule requête paramétrée dans un exemple.

Reconnaître plusieurs familles de défauts

Le défaut d'autorisation concerne la permission d'une identité sur une ressource. L'injection SQL concerne la séparation entre valeur et instruction de base de données. Un rendu HTML non maîtrisé concerne l'interprétation d'une sortie par le navigateur. La traversée de chemin concerne la possibilité de sortir d'un emplacement prévu pour un fichier ; une désérialisation non fiable concerne le traitement d'un objet reçu. Les exercices obligatoires démontrent les deux premières familles sur du code local. Les autres sont étudiées comme mécanismes et ne doivent pas être présentées comme des vulnérabilités testées du serveur de cours.

Revue de code et test dynamique

Une revue statique lit le code sans exécuter l'application. Un test dynamique observe le comportement en fonctionnement. Dans le TP SQL, vous faites les deux : lecture de la concaténation, puis comparaison des résultats. La revue suggère un problème et le test vérifie un cas précis. Ni une ligne jugée correcte ni un outil automatique ne garantit la qualité de tout le programme. Une fiche de constat cite la fonction, la condition, le résultat et un test de non-régression. Le correctif utilise l'interface sûre au lieu d'ajouter une liste de mots interdits.

Injection, encodage et règles de validation

Le TP SQL compare une donnée et une instruction. Une requête qui concatène directement l'entrée peut laisser cette entrée modifier la structure de la demande. La recherche paramétrée transmet au moteur la valeur à comparer, au lieu de lui ajouter un fragment de requête. Testez aussi une valeur ordinaire contenant une apostrophe : la fiabilité du traitement des données légitimes compte autant que le refus du contre-exemple. Les trois produits de la base en mémoire sont fictifs et aucune base du site n'est interrogée.

Valider signifie vérifier qu'une donnée respecte une règle de l'application : type, longueur, plage ou ensemble permis. Encoder signifie produire une représentation adaptée au contexte de sortie. Paramétrer une requête sépare les données de la commande SQL. Ces mesures répondent à des problèmes différents. Une règle qui limite la longueur d'un nom ne remplace pas le paramétrage SQL ; le paramétrage ne décide pas si l'utilisateur a le droit de lire le produit. Expliquez quelle propriété chaque correction protège.

Dans une page HTML, traiter du texte comme du balisage peut créer une autre classe de défaut. Le laboratoire affiche sa réponse JSON comme du texte dans l'interface, sans demander au navigateur d'exécuter ce texte. Un contrôle débutant consiste à observer le mécanisme de sortie et à distinguer textContent d'un assemblage de HTML non maîtrisé. Nous n'injectons pas de script dans un site tiers. Les scénarios XSS détaillés, les politiques de contenu et les contextes JavaScript exigent des exercices spécifiques de niveau suivant.

Une liste de chemins autorisés permet aussi d'éviter d'exposer accidentellement des fichiers. Notre application ne transforme pas un chemin arbitraire de l'utilisateur en nom de fichier à lire. Pour relire un autre laboratoire, cherchez quels fichiers peuvent être servis, où la décision est prise et ce que produit un chemin absent. Le test doit vérifier un refus attendu sans tenter de lire les fichiers du système. Un rejet propre est utile ; une trace d'erreur contenant un chemin privé peut révéler plus d'informations que nécessaire.

Validation, paramétrage et encodage

La validation vérifie qu'une donnée respecte le contrat attendu : type, longueur, valeurs admises et relations utiles. Le paramétrage SQL conserve la séparation entre la structure de la requête et ses valeurs. L'encodage de sortie représente une donnée selon son contexte de destination, par exemple du texte HTML. Ces opérations sont complémentaires ; retirer toutes les apostrophes n'est pas une stratégie générale.

Pour un identifiant de document, valider un entier positif ne dit pas si l'utilisateur peut accéder au document. Pour un nom de produit, accepter une apostrophe permet un usage légitime ; le paramètre SQL évite de la traiter comme une partie de la syntaxe. Pour un commentaire affiché dans une page, l'encodage dépend du contexte. Un texte sûr dans un paragraphe n'est pas automatiquement sûr dans une URL ou un script.

Lire du code sans chercher un mot magique

Repérez d'abord la source de la donnée, son chemin dans les fonctions et le point où elle est utilisée. Identifiez une frontière : requête HTTP, fichier, base, interpréteur ou affichage. Cherchez ensuite le contrôle réellement appliqué. Un nom de fonction rassurant ne constitue pas une preuve ; une bibliothèque peut être bien utilisée à un endroit et mal utilisée ailleurs.

Question de revue	Exemple de constat	Correction à vérifier
D'où vient la valeur ?	Paramètre reçu du client	Type et limites explicites
Qui peut agir ?	Identité supposée connue	Autorisation sur la ressource

Question de revue	Exemple de constat	Correction à vérifier
Comment SQL reçoit-il la valeur ?	Chaîne concaténée	Paramètre séparé
Que voit le lecteur ?	Donnée injectée dans HTML	Encodage selon le contexte
Que garde le journal ?	Corps complet de requête	Minimisation des données

Exercice : pour les trois entrées du TP SQL, indiquez comportement attendu, comportement fragile et comportement corrigé. Ajoutez un test légitime avec apostrophe. Corrigé : la recherche paramétrée doit préserver les données textuelles sans permettre qu'elles changent la structure SQL. Un rejet global des caractères spéciaux masquerait le problème et dégraderait l'usage.

Reconnaître sans exploiter hors du laboratoire

Une injection SQL concerne une frontière avec le langage de requête. Une injection de commandes concerne un interpréteur système. Un défaut XSS concerne un contenu qui peut être exécuté dans un contexte de navigateur. Un parcours de chemin peut dépasser le dossier prévu si le serveur ne maîtrise pas la résolution. Une requête côté serveur mal contrôlée peut atteindre une destination non prévue. Ce niveau apprend à distinguer ces familles et leurs mesures de prévention ; il ne fournit aucun mandat pour les essayer sur Internet.

Les futurs ateliers intermédiaires pourront examiner des applications locales volontairement vulnérables avec un plan de tests dédié. Pour l'instant, la preuve exécutée est celle du laboratoire distribué, avec ses routes et données fictives. Ne décrivez pas un scénario seulement lu comme un essai que vous auriez réalisé.

Méthode pas à pas

Exécutez tp_sql.py : le script n'utilise que sqlite3 et sa base en mémoire. Relevez les sorties pour un nom ordinaire, un nom avec apostrophe et l'entrée de démonstration incluse. Lisez les deux fonctions et identifiez le passage entre donnée et syntaxe. Dans la page du laboratoire, comparez l'affichage de la balise de texte proposée avec les deux méthodes. Le mode corrigé affiche le balisage littéralement. Écrivez des tests de non-régression sur le fonctionnement attendu.

Votre exercice

Remplacez, dans une copie de tp_sql.py, la construction concaténée par une requête avec paramètres. Fournissez une explication du rôle du paramètre et un tableau d'entrées et de résultats. Ajoutez un exemple qui ne devrait pas être rejeté, tel qu'un nom avec apostrophe.

Cas entièrement fictif

Cas fictif : un développeur interdit le caractère apostrophe pour corriger la recherche. Le nom fictif d'un produit n'est plus retrouvable, mais la séparation entre donnée et commande n'est toujours pas expliquée.

Corrigé raisonné

La correction utilise une requête paramétrée avec la valeur fournie séparément ; l'apostrophe devient une donnée ordinaire. Le tableau contient les cas valides et le contre-exemple, pas seulement un échec spectaculaire. L'affichage HTML emploie textContent pour le champ prévu comme texte. Les essais ne concernent ni une base d'entreprise ni une route Internet. La conclusion est limitée aux fonctions étudiées et indique les contextes de sortie non couverts.

MODULE 05 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Le paramétrage SQL sépare…

A. Les données et la structure de la requête

B. Le propriétaire et le groupe du fichier

C. Les ports et les interfaces

2. Pourquoi tester Bureau d’étude avec une apostrophe ?

A. Pour prouver une attaque externe

B. Pour supprimer la base

C. Pour vérifier le traitement d’une donnée légitime

3. Une limite de longueur remplace-t-elle le paramétrage SQL ?

A. Oui si le champ est obligatoire

B. Non

C. Oui pour tous les champs

4. Dans quel environnement tourne notre TP SQL ?

A. Une base SQLite en mémoire

B. La base Turso du site

C. Un serveur client réel

5. Quel mécanisme affiche une chaîne comme texte dans la page ?

A. Une concaténation HTML non maîtrisée

B. La désactivation TLS

C. textContent

MODULE 06 / BASES LINUX

Inventorier, vérifier et prioriser une vulnérabilité

À l’issue du module : Constituer un inventaire utile ; Écarter un faux positif ; Expliquer une priorité

Un inventaire indique ce qui existe, son propriétaire, sa fonction et sa version observée. Il précède une recherche de vulnérabilités, car un nom de produit ne suffit pas. Deux systèmes peuvent afficher une version semblable et recevoir des correctifs différents selon leur distribution ou leur éditeur. Les paquets Linux peuvent intégrer une correction rétroportée sans adopter le numéro de version attendu par un simple détecteur.

Une CVE désigne une vulnérabilité référencée, pas une preuve que votre système y est exposé. Vérifiez le produit, la plage de versions, la configuration nécessaire, les conditions d’attaque et l’avis du fournisseur. Si une version ne peut pas être déterminée, écrivez « à confirmer ». Le parcours utilise un inventaire fictif et des avis de démonstration datés, sans affirmer qu’une vulnérabilité historique existe encore sur un produit actuel.

Un outil de détection produit des observations et parfois des hypothèses. Un service à l’écoute ou un en-tête absent ne justifie pas automatiquement un niveau critique. Une bibliothèque présente dans un fichier ne prouve pas que la fonction vulnérable est accessible. La validation manuelle consiste à relier la condition annoncée à la configuration observée, avec un essai minimal permis par le mandat. Le niveau débutant n’exécute pas un programme trouvé au hasard sur un dépôt public pour « confirmer » une alerte.

