FORMATION GRATUITE / NIVEAU AVANCÉ

Hacking éthique Linux

Manuel professionnel et syllabus



Visuel pédagogique original. Aucun reportage ou environnement tiers.

18 chapitres / 100 h de travail indicatives / TP corrigés / 18 QCM / Examen de 40 questions

Unité de formation Mashaah LLC, Delaware. Identité fournie par Bouda. Support original en français, daté du 6 octobre 2026. Mise en page justifiée le 7 octobre 2026. Formation et diplôme interne gratuits.

Formation en ligne : leçons, vidéos, exercices et QCM sur votre compte Vise la Réussite. Ce PDF est un support complémentaire facultatif. Les ateliers se réalisent dans votre laboratoire local autorisé.

Sommaire cliquable

Syllabus et organisation 3
Concevoir une mission avancée fondée sur des invariants 4
Examiner les frontières de confiance et les données de décision 7
Vérifier une politique combinant rôle, organisation et responsabilité 9
Auditer les transitions et empêcher une décision sur un état périmé 14
Démontrer une perte de mise à jour et vérifier l’atomicité 16
Relier une clé de déduplication à une demande précise 20
Contrôler un plan d’archive sans écrire hors du dossier prévu 22
Analyser une liste fermée d