La sévérité technique et la priorité métier sont différentes. CVSS fournit un cadre de description de sévérité ; son score de base ne suffit pas à exprimer le risque de l’organisation. Une exposition Internet, une donnée importante, un contrôle compensatoire et une activité d’exploitation connue peuvent modifier la priorité. Si vous utilisez un score, indiquez sa version et le vecteur ou la source ; ne fabriquez pas un nombre pour rendre le rapport impressionnant.

Le dossier Atelier Aurore présente trois constats : contrôle d’accès incorrect dans une route de démonstration, serveur de laboratoire limité à la boucle locale et fichier de test modifié. Le premier demande une correction applicative démontrable. Le deuxième décrit une exposition volontairement réduite et ne devient pas une faille parce que le port est ouvert. Le troisième peut être normal si la modification était prévue. Le bon analyste sait également écarter un faux positif et décrire une limite de visibilité.

Un inventaire technique utilisable

Consignez OS, architecture, version de l’interpréteur, version du laboratoire, commande de lancement, adresse d’écoute et fichiers de TP. Ajoutez un propriétaire et un usage à chaque actif. Une liste sans version ni rôle rend l’analyse difficile. Les commandes d’inventaire peuvent produire beaucoup d’informations ; le rapport ne conserve que celles du mandat. Les paquets et dépendances externes ne sont pas nécessaires au laboratoire standard. Un ajout facultatif, comme Nmap, est documenté avec sa source et sa version plutôt que considéré comme déjà présent.

Preuve minimale, faux positif et décision

Un constat confirmé doit relier une condition du système à un résultat. Un faux positif est une alerte que les vérifications ne soutiennent pas dans ce contexte. Une information insuffisante reste à vérifier. Ces trois états doivent être conservés séparément. Dans l’application fictive, Alice recevant 102 est confirmé ; l’ouverture du port de boucle locale est prévue ; la version d’un outil non relevée reste inconnue. Le responsable peut ainsi corriger le premier point, conserver le deuxième et organiser la lecture manquante pour le troisième.

Prioriser, confirmer et écarter les faux positifs

Une vulnérabilité est un défaut susceptible de produire un effet indésirable dans certaines conditions. Un résultat de scanner est une observation à examiner. Distinguez la présence d'un composant, sa version annoncée, sa configuration réelle et les conditions d'exploitation. Le laboratoire d'autorisation possède volontairement un défaut confirmé par une paire de tests ; un port ouvert, à lui seul, n'est pas un défaut. Dans le rapport, expliquez la chaîne de preuve plutôt que de reprendre une alerte comme verdict.

CVE identifie des vulnérabilités documentées ; CWE classe des familles de faiblesses. Un identifiant ne dit pas que votre machine est concernée. Vérifiez le produit, la version, la configuration et l'avis de l'éditeur. Le calcul CVSS décrit une sévérité technique selon une version et un vecteur. Il ne remplace pas le risque propre à l'organisation. Pour ce cours, une priorité raisonnée suffit si elle distingue impact fictif, conditions, exposition et solution. Aucun score CVSS artificiel n'est attribué à un défaut sans justification.

ZAP propose notamment une observation passive des échanges qui lui sont confiés. Une analyse passive ne se confond pas avec un balayage actif, qui envoie des demandes supplémentaires. Pour un débutant, les outils du navigateur et la lecture du code fourni sont suffisants. Si ZAP est déjà disponible dans votre laboratoire, observez uniquement le service local et décrivez les messages reçus. Ne lancez pas l'attaque active par défaut et ne changez pas les certificats de votre navigateur personnel pour contourner un problème de connexion.

Un faux positif pédagogique utile est l'alerte « HTTP sans TLS » sur un service de démonstration limité à 127.0.0.1 et à des données fictives. Le contexte n'autorise pas à déployer une application réelle de cette façon. Il permet de dire que le défaut supposé de transport ne démontre pas ici une exposition Internet. Le rapport mentionne la limite et conserve le vrai défaut de permission. Une bonne priorisation retire des conclusions excessives sans dissimuler les problèmes confirmés.

CVE, CWE et preuve d'applicabilité

CVE identifie une vulnérabilité publiée ; CWE classe des types de faiblesse. Un produit portant une version proche de celle d'un avis ne prouve pas l'applicabilité de cet avis. Il faut vérifier l'édition, la configuration, les composants, les conditions d'exploitation et les correctifs éventuellement rétroportés. Le fournisseur constitue la source principale pour ses versions corrigées et ses mesures de réduction du risque.

Un outil de détection produit des indications qui peuvent être incomplètes. Une bannière peut être ancienne, masquée ou différente du paquet installé. Notez « à confirmer » tant que les conditions ne sont pas établies. Un faux positif documenté est utile : il montre ce qui a été examiné et pourquoi l'alerte a été écartée, sans inventer une faiblesse pour compléter le rapport.

Une priorisation qui ne dépend pas d'un seul nombre

Le score CVSS décrit une sévérité technique selon les métriques choisies. Le risque de l'organisation dépend également des actifs, de l'exposition, des protections, de l'exploitation observée et des conséquences. Conservez la version du cadre et le vecteur lorsqu'un avis les fournit. Ne transformez pas automatiquement un score de base en délai contractuel de correction.

Exercice fictif : un composant non exposé porte un avis applicable ; une route locale de démonstration divulgue un document fictif. Rédigez deux fiches distinctes sans les classer arbitrairement « critiques ». Corrigé : vous disposez d'informations techniques et d'un contexte pédagogique, mais pas des données métier permettant une estimation financière. Décrivez les conditions et proposez les vérifications manquantes.

Méthode pas à pas

Inventoriez uniquement le programme de laboratoire et les fichiers de TP. Pour chaque constat, notez actif, condition, observation, conséquence et incertitude. Utilisez le registre fictif fourni pour classer confirmé, faux positif et à vérifier. Pour un avis officiel de votre OS, relevez organisme, date et conditions d’application ; ne remplacez pas cette lecture par un résumé de moteur de recherche. Donnez une priorité justifiée en langage clair avant tout score.

Votre exercice

Complétez les trois fiches de constat du cas Atelier Aurore. Identifiez une hypothèse qui ne peut pas être confirmée à partir du seul numéro de version. Proposez les éléments supplémentaires à obtenir.

Cas entièrement fictif

Cas fictif : un outil associe une ancienne CVE au nom d’une bibliothèque. L’éditeur a rétroporté un correctif ; l’apprenant doit examiner l’avis et le paquet au lieu d’annoncer une intrusion.

Corrigé raisonné

Une alerte devient confirmée lorsque les conditions et la preuve correspondent au système étudié. L’écoute locale prévue n’est pas une ouverture Internet ; le fichier modifié volontairement n’est pas une preuve de malware. Le rapport indique l’avis du fournisseur, le produit et les limites de la vérification. La priorité métier s’appuie sur impact, exposition et contexte. Aucune CVE ou statistique d’exploitation actuelle n’est inventée dans le dossier pédagogique.

MODULE 06 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Une alerte de scanner est…

A. Une observation à confirmer

B. Toujours un verdict définitif

C. Une autorisation de mission

2. Un CVE suffit-il à prouver qu’un poste est concerné ?

A. Oui quel que soit le logiciel

B. Oui si le numéro est récent

C. Non, les conditions doivent correspondre

3. CVSS décrit principalement…

A. Une autorisation légale

B. Une sévérité technique selon un vecteur

C. Le chiffre d’affaires de l’entreprise

4. Une analyse passive ZAP se distingue d’une active parce que…

A. Elle observe les échanges qui lui sont confiés

B. Elle attaque toutes les routes automatiquement

C. Elle remplace le mandat

5. Le HTTP local du TP permet de conclure…

A. Que tout site public peut fonctionner sans TLS

B. Qu’aucune permission ne doit être testée

C. Qu’il faut préciser le contexte avant de parler d’exposition Internet

MODULE 07 / BASES LINUX

De la preuve à la correction et au contre-test

À l'issue du module : Rédiger un constat reproductible ; Proposer une correction ciblée ; Vérifier sans casser l'usage

Un rapport permet à une autre personne de comprendre le problème et de décider quoi faire. Son premier lecteur peut être un responsable non spécialiste. Commencez par la conséquence concrète, puis précisez le mécanisme, les conditions et la preuve. « Une identité de démonstration reçoit le document d'une autre » est plus utile que « système vulnérable » sans contexte. Le titre doit décrire le défaut sans transformer un essai local en incident réel.

Une fiche de constat contient un identifiant, l'actif, la date, la version, les permissions utilisées, le résultat attendu, le résultat obtenu, l'impact limité au scénario, le correctif proposé et le contre-test. Les étapes sont suffisamment précises pour être reproduites dans le même laboratoire. Vous ne promettez pas une reproduction sur une machine dont la version et les règles diffèrent. Les captures et extraits ne doivent pas faire apparaître des secrets qui ne sont pas nécessaires à la compréhension.

La qualité de preuve dépend de sa traçabilité, pas du nombre d'images. Conservez l'original de la capture et une copie annotée si nécessaire. Une annotation doit expliquer une zone visible, pas ajouter un résultat jamais observé. Notez le moment de l'essai et le script utilisé. Un journal pédagogique peut contenir des événements fictifs ; le rapport doit le signaler plutôt que les présenter comme une collecte sur le poste de travail d'un client.

La correction porte sur la cause. Pour l'accès au document, elle vérifie la permission côté serveur à chaque demande. Pour une recherche SQL, elle sépare la valeur et la structure de la requête. Pour un fichier sensible dans une VM de TP, elle restreint les droits selon le besoin et vérifie l'accès prévu. Un conseil général comme « renforcer la sécurité » ne donne ni action, ni responsable, ni moyen de contrôler le résultat.

Le contre-test répète le scénario interdit et un scénario légitime après correction. Le refus de l'accès illégitime doit être conservé ; l'usage autorisé doit continuer à fonctionner. Une preuve après correction n'efface pas le constat initial. Le rapport garde les deux états, la date et la version de chacun. Enfin, il explique la couverture : le test d'une route ne valide pas toutes les routes, le résultat du QCM n'atteste pas une compétence universelle et le laboratoire ne remplace pas une mission professionnelle supervisée.

La chaîne de conservation des preuves

Pour chaque preuve, notez qui l'a produite dans le TP, quand, avec quel outil et depuis quel état du laboratoire. Conservez l'original et distinguez une annotation de sa source. Une empreinte permet de détecter un changement entre copies, mais elle ne crée pas à elle seule une chaîne de confiance. Dans un exercice pédagogique, cette discipline sert à rendre le dossier reproductible. Elle ne constitue pas une procédure judiciaire certifiée et ne permet pas d'attribuer une action à une personne réelle.

Relecture et restitution

Une restitution présente d'abord les conséquences et les actions, puis laisse le détail technique consultable. Le destinataire doit pouvoir relier chaque demande de correction à une preuve et à un critère de contre-test. Signalez les opérations non exécutées et les limites propres à l'OS. Relisez les dates, les noms de fichiers, les identités fictives et les versions. Un vocabulaire spectaculaire ne compense pas une preuve absente. La bonne restitution rend la suite praticable, sans transformer un apprentissage en mission réelle réussie.

Un dossier qu’un autre apprenant peut reproduire

Une preuve comporte un contexte, une action et un résultat. Numérotez-la, indiquez la version du laboratoire et expliquez les données fictives. Une capture seule peut montrer un résultat sans expliquer comment il a été obtenu. Joignez la méthode ou la matrice d’essais. Dans le rapport d’Atelier Aurore, le lecteur doit pouvoir retrouver la requête d’Alice, le propriétaire du document et la différence entre les deux routes. Il ne doit pas avoir à deviner si l’image vient de Windows, Linux ou macOS.

Pour chaque constat, séparez observation, conséquence dans le cas fictif, cause probable, correction et vérification. Une cause probable reste une hypothèse tant que vous n’avez pas lu ou testé le mécanisme correspondant. La recommandation précise qui change quoi et comment vérifier le résultat. Le contre-test comprend le refus de l’accès incorrect et le maintien de l’accès légitime. Les résultats « non exécuté », « inaccessible » et « conforme » n’ont pas le même sens et ne doivent pas être fusionnés.

Votre résumé destiné à un responsable peut tenir en une page : périmètre, deux défauts confirmés, conséquence, décisions et limite. L’annexe conserve les détails techniques. Évitez de multiplier les captures qui répètent la même observation. Masquez tout élément extérieur au TP et gardez une copie de référence des preuves utiles. L’empreinte permet de comparer des fichiers, mais n’établit pas à elle seule leur origine. La provenance écrite reste nécessaire.

Un rapport final n’efface pas le registre d’essais. Il le sélectionne et le rend lisible. Avant livraison, demandez à une autre personne de suivre la méthode sur son propre laboratoire ; si cette relecture n’a pas eu lieu, ne l’affirmez pas. La grille formative du projet est une autoévaluation, distincte du QCM. Vous pouvez réussir le questionnaire et avoir encore un rapport à améliorer. Le portfolio explique cette différence avec honnêteté.

Un constat complet, de la preuve à la décision

Un constat comprend un titre précis, l’actif, la condition préalable, les étapes bornées, l’observation, l’impact plausible, la correction et le contre-test. L’impact doit découler de la preuve. L’accès à un document fictif n’autorise pas à affirmer une compromission complète, une persistance ou une exfiltration massive. Si une conséquence est seulement possible, indiquez qu’il s’agit d’une hypothèse.

Exemple original : « AUTH-02 : la route de contre-exemple permet à Alice de lire 102 ». Condition : laboratoire local et identité pédagogique Alice. Observation : statut 200 et propriétaire Bob dans la réponse prévue. Correction : comparaison côté serveur de l’identité avec la politique de propriété. Contre-tests : Alice/102 refusé, Alice/101 permis, Bob/102 permis et visiteur/102 non authentifié. Le rapport nomme la route corrigée et distingue ses résultats.

Mesurer la qualité de la preuve

Une capture utile montre la requête ou l’action, l’identité fictive, le résultat pertinent et le contexte du TP. Elle ne doit pas exposer l’ensemble du bureau personnel. Conservez le fichier original ; recadrez une copie pour la restitution et mentionnez le traitement. Le texte doit rester lisible après insertion dans le rapport. Une capture décorative ne remplace pas le résultat enregistré.

Le journal d’activité décrit ce que vous avez réellement fait. Ajoutez les essais infructueux utiles et les limites de visibilité. Un horodatage UTC facilite la comparaison, mais des horloges non synchronisées peuvent créer un écart. Une empreinte vérifie la cohérence du fichier conservé ; elle ne prouve pas la justesse de l’horloge ou la véracité d’un commentaire.

Relire avec deux lecteurs

Le premier lecteur doit comprendre la décision sans relancer le laboratoire. Le second doit pouvoir reproduire le test à partir des étapes, de la version et des données. Relisez donc le résumé pour sa clarté et les annexes pour leur reproductibilité. Vérifiez les chemins, les statuts, les légendes, les références et la présence des contre-tests. Retirez les secrets et les conclusions dépassant la preuve.

Du constat au dossier partageable

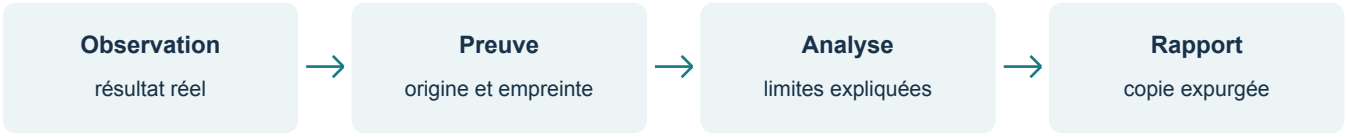


Schéma original VLR : chaque étape doit rester traçable et proportionnée.

Méthode pas à pas

Utilisez le modèle de rapport fourni. Rédigez le constat d’autorisation en deux versions : un résumé de cinq lignes pour la direction et une fiche technique reproductible. Associez une capture réelle du laboratoire, l’identité fictive et la version. Complétez le tableau de contre-test avec accès légitime et accès interdit. Relisez le rapport comme si vous n’aviez jamais vu le laboratoire : chaque conclusion doit être reliée à une preuve.

Votre exercice

Produisez une fiche de constat et une demande de correction qui indique propriétaire, priorité, action et critère d’acceptation. Trouvez une affirmation trop large dans votre texte et remplacez-la par une conclusion limitée aux essais réalisés.

Cas entièrement fictif

Cas fictif : le rapport initial contient dix captures et le mot « critique », mais aucune version ni correction. La responsable ne peut pas attribuer la tâche ni vérifier sa résolution.

Corrigé raisonné

Le rapport final préfère deux captures identifiées avant/après, la matrice d’accès, une description de l’impact fictif et le contrôle serveur à modifier. Il explique pourquoi la priorité est proposée et évite un score arbitraire. Le contre-test contient aussi Alice vers son propre document. Les éléments non étudiés restent mentionnés. Le document permet une action concrète et ne revendique ni audit complet de l’organisation ni incident réellement constaté.

MODULE 07 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Quelle structure rend un constat utile ?

A. Observation, conséquence, correction et contre-test

B. Logo, slogan et nombre de clics

C. Liste de commandes sans résultat

2. Une empreinte seule prouve-t-elle l’origine d’un fichier ?

A. Oui pour tout fichier

B. Oui si elle est longue

C. Non

3. Comment noter un test non réalisé ?

A. Réussi si la documentation existe

B. Non exécuté, avec la limite

C. Conforme par défaut

4. Que doit contenir la preuve de permission ?

A. L’identité fictive, la route, l’attente et le résultat

B. Un secret de session réel

C. Le journal global de l’ordinateur

5. À quoi sert la grille du projet ?

A. À prétendre qu’un jury a corrigé

B. À remplacer automatiquement le QCM final

C. À une autoévaluation formative

MODULE 08 / BASES LINUX

Linux : terminal, utilisateurs et permissions

À l'issue du module : Lire les permissions POSIX ; Utiliser un shell avec précision ; Justifier le moindre privilège

Linux fournit un modèle de fichiers et de permissions que l'on peut observer depuis le terminal. Le shell interprète les commandes ; il n'est pas le noyau. Une commande peut produire une sortie standard, une sortie d'erreur et un code de retour. Si vous redirigez tout sans distinction, vous pouvez perdre l'information qui explique l'échec d'un exercice. Gardez une trace de la commande et de sa sortie utile.

Les droits classiques indiquent lecture, écriture et exécution pour le propriétaire, le groupe et les autres. Pour un dossier, le droit d'exécution permet notamment de traverser le chemin ; il ne signifie pas « lancer le dossier ». Une permission de fichier doit être analysée avec celles des dossiers parents. Les ACL et les mécanismes de contrôle obligatoire peuvent compléter les droits classiques : une simple lecture de rwx ne décrit pas toujours toute la décision.

Créez uniquement votre dossier de TP. pwd montre l'emplacement courant, ls -l et stat permettent d'examiner les fichiers, id montre l'identité de la session. N'exécutez pas chmod de manière récursive sur votre dossier personnel ou sur un répertoire système. L'exemple chmod 600 preuve.txt concerne seulement le fichier fictif dont vous êtes propriétaire. Il autorise le propriétaire à lire et écrire et retire les droits classiques du groupe et des autres sur ce fichier.

sudo sert à exécuter une opération selon une délégation. Il n'est pas nécessaire pour calculer une empreinte de votre propre fichier. Une commande qui fonctionne seulement avec sudo mérite une explication du droit demandé. Le cours ne cherche pas à contourner une restriction d'administration ni à exploiter un mécanisme d'élévation de privilèges. L'exercice consiste à choisir des droits suffisants et à vérifier le résultat.

sha256sum permet de comparer les octets de preuve.txt. La ligne produite associe une empreinte et un chemin ; conservez les deux. Si le fichier change entre le calcul et la copie de la preuve, votre observation ne décrit plus le même état. Le TP fournit un fichier court et un arrêt entre les étapes. Notez les conditions de collecte et distinguez une modification volontaire d'un changement inexpliqué.

Comment documenter un essai

Conservez la version de l'OS, la commande exacte, le fichier ou processus de TP concerné, la sortie observée et sa limite. Les illustrations de commande ne remplacent pas une exécution native. Les manipulations d'administration restent dans une machine de laboratoire, avec restauration prévue ; les protections de la machine principale restent actives.

Comprendre propriétaire, groupe et permission

La sortie ls -l d'un fichier indique notamment des bits de permission et un propriétaire ; une ACL peut ajouter des règles. Pour le TP, lisez ls -l preuve.txt et sha256sum preuve.txt. Les permissions d'un dossier interviennent aussi dans l'accès au chemin. Le bit d'exécution d'un fichier et celui d'un répertoire ne représentent pas la même opération. Une configuration qui autorise tout le monde avec chmod 777 ne résout pas une erreur de conception. Étudiez les règles sur une copie autorisée et gardez les valeurs initiales avant une manipulation.

Vérification formative : expliquez à un pair le sens de chaque paramètre, le périmètre de la commande et une conclusion qu'elle ne permet pas. Si vous ne pouvez pas l'expliquer, relisez le support avant de l'exécuter. Livrable supplémentaire : trois colonnes commande, observation, limite. Une observation non exécutée reste marquée comme telle ; la documentation du fournisseur sert alors de référence et ne devient pas une preuve

d’exécution.

Comprendre les bits sans changer tout un dossier

Les droits classiques distinguent propriétaire, groupe et autres. Pour un fichier ordinaire, r signifie lire, w modifier et x autoriser l’exécution selon le contexte. Pour un dossier, x permet notamment de traverser le chemin ; r permet de lire la liste des noms et w participe à la création ou suppression d’entrées lorsque les autres conditions sont réunies. Les ACL et les contrôles obligatoires peuvent ajouter des règles.

```
pwd
id
ls -l -- preuve.txt
sha256sum -- preuve.txt
cp -- preuve.txt copie-preuve.txt
chmod 600 -- copie-preuve.txt
ls -l -- copie-preuve.txt
```

Ces commandes correspondent à un environnement GNU/Linux courant et s’exécutent dans le dossier du TP. La copie doit être absente avant cp pour éviter de remplacer un travail antérieur. chmod 600 sur votre propre copie donne lecture et écriture au propriétaire et retire les bits du groupe et des autres. Cela ne chiffre pas le contenu et ne garantit pas une protection contre un administrateur. Les commandes natives Linux ne sont pas déclarées exécutées dans cette édition.

Exercice : interprétez 640 puis 600. Corrigé : 640 accorde lecture/écriture au propriétaire, lecture au groupe et aucun de ces bits aux autres ; 600 retire aussi la lecture au groupe. Ce calcul décrit les bits classiques, pas l’ensemble des permissions effectives. N’utilisez pas chmod récursif sur un dossier système ou votre répertoire personnel pour ce TP.

Méthode pas à pas

Dans votre dossier de TP Linux : pwd ; id ; ls -l preuve.txt ; stat preuve.txt ; sha256sum preuve.txt. Relevez les droits et l’empreinte. Sur une copie locale dont vous êtes propriétaire, essayez chmod 600 preuve-copie.txt et relisez les droits. N’utilisez pas sudo, ne modifiez aucun fichier système et ne faites pas de changement récursif. Les exemples de sortie attendue doivent rester identifiés comme exemples tant que vous n’avez pas exécuté les commandes sur Linux.

Votre exercice

Expliquez la différence de sens du droit x entre fichier et dossier. Produisez les permissions et l’empreinte de votre fichier de TP, puis le résultat de la modification limitée sur sa copie.

Cas entièrement fictif

Cas fictif : un fichier est lisible par son propriétaire, mais un dossier parent empêche un compte fictif d’atteindre son chemin. L’apprenant conclut trop vite que le fichier manque.

Corrigé raisonné

L’analyse suit le chemin et les permissions des dossiers, pas seulement la dernière ligne. chmod 600 n’est appliqué qu’à la copie du TP. La réponse ne propose pas 777 comme correction universelle. L’empreinte prouve une comparaison d’octets et les résultats non exécutés restent signalés. Aucun accès à un compte tiers n’est nécessaire.

MODULE 08 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Quel outil compare le contenu sous Linux ?

A. sha256sum

B. ss

C. journalctl

2. chmod 777 constitue-t-il une correction universelle ?

A. Oui

B. Oui sur un serveur partagé

C. Non

3. Pourquoi les permissions du dossier comptent-elles ?

A. Elles changent le propriétaire SQL

B. Elles interviennent dans l’accès au chemin

C. Elles remplacent TLS

4. Une ACL peut…

A. Ajouter des règles à examiner

B. Prouver automatiquement une attaque

C. Remplacer toutes les versions OS

5. Que garder avant une modification autorisée ?

A. Un secret réel publié

B. Un balayage Internet

C. Les valeurs initiales et une restauration

MODULE 09 / BASES LINUX

Linux : services, sockets et frontière réseau

À l'issue du module : Observer une socket ; Adapter un diagnostic à une distribution ; Délimiter la surface d'exposition

Un système Linux peut utiliser systemd pour gérer ses services, mais tous les environnements Linux ne disposent pas de systemd. Un conteneur, une distribution différente ou un environnement spécialisé peut produire des commandes différentes. Avant d'écrire « service absent », vérifiez le système d'initialisation et le contexte. Le serveur Python du cours reste un processus lancé manuellement dans votre terminal.

ss affiche des sockets et peut montrer l'état d'écoute TCP. Le filtre sur le port 8876 limite l'observation au TP. Selon les droits, l'information sur le processus propriétaire peut être partielle. Cette absence d'information n'est pas une preuve que le processus est caché. Le rapport mentionne ce qui est visible avec votre compte et ce qui exigerait un autre accès autorisé.

Un service peut écouter sur 127.0.0.1, une adresse d'interface précise ou toutes les interfaces. L'adresse d'écoute et les règles de filtrage déterminent ensemble la surface d'exposition. Un pare-feu ne transforme pas une application incorrecte en application sûre. Inversement, une écoute locale documentée ne devient pas une vulnérabilité simplement parce qu'un outil montre un port ouvert.

Sur Ubuntu, ufw est une interface courante de configuration du filtrage. Lire une règle n'est pas la même chose que l'appliquer. Un changement peut vous couper un accès distant ou modifier le comportement d'une VM. Le niveau débutant travaille en lecture seule sur les protections du poste principal et propose les modifications dans un plan. Les essais d'application d'une règle restent dans une VM isolée avec procédure de restauration.

Une validation simple consiste à démarrer votre service, lire son adresse d'écoute, interroger /api/health puis arrêter votre processus et relire l'état. Il n'est pas nécessaire de scanner tous les ports d'un sous-réseau. Vous apprenez à établir une correspondance entre la configuration prévue et le résultat observé. Si un autre service occupe le port, vous identifiez la situation et changez d'environnement au lieu de tuer un processus inconnu.

Comment documenter un essai

Conservez la version de l'OS, la commande exacte, le fichier ou processus de TP concerné, la sortie observée et sa limite. Les illustrations de commande ne remplacent pas une exécution native. Les manipulations d'administration restent dans une machine de laboratoire, avec restauration prévue ; les protections de la machine principale restent actives.

Relier ss à votre processus local

ss -ltnp affiche des écoutes TCP et, lorsque les droits le permettent, des informations de processus. Cherchez la ligne du port 8876 et distinguez 127.0.0.1 d'une écoute sur toutes les interfaces. Une colonne de processus vide peut provenir des droits de lecture. Elle ne justifie pas automatiquement sudo. Fermez seulement votre serveur de TP et comparez la sortie. systemctl est pertinent sur les environnements qui utilisent systemd ; sa présence ne peut pas être supposée sur toute distribution, tout conteneur ou tout environnement Linux.

Vérification formative : expliquez à un pair le sens de chaque paramètre, le périmètre de la commande et une conclusion qu'elle ne permet pas. Si vous ne pouvez pas l'expliquer, relisez le support avant de l'exécuter. Livrable supplémentaire : trois colonnes commande, observation, limite. Une observation non exécutée reste marquée comme telle ; la documentation du fournisseur sert alors de référence et ne devient pas une preuve d'exécution.

systemd, processus et sockets

ss décrit des sockets ; ps décrit des processus. systemctl concerne les unités gérées par systemd, lorsqu'il est présent. Une distribution utilisant un autre système d'initialisation, un conteneur minimal ou un environnement WSL peut ne pas offrir les mêmes observations. Un programme lancé dans votre terminal n'est pas automatiquement une unité systemd.

```
ss -ltnp 'sport = :8876'
```

Les options demandent les sockets TCP en écoute, une présentation numérique et des informations de processus disponibles. La ligne attendue concerne 127.0.0.1:8876 ; son absence mérite une vérification du lancement. Le nom du processus peut ne pas être visible dans certains contextes de droits. N'étendez pas l'observation au réseau de l'entreprise pour compléter une sortie. Relevez avant et après l'arrêt de votre serveur.

Exercice : expliquez pourquoi une écoute locale n'a pas besoin d'une règle autorisant tous les clients réseau. Corrigé : le service ne se lie pas à une interface extérieure ; ouvrir une règle large ne répond pas à un besoin de ce TP. L'examen d'un service distant exige un autre périmètre et une architecture de laboratoire dédiée.

Méthode pas à pas

Pendant le laboratoire : ss -ltnp ; repérez seulement la ligne de 127.0.0.1:8876. Interrogez la route health avec client_tp.py. Si votre VM utilise systemd, systemctl --type=service --state=running permet une lecture des services ; limitez le rapport aux éléments pertinents au TP. Arrêtez le serveur Python avec Ctrl+C et répétez l'observation. Une commande nécessitant un droit absent est notée, pas contournée.

Votre exercice

Associez adresse, port, processus et commande de lancement du laboratoire. Expliquez pourquoi une absence de nom de processus dans une sortie ss n'est pas une preuve de dissimulation.

Cas entièrement fictif

Cas fictif : une VM pédagogique utilise un environnement sans systemd. L'apprenant applique une procédure systemctl et conclut que tous les services sont en panne.

Corrigé raisonné

La fiche d'environnement identifie le contexte et le système d'initialisation. Le port du laboratoire est observé avec ss et la requête locale confirme sa fonction. Le résultat est limité à cette écoute. Aucune règle de pare-feu ou configuration SSH du poste principal n'est changée. La restauration reste prévue avant un essai d'administration en VM.

MODULE 09 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. ss -ltnp observe notamment…

A. Des écoutes TCP et des processus lorsque lisibles

B. Des mots de passe

C. Les clauses du mandat

2. Une colonne de processus vide implique…

A. Une attaque certaine

B. Une obligation universelle de sudo

C. Une limite de lecture à examiner

3. systemctl est pertinent…

A. Uniquement sous macOS

B. Sur les environnements utilisant systemd

C. Sur absolument tout Linux

4. Quelle écoute correspond au laboratoire de référence ?

A. 127.0.0.1:8876

B. Toutes les interfaces Internet

C. Un serveur distant inconnu

5. Quel service arrêter pour le contre-test ?

A. Tous les services du système

B. Le pare-feu de l’organisation

C. Uniquement votre serveur de TP identifié

MODULE 10 / BASES LINUX

Linux : journaux, horodatage et analyse limitée

À l'issue du module : Filtrer sans perdre le contexte ; Comparer des événements datés ; Décrire une limite de couverture

Les journaux peuvent provenir d'une application, de journald ou de fichiers gérés par un service. Ils ne suivent pas tous le même format. journalctl est pertinent lorsque le système utilise le journal de systemd ; il n'est pas un lecteur universel de tous les environnements Linux. La rotation et la rétention peuvent faire disparaître des événements anciens sans indiquer une suppression malveillante.

Pour apprendre l'analyse, utilisez événements-fictifs.json fourni avec le cours. Le fichier décrit uniquement les demandes d'Alice et de Bob sur le laboratoire. Python permet de sélectionner un identifiant de test et de classer les événements sans dépendre d'un logiciel supplémentaire. Une commande de filtre ne prouve pas que le journal source est complet : elle montre les éléments conservés qui correspondent au filtre.

L'horodatage porte un fuseau. Un événement UTC et un affichage local peuvent sembler désynchronisés. La chronologie du TP garde la valeur UTC d'origine, le nom du fichier et la version du laboratoire. Elle distingue demande, réponse et application du correctif. Un événement de demande n'établit pas que l'opération a été autorisée ou terminée ; il faut une observation du résultat.

Les journaux d'un vrai poste peuvent contenir des noms de compte, des chemins et des adresses. On ne les copie pas dans un devoir public. Même sur une VM de TP, on collecte les seules lignes nécessaires au mandat. La disponibilité de sudo ne rend pas légitime une collecte de toutes les traces. Le dossier de preuves indique la source, la méthode de filtrage et ses limites.

Au projet final, votre chronologie doit faire apparaître le contre-test et le succès attendu du propriétaire. Si vous ne gardez que le refus après correction, vous pourriez masquer un service arrêté. L'analyse compare donc deux opérations opposées sur des ressources identifiées. Elle montre à quel moment les données fictives ont été utilisées et ne décrit pas un incident d'entreprise qui n'a jamais eu lieu.

Comment documenter un essai

Conservez la version de l'OS, la commande exacte, le fichier ou processus de TP concerné, la sortie observée et sa limite. Les illustrations de commande ne remplacent pas une exécution native. Les manipulations d'administration restent dans une machine de laboratoire, avec restauration prévue ; les protections de la machine principale restent actives.

Interpréter une absence de trace Linux

Le journal système dépend de la distribution et des services utilisés. journalctl suppose un environnement qui lui fournit les données appropriées. Le jeu événements-fictifs.json reste portable et permet d'apprendre le filtrage sans lire des journaux privés. Gardez l'heure UTC du jeu et indiquez le fuseau de votre propre capture. Dans une VM, une absence de ligne peut venir d'un intervalle, d'une rotation ou d'une source différente. Décrivez ce que vous avez effectivement lu, puis proposez une vérification adaptée plutôt que d'affirmer qu'aucun accès n'a eu lieu.

Vérification formative : expliquez à un pair le sens de chaque paramètre, le périmètre de la commande et une conclusion qu'elle ne permet pas. Si vous ne pouvez pas l'expliquer, relisez le support avant de l'exécuter. Livrable supplémentaire : trois colonnes commande, observation, limite. Une observation non exécutée reste

marquée comme telle ; la documentation du fournisseur sert alors de référence et ne devient pas une preuve d’exécution.

Sources, fuseaux et conservation

Sur un système utilisant systemd, journalctl peut consulter le journal selon les droits et sa configuration. D’autres distributions utilisent des fichiers et des mécanismes différents. La rétention peut être volatile ou persistante ; un redémarrage et la rotation peuvent réduire l’historique. Un fichier vide ne démontre pas l’absence d’activité.

Pour le TP, l’analyse obligatoire porte sur le jeu JSON fictif distribué. Une exploration native du journal reste facultative et limitée à votre VM ou à vos événements propres. Documentez la commande de filtre, l’intervalle, le fuseau et la source. Ne copiez pas tout /var/log dans le rapport : cette collecte serait disproportionnée et pourrait inclure des informations privées.

Corrigé de méthode : deux lignes avec la même seconde ne suffisent pas à établir une causalité. Identifiez l’action, la source, un identifiant de corrélation s’il existe et les limites d’horloge. Une chronologie doit permettre de vérifier une hypothèse précise, pas de raconter une compromission imaginaire.

Méthode pas à pas

Exécutez analyse_journal.py sur le fichier fictif. Sélectionnez AUTH-02 et comparez les modes vulnérable et corrigé. Relevez les heures UTC et les statuts. Dans une VM Linux prévue pour le TP, journalctl --since peut être étudié sur une période courte et un service autorisé ; l’accès absent à un journal se note comme limite. Aucun journal personnel de la machine hôte n’est demandé.

Votre exercice

Rédigez une chronologie de six lignes et une note de couverture. Identifiez la différence entre une requête reçue, un accès autorisé et un refus de permission.

Cas entièrement fictif

Cas fictif : un journal tournant commence après l’événement supposé. Le responsable veut conclure que l’événement n’a pas eu lieu parce qu’aucune ligne n’apparaît.

Corrigé raisonné

L’absence d’événement ne prouve pas l’absence d’action quand la rétention, le filtre ou la couverture sont incomplets. La note indique la période disponible. La chronologie pédagogique ne porte que sur les données fictives, conserve UTC et associe chaque statut au document demandé. Le rapport n’emploie pas le mot compromission pour une simple erreur de laboratoire.

MODULE 10 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. journalctl suppose…

A. Un environnement qui lui fournit les données appropriées

B. Un accès universel à tout journal

C. Une reconnaissance universitaire

2. Le fuseau du jeu fictif est…

A. Toujours l’heure locale du lecteur

B. Une donnée secrète

C. UTC

3. Une absence de ligne peut provenir…

A. Du seul score QCM

B. D’un filtre, d’une rotation ou d’une autre source

C. D’une absence certaine de tentative

4. Quel export ne convient pas au devoir ?

A. Un journal privé sans rapport avec le TP

B. La chronologie du jeu fictif

C. Les seules lignes autorisées du laboratoire

5. Le rapport doit indiquer…

A. Une exécution inventée

B. Une preuve empruntée à un autre OS

C. Ce qui a réellement été lu et sa limite

MODULE 11 / BASES LINUX

Linux : correctifs, confinement et restauration

À l'issue du module : Lire une politique de confinement ; Vérifier un correctif de distribution ; Construire un retour arrière

Le durcissement d'un système Linux commence par sa distribution, sa version et ses dépôts autorisés. Une procédure Ubuntu ne s'applique pas automatiquement à toutes les distributions. Les mises à jour de sécurité doivent être lues avec les avis du mainteneur et les versions des paquets. Une version amont différente n'est pas toujours une absence de correction, notamment lorsque la distribution rétroporte un correctif.

Les permissions classiques décrivent les accès discrétionnaires. AppArmor et SELinux sont des exemples de mécanismes pouvant ajouter un contrôle obligatoire selon l'environnement. Sur Ubuntu, AppArmor peut limiter les capacités d'un programme selon un profil. Lire son état n'est pas désactiver son application. Le cours ne demande pas de retirer un profil de protection pour faire fonctionner un outil de TP.

Un compte quotidien ne doit pas recevoir des droits élevés sans besoin. Une délégation sudo doit être comprise et limitée. Le niveau débutant examine une configuration fictive et propose une séparation des rôles. Il n'apprend pas à rechercher une faille locale pour devenir administrateur d'un système tiers. Les éventuelles modifications de configuration se font dans une VM dont vous possédez le mandat et l'état initial.

Le chiffrement d'un support et une sauvegarde répondent à des besoins distincts. LUKS est un mécanisme courant de chiffrement de volumes sous Linux, mais sa disponibilité et sa configuration dépendent de l'installation. Vous ne changez pas le chiffrement du disque hôte dans ce parcours. Une clé, une phrase de récupération ou un fichier d'ouverture ne doit pas figurer dans un rapport illustré.

La restauration du TP porte sur un dossier et un fichier fictifs. Comparez les empreintes, vérifiez la présence des fichiers et relancez un test fonctionnel. Copier un dossier ne restaure pas nécessairement un système complet, ses permissions, sa base ou sa configuration de démarrage. Le plan de durcissement indique donc les composants couverts, ceux qui ne le sont pas et la procédure de vérification après un changement.

Comment documenter un essai

Conservez la version de l'OS, la commande exacte, le fichier ou processus de TP concerné, la sortie observée et sa limite. Les illustrations de commande ne remplacent pas une exécution native. Les manipulations d'administration restent dans une machine de laboratoire, avec restauration prévue ; les protections de la machine principale restent actives.

Adapter la mesure à la distribution

Le moindre privilège limite les capacités au besoin identifié. Les permissions de fichiers, le pare-feu et un mécanisme comme AppArmor ne se remplacent pas mutuellement. AppArmor est une référence pertinente pour Ubuntu, sans être une caractéristique universelle de tous les Linux. Pour votre dossier, notez distribution, version, service concerné, règle proposée et méthode de contrôle. Une modification de règle dans une VM exige un état de référence et une procédure de restauration. Aucun désarmement global du pare-feu n'est nécessaire pour notre laboratoire local.

Vérification formative : expliquez à un pair le sens de chaque paramètre, le périmètre de la commande et une conclusion qu'elle ne permet pas. Si vous ne pouvez pas l'expliquer, relisez le support avant de l'exécuter. Livrable supplémentaire : trois colonnes commande, observation, limite. Une observation non exécutée reste

marquée comme telle ; la documentation du fournisseur sert alors de référence et ne devient pas une preuve d’exécution.

Trois couches de protection à ne pas confondre

Les permissions classiques limitent des usages selon les identités. Un mécanisme tel qu’AppArmor ajoute des contraintes aux programmes selon les profils disponibles. Le pare-feu filtre les échanges selon ses règles. Aucune couche ne corrige automatiquement la logique d’autorisation du laboratoire. Les choix de gestionnaire de paquets et de contrôle obligatoire dépendent de la distribution.

Dans votre plan, indiquez la distribution, les sources de paquets et la disponibilité des mises à jour. Ne désactivez pas AppArmor ou SELinux pour supprimer un message sans diagnostic. Une mesure de sécurité qui bloque une application doit être comprise dans une VM de TP, puis adaptée au besoin minimal. La restauration doit être testée sur une copie de fichier avant d’envisager une intervention plus large.

Méthode pas à pas

Écrivez une fiche de distribution et une grille de lecture des protections. Consultez l’avis officiel d’un paquet de votre environnement de TP sans appliquer de mise à jour au poste principal. Dans le seul dossier du cours, copiez preuve.txt, modifiez-le puis restaurez-le à partir de votre copie de référence. Comparez sha256sum et testez le laboratoire. Pour AppArmor, une lecture avec aa-status concerne une VM autorisée et peut nécessiter un droit administrateur ; elle reste facultative si ce droit manque.

Votre exercice

Proposez un plan de durcissement Linux comprenant version supportée, comptes, services nécessaires, confinement et restauration. Pour chaque mesure, donnez une preuve de lecture ou inscrivez « non exécuté ».

Cas entièrement fictif

Cas fictif : un script ne fonctionne pas avec un profil AppArmor. Une personne propose de désactiver tous les profils au lieu de vérifier les accès effectivement nécessaires au programme.

Corrigé raisonné

Le plan préserve le confinement et identifie l’accès demandé avant tout changement limité en VM. Les paquets sont interprétés avec les avis de la distribution, pas seulement leur version amont. La restauration est vérifiée sur le fichier fictif et n’est pas annoncée comme une sauvegarde complète du poste. Les secrets de chiffrement restent hors du support.

MODULE 11 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Le moindre privilège consiste à…

A. Limiter les capacités au besoin identifié

B. Donner tous les droits pour éviter les erreurs

C. Désactiver toutes les ACL

2. AppArmor est-il universel à tous les Linux ?

A. Oui

B. Oui dès que Python est installé

C. Non

3. Avant une règle dans une VM, que préparer ?

A. Une désactivation permanente

B. Un état de référence et une restauration

C. Un déploiement public automatique

4. Que doit préciser le plan ?

A. Distribution, version, règle et contrôle

B. Seulement le nom Linux

C. Le mot de passe administrateur

5. Le laboratoire exige-t-il de désarmer le pare-feu ?

A. Oui toujours

B. Oui pour obtenir le certificat

C. Non

MODULE 12 / BASES LINUX

Projet final : audit limité d’Atelier Aurore

À l’issue du module : Conduire un audit local de bout en bout ; Produire un dossier reproductible ; Autoévaluer sans surattester

Atelier Aurore est une entreprise entièrement fictive. Son application pédagogique gère deux documents sans données personnelles. Vous disposez d’un mandat limité au serveur local du cours, au fichier de preuve et au jeu de journaux fourni. Le projet évalue votre capacité à observer, expliquer, corriger et vérifier. Il ne vous demande pas de rechercher une victime ni de démontrer une intrusion sur un service tiers.

Commencez par le mandat et l’inventaire. Démarrez une version connue du laboratoire et relevez l’environnement. Votre plan contient la matrice d’accès, la vérification du service local, le test des entrées de recherche, l’empreinte d’un fichier et la chronologie. Pour chaque action, indiquez la question qu’elle permet de trancher. Le volume de commandes n’est pas un critère de qualité.

Décrivez au moins deux constats confirmés dans les exemples pédagogiques et un faux positif à écarter. Le défaut d’autorisation et la construction SQL concaténée sont les cas d’étude. L’écoute sur la boucle locale prévue ne doit pas être présentée comme une ouverture Internet. Le fichier modifié volontairement ne doit pas être annoncé comme une preuve de compromission. Votre rapport doit montrer que vous savez limiter une conclusion.

Proposez et vérifiez les corrections. Pour l’autorisation, comparez les routes explicites de démonstration et de correction. Pour SQL, adaptez votre copie du script en conservant les recherches légitimes. Pour le fichier de TP, appliquez la procédure de restauration de votre OS. Ajoutez les essais de non-régression : le propriétaire consulte son propre document et un nom contenant une apostrophe reste une donnée de recherche acceptable.

Le projet est autoévalué avec une grille transparente ; le site ne dispose pas d’une correction humaine de ces livrables. Le certificat du parcours atteste une réussite au QCM final de 40 questions après les confirmations de leçons. Il ne valide pas formellement la qualité du rapport ni l’identité de l’auteur. Le dossier reste néanmoins une preuve personnelle utile de travail réalisé, à présenter honnêtement avec son caractère pédagogique et ses limites.

Consigne de soutenance et critères de maîtrise

Votre dossier final doit contenir une page de synthèse, le mandat, l’environnement, l’inventaire, la matrice HTTP, la comparaison SQL, une observation propre à l’OS, une chronologie et les corrections. Numérotez les preuves et reliez chacune à une conclusion. Présentez aussi une hypothèse que vous avez écartée. Le projet se déroule dans le laboratoire distribué : aucune cible externe n’est nécessaire pour produire un travail solide.

Préparez une explication orale de cinq minutes, seul ou avec un pair volontaire. Répondez à quatre questions : qu’avez-vous réellement observé, quel mécanisme l’explique, comment avez-vous corrigé, et qu’est-ce qui reste inconnu ? Enregistrez éventuellement cette explication pour vous relire, sans données réelles. Cet exercice formatif n’est pas envoyé automatiquement à la plateforme et n’est pas évalué par un jury.

Une copie du projet est satisfaisante lorsque les résultats sont reproductibles, les limites assumées et les corrections contre-testées. Le nombre de commandes lancées n’est pas un critère de qualité. Le niveau intermédiaire pourra approfondir des environnements multi-machines, des permissions complexes et des applications de laboratoire dédiées. Le niveau avancé nécessitera une méthode de mission et un encadrement plus approfondis ; ces étapes ne sont pas incluses dans le certificat de fondamentaux.

Cahier des charges de votre dossier final

Votre dossier contient le mandat limité, la fiche d'environnement, un inventaire du laboratoire, une matrice d'accès, la comparaison SQL, la comparaison d'empreintes, une chronologie et le rapport. Deux constats doivent être expliqués à partir des essais fournis ; ajoutez une alerte écartée avec sa justification. Les captures viennent de votre laboratoire. Les captures du manuel sont des références de présentation et ne doivent pas être présentées comme vos propres essais.

Le livrable inclut un README expliquant comment préparer le dossier et relancer les exercices, sans secret. Le destinataire doit pouvoir distinguer les données fictives initiales, les résultats réellement produits et les notes d'analyse. N'archivez pas une copie de votre profil navigateur ni des journaux personnels. La grille de projet reste une autoévaluation distincte des QCM de certification.

Critères de passage vers le niveau intermédiaire

Vous êtes prêt à poursuivre lorsque vous pouvez expliquer une requête HTTP, justifier un périmètre, retrouver une écoute locale, lire une permission, distinguer hachage et chiffrement, confirmer un contrôle d'accès et rédiger un contre-test. Réussir un QCM aide à vérifier les connaissances ; refaire un exercice sans suivre chaque ligne du manuel vérifie une autre compétence. Conservez les deux résultats dans votre portfolio privé.

Pour préparer la suite, choisissez un exercice et changez une seule variable prévue, par exemple le préfixe du calcul local ou le contenu d'une copie de preuve. Prédisez le résultat, exécutez, puis expliquez l'écart. Cette démarche constitue une base pour les futures missions plus complexes ; elle ne vous attribue aucun niveau professionnel ni autorisation supplémentaire.

Méthode pas à pas

Consacrez environ douze heures au projet et quatre heures à la révision et au QCM. Produisez mandat, inventaire, plan, résultats, captures réelles, rapport de constats, corrections et contre-tests. Pour la version adaptée à votre OS, ajoutez une lecture des droits, une observation de l'écoute et une preuve de restauration. Remplissez la grille de projet : chaque critère exige une référence à un fichier ou une capture. Une opération non réalisée reste marquée comme telle.

Votre exercice

Remettez un dossier de huit pièces suivant le modèle fourni, puis réalisez votre autoévaluation. Révisez les questions formatives avant le QCM. Ne demandez pas de certificat avec une identité fictive pour tester le site public.

Cas entièrement fictif

Cas fictif : une apprenante copie la sortie attendue de l'atelier sans exécuter son essai. Son rapport serait plus complet en apparence, mais ne peut pas soutenir la conclusion annoncée.

Corrigé raisonné

La grille distingue support d'exemple et preuve personnelle. Les constats citent des fichiers de résultat réellement produits, le mandat reste local et les limites sont visibles. Les corrections conservent les usages légitimes. Un dossier avec une étape non exécutée doit l'indiquer plutôt que fabriquer un résultat. Le QCM et le projet ont des fonctions différentes ; seule la réussite au questionnaire selon le circuit du site ouvre le certificat interne.

MODULE 12 / AUTOÉVALUATION

QCM du chapitre

Entraînement avant le QCM enregistré sur le site. Cinq questions, réussite à quatre réponses correctes. La correction officielle est affichée après votre réponse en ligne.

1. Quelle cible faut-il pour le projet final ?

A. Le laboratoire distribué

B. Une entreprise choisie sans accord

C. Un site public aléatoire

2. Quel élément complète les défauts confirmés ?

A. Un faux témoignage client

B. Une capture d’attaque inventée

C. Une hypothèse écartée et sa justification

3. Quel travail est attendu après correction ?

A. Un changement de cible

B. Un contre-test et les accès légitimes

C. La suppression de toutes les preuves

4. Une soutenance formative signifie-t-elle un jury universitaire ?

A. Non

B. Oui systématiquement

C. Oui si elle dure cinq minutes

5. Que prouve le diplôme interne obtenu ici ?

A. Une qualification professionnelle universelle

B. Une reconnaissance universitaire automatique

C. La réussite au parcours et au QCM non surveillé selon les règles VLR

Travaux pratiques et dossier final

Laboratoire des trois formations

Ces fichiers créent une application de démonstration locale. Ils ne constituent pas un outil de test d'un système tiers ni une application à déployer publiquement. Toutes les identités, documents et journaux sont fictifs.

Lancement

Dans le dossier `laboratoire`, avec Python 3 installé depuis la source officielle :

Windows : `py -3 laboratoire.py` (ou `python laboratoire.py` selon l'installation).

Linux et macOS : `python3 laboratoire.py`.

Ouvrir `http://127.0.0.1:8876`. Conserver l'adresse de boucle locale. Aucun droit administrateur n'est requis. Si le port est occupé, identifier d'abord l'ancien processus du laboratoire. Ne pas arrêter un service inconnu. Arrêt : Ctrl+C dans la fenêtre de lancement.

TP 1. Mandat et environnement (2 h)

Remplir `mandat.md` et relever OS, architecture, version Python, version du cours et dossier de travail. Identifier les routes et fichiers autorisés. Ouvrir la page locale, cliquer « Vérifier le laboratoire » et capturer le résultat. Fermer le serveur, constater l'échec d'une nouvelle demande et le relancer. Livrable : fiche de version et preuve avant/après arrêt. Corrigé : le lancement et l'arrêt sont documentés, sans exposition sur une interface réseau extérieure.

TP 2. Réseau et HTTP (2 h)

Ouvrir les outils de développement du navigateur, onglet Réseau. Observer seulement les routes du TP. Comparer page HTML et `/api/health`. Relever adresse, port, méthode, statut et type de contenu. Distinguer le temps d'ouverture du navigateur du temps de réponse de l'application. Livrable : une requête annotée et le trajet client/service/réponse. Corrigé : un statut health 200 ne valide pas les contrôles d'accès.

TP 3. Contrôle d'accès (3 h)

Alice possède 101, Bob possède 102. Sélectionner Alice et le document 102. Le contre-exemple renvoie HTTP 200 et le document de Bob : c'est le défaut fabriqué pour le TP. La route corrigée renvoie HTTP 403. Tester ensuite Alice vers 101 et Bob vers 102 : le statut attendu est 200. Tester visiteur vers 102 : 401. `client_tp.py` exécute ces quatre cas sur la seule boucle locale. Livrable : matrice attendue/observée et deux captures avant/après. Corrigé : le serveur compare la permission au document ; un identifiant aléatoire ne serait pas une correction suffisante.

L'identité `X-Lab-User` est une facilité pédagogique contrôlée par l'élève, pas une véritable session. Le serveur illustre l'autorisation à partir d'une identité supposée connue. Il ne doit pas être utilisé comme exemple complet d'authentification.

TP 4. Entrées et SQL en mémoire (3 h)

Exécuter `tp_sql.py`. Lire la fonction qui concatène une chaîne dans SQL, puis celle qui transmet un paramètre. Les produits sont fictifs et la base disparaît à la fin. Avec `Lampe`, les deux recherches donnent le produit attendu. Avec `Bureau d'étude`, la construction fragile provoque une erreur et la recherche paramétrée retrouve le produit. Le contre-exemple SQL prévu renvoie plusieurs produits avec la concaténation, mais aucun avec le paramètre. Modifier une copie du script pour n'utiliser que la requête paramétrée. Livrable : tableau de trois entrées et correction argumentée. Corrigé : la valeur est traitée comme une donnée et les recherches légitimes restent possibles.

TP 5. Empreinte et droits (3 h)

Examiner seulement `preuve.txt`. Windows : `Get-FileHash -LiteralPath .\preuve.txt -Algorithm SHA256` et `Get-Acl -LiteralPath .\preuve.txt`. Linux : `sha256sum preuve.txt` et `ls -l preuve.txt`. macOS : `shasum -a 256 preuve.txt` et `ls -le preuve.txt`. Créer une copie, modifier cette copie puis comparer les empreintes. Livrable : chemin, méthode, heure, droits et empreintes des deux états. Corrigé : une différence montre des octets différents ; elle ne prouve pas une attaque. Une modification de droits reste limitée à une copie du TP ou à une VM autorisée.

TP 6. Écoute et processus (3 h)

Observer le port 8876 avec la commande propre à l'OS et sa documentation. Windows : `Get-NetTCPConnection -LocalPort 8876 -State Listen`. Linux : `ss -ltnp`, en ne conservant dans le rapport que la ligne du TP. macOS : `lsof -nP -iTCP:8876 -sTCP:LISTEN`. Lancer health, arrêter le serveur et constater la fermeture. Livrable : tableau processus/adresse/port avant/après. Corrigé : l'écoute locale est prévue, elle n'est pas une ouverture Internet. Une visibilité partielle du processus reste une limite de lecture.

TP 7. Chronologie fictive (3 h)

Exécuter `analyse_journal.py`. Le jeu fourni est fabriqué ; le fichier `resultats-locaux.jsonl`, si présent, contient vos véritables essais sur ces données fictives. Les deux sources ne sont pas interchangeables. Filtrer AUTH-02 et construire une chronologie UTC. Livrable : six lignes expliquées et une note sur la couverture. Corrigé : refus après correction et accès légitime doivent être distingués. Une absence de ligne ne prouve pas l'absence de tentative.

TP 8. Restauration et durcissement (3 h)

Sauvegarder le seul fichier de TP, le modifier puis restaurer la référence et comparer son empreinte. Rédiger ensuite un plan de protections propre à l'OS, sans les changer sur la machine principale. Livrable : preuve de restauration et grille mesure/bénéfice/limite/contrôle. Corrigé : chiffrement du disque, permission de fichier et contrôle d'accès applicatif répondent à des risques différents. Aucun secret de récupération n'est affiché.

Projet final (12 h) et préparation au QCM (4 h)

Assembler mandat, inventaire, plan de tests, captures réelles, deux constats, un faux positif, corrections, contre-tests et chronologie dans `rapport-modele.md`. Utiliser `grille-projet.md`. Le projet est autoévalué ; le QCM final de 40 questions conditionne le certificat VLR au seuil de 32/40. Le site ne prétend pas avoir corrigé humainement les livrables.

Autoévaluation du projet (100 points indicatifs)

Cette grille sert à relire son dossier. Elle ne constitue pas une correction humaine et ne calcule pas le résultat du certificat.

Critère	Points	Preuve demandée
Mandat et conditions d'arrêt	10	Mandat local rempli
Environnement et versions	10	Fiche OS/Python/cours
Inventaire et hypothèses testables	10	Tableau et plan
Autorisation : accès incorrect, refus et accès légitime	15	Matrice et captures réelles
SQL : contre-exemple et paramètres	15	Résultats et correction locale
Observation propre à l'OS	10	Droits et écoute, ou limite explicitement déclarée
Faux positif écarté	10	Observation et justification
Correction, restauration et contre-test	10	Fichiers et empreintes
Rapport et confidentialité des preuves	10	Rapport lisible, aucune donnée réelle

Pour chaque ligne, attribuer 0 si l'élément manque, la moitié si l'élément est présent mais sa preuve ou sa justification est insuffisante, la totalité si l'élément est expliqué et reproductible. Si une opération sort du mandat, interrompre et reprendre dans le bon cadre avant de considérer le dossier comme satisfaisant.

Objectif formatif recommandé : 80/100 et correction des points incomplets. Le certificat du site reste attribué uniquement par le QCM final à 32/40 après progression confirmée, sans confusion avec cette note personnelle.

Rapport d’audit pédagogique : Atelier Aurore

Caractère fictif du cas, version du cours et OS :

Résumé pour le responsable

Question étudiée, conséquence démontrée, correction proposée et limites de couverture :

Mandat et environnement

Actifs, versions, horaires, comptes fictifs et opérations réellement exécutées :

Inventaire et plan

Actif	Propriétaire fictif	Fonction	Question	Méthode autorisée

Constat HE-01

Titre précis :

Actif et version :

Condition et permission :

Résultat attendu :

Résultat observé :

Preuve : chemin du fichier ou capture, date, caractère fictif des données.

Conséquence limitée au scénario :

Cause :

Action de correction et responsable fictif :

Priorité justifiée, sans score inventé :

Contre-test interdit et légitime :

Limites :

Constat HE-02

Même structure, avec un défaut différent et sa preuve.

Faux positif écarté

Observation, hypothèse initiale, vérification et conclusion limitée :

Chronologie

Date UTC	Source	Identité fictive	Objet	Mode/version	Résultat

Retour à l’état initial

Fichiers restaurés, empreintes et contrôle de fonctionnement :

Couverture et autoévaluation

Opérations non exécutées ; différences propres à l’OS ; critères du projet satisfaits et preuves correspondantes :

Révision et préparation de la suite

Portfolio de passage

Conservez un mandat, une fiche de versions, une matrice d’accès correctement expliquée, un résultat SQL corrigé, une preuve d’intégrité, une chronologie et un rapport avec contre-tests. Pour chaque document, notez ce qui a été exécuté, ce qui a seulement été étudié et ce qui reste à vérifier. Un portfolio soigné montre votre raisonnement ; l’accumulation de captures sans analyse ne suffit pas.

Quand un résultat vous surprend

Situation	Premier contrôle	Ce qu’il faut éviter
Commande absente	OS, version et installation officielle	Installer un outil d’une source inconnue
Accès refusé	Chemin exact et droits nécessaires	Élever automatiquement les privilèges
Port occupé	Identifier votre propre programme	Arrêter un service inconnu
Réponse inattendue	Identité, route et résultat attendu	Conclure à une attaque sans preuve
Fichier différent	Copie, octets et référence d’origine	Déduire une intention de l’empreinte
Journal incomplet	Source, période et droits de lecture	Déduire une absence d’activité

Reconnaître un acquis

Vous maîtrisez une notion lorsque vous pouvez l’expliquer avec un exemple, exécuter son exercice dans le périmètre prévu et décrire une limite. Pour le réseau, distinguez une erreur de connexion d’un refus applicatif. Pour les permissions, reliez un droit à une identité et à une ressource. Pour une correction, vérifiez à la fois le cas refusé et l’usage légitime conservé.

Autoévaluation avant le QCM final

Expliquez à voix haute six points : pourquoi le mandat précède l’outil ; pourquoi un 200 ne valide pas l’autorisation ; pourquoi une apostrophe est une donnée légitime ; pourquoi le hachage ne chiffre pas ; pourquoi un score de sévérité ne suffit pas au risque métier ; pourquoi un contre-test doit préserver l’usage normal. Comparez vos réponses aux modules concernés, puis refaites les exercices qui restent difficiles.

La certification sur le site est une évaluation de connaissances au seuil indiqué. Le projet est autoévalué séparément. Ne présentez pas une réussite au QCM comme une vérification supervisée de vos compétences pratiques. Pour la suite, privilégiez la capacité à expliquer et reproduire un résultat plutôt qu’un nombre d’outils mémorisés.

Glossaire des bases

ACL

Liste de contrôle d’accès ; elle peut compléter les droits classiques.

Actif

Élément identifié utile à une activité et auquel on associe un propriétaire ou un usage.

Authentification

Établissement d’une identité selon les garanties du mécanisme.

Autorisation

Décision d’accès d’une identité à une action sur une ressource.

Boucle locale

Communication logique avec la machine elle-même ; ici 127.0.0.1.

CIDR

Notation adresse/préfixe décrivant une plage par ses bits communs.

Client

Programme qui initie une demande à un service.

Contre-test

Vérification de la correction et des usages légitimes conservés.

Cookie

Donnée conservée par le navigateur et transmise selon ses règles.

CSRF

Action indue tirant parti du contexte authentifié d’un navigateur.

CVE

Identifiant d’une vulnérabilité publiée, dont l’applicabilité doit être vérifiée.

CVSS

Cadre de sévérité technique ; sa version et son vecteur doivent être précisés.

CWE

Classification de familles de faiblesse logicielle.

DNS

Système de noms fournissant notamment des informations d’adressage.

Empreinte

Valeur calculée par hachage pour comparer notamment les octets.

Encodage de sortie

Représentation d’une donnée adaptée au contexte où elle sera insérée.

Faux positif

Alerte non confirmée dans le contexte et avec les conditions examinées.

FileVault

Mécanisme de chiffrement de volume macOS, avec ses conditions de récupération.

Gatekeeper

Protection macOS participant au contrôle des logiciels ouverts.

Hachage

Calcul d’une empreinte ; il ne constitue pas un chiffrement réversible.

HTTP

Protocole d’échange entre clients et serveurs ; son statut nécessite un contexte.

Injection

Donnée interprétée comme structure ou instruction dans un autre contexte.

Instantané

État enregistré d’un environnement ; pas une sauvegarde indépendante à lui seul.

Intégrité

Propriété de cohérence et de protection contre des modifications indues.

Journal

Trace issue d’une source, avec sa couverture, sa rétention et ses limites.

JSON

Format de données structurées, distinct d’un programme exécutable.

Mandat

Autorisation définissant les actifs, méthodes, acteurs et conditions d’arrêt.

Moindre privilège

Attribution des droits nécessaires à la fonction sans extension injustifiée.

NAT

Traduction d’adresses ; ne garantit pas l’absence de communications extérieures.

Paramètre SQL

Valeur transmise séparément de la structure d’une requête.

PID

Identifiant d’un processus dans un environnement donné.

Port

Numéro distinguant un point de communication au sein d’une adresse.

Processus

Instance d’un programme en cours d’exécution.

Risque

Évaluation contextualisée d’événements défavorables et de leurs conséquences.

Rétroportage

Correctif intégré à une version maintenue sans reprendre toute la version amont.

Sel

Valeur propre à un enregistrement de mot de passe, non nécessairement secrète.

Serveur

Programme qui traite des demandes selon son interface et ses règles.

Session

Contexte permettant de rattacher des requêtes à une identité ou à un état reconnu.

Shell

Interpréteur de commandes ; ses règles diffèrent selon l’environnement.

Signature numérique

Mécanisme vérifié par une clé publique, avec un modèle de confiance.

SIP

Protection macOS limitant certaines modifications de zones protégées.

Socket

Point de communication décrit selon une famille et un protocole.

Surface d’exposition

Interfaces accessibles dans des conditions identifiées.

TCP

Protocole fournissant un flux ordonné et des mécanismes de retransmission.

TLS

Protection des échanges avec validation et configuration correctes.

UDP

Transport par datagrammes sans les mêmes garanties de flux que TCP.

UTC

Référence temporelle utile pour comparer des traces, sous réserve des horloges.

Validation

Contrôle du contrat d’une donnée ; ne remplace pas l’autorisation.

VM

Machine virtuelle dont l’isolation dépend de l’architecture et des réglages.

XSS

Défaut permettant l’exécution de contenu indu dans un contexte de navigateur.

Références et provenance

Références primaires consultées le 6 octobre 2026. Textes, scénarios et corrections originaux VLR. Aucun partenariat ou agrément de ces organismes n'est revendiqué. Les réglages dépendent des versions : consultez la documentation de votre environnement. Édition 2 : approfondissement des bases et refonte de présentation. Les captures, vidéos et TP existants sont conservés avec leur provenance. Aucune nouvelle exécution native Windows ou Linux n'est revendiquée. Les niveaux intermédiaire et avancé disposent de leurs programmes et évaluations en ligne.

NIST / NICE Framework, versions actuelles

Cadre de compétences ; aucun agrément ni équivalence revendiqués

https://www.nist.gov/itl/applied-cybersecurity/nice/nice-framework-resource-center/nice-framework-current-versions

OWASP / Web Security Testing Guide, version stable

Familles de vérification web, approche structurée

https://wstg.owasp.org/stable/

Microsoft / Get-NetTCPConnection

Lecture des écoutes et identifiant de processus

https://learn.microsoft.com/en-us/powershell/module/nettcpip/get-nettcpconnection

Microsoft / Get-FileHash

Empreinte de fichiers dans PowerShell

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/get-filehash

Microsoft / Sécurité Windows

Fonctions et conditions de disponibilité Windows

https://learn.microsoft.com/en-us/windows/security/

Microsoft / Windows Firewall

Profils et conservation du pare-feu

https://learn.microsoft.com/windows/security/operating-system-security/network-security/windows-firewall/

Canonical / Sécurité Ubuntu Server

Distinguer distribution, permissions et confinement

https://ubuntu.com/server/docs/how-to/security/

Canonical / AppArmor

Contrôle obligatoire et profils

https://ubuntu.com/server/docs/how-to/security/apparmor/

Canonical / Pare-feu Ubuntu

Netfilter et ufw

https://documentation.ubuntu.com/server/how-to/security/firewalls/index.html

Apple / Gatekeeper et protection à l'exécution

Fonctions de protection macOS

https://support.apple.com/guide/security/gatekeeper-and-runtime-protection-sec5599b66df/web

Apple / Sécurité des plateformes Apple

SIP, chiffrement et modèle de sécurité

https://support.apple.com/guide/security/welcome/web

FIRST / CVSS v4.0 : guide utilisateur

Sévérité de base distincte du risque métier

https://www.first.org/cvss/v4.0/user-guide

Python Software Foundation / Serveur HTTP Python

Serveur pédagogique, non destiné à la production

https://docs.python.org/3/library/http.server.html

Nmap / Nmap : TCP connect scan

Référence technique consultée ; exercices originaux

https://nmap.org/book/man-port-scanning-techniques.html

ZAP / Passive Scan

Référence technique consultée ; exercices originaux

https://www.zaproxy.org/docs/desktop/start/features/pscan/

Wireshark / Capture de trafic

Référence technique consultée ; exercices originaux

https://www.wireshark.org/docs/wsug_html_chunked/ChapterCapture.html

Apple / System Integrity Protection

Référence technique consultée ; exercices originaux

https://support.apple.com/en-us/102149

Apple / FileVault

Référence technique consultée ; exercices originaux

https://support.apple.com/guide/mac-help/protect-data-on-your-mac-with-filevault-mh11785/mac

Python Software Foundation / Calculs IPv4 et IPv6

https://docs.python.org/3/library/ipaddress.html

OWASP / Stockage des mots de passe

https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

MDN / Vue d’ensemble HTTP

https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview

GNU / Structure des permissions

https://man7.org/linux/man-pages/man1/chmod.1.html

Linux man-pages / Lecture des sockets avec ss

https://man7.org/linux/man-pages/man8/ss.8.html

Microsoft / Lecture des ACL

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-acl

Microsoft / Politiques d’exécution PowerShell

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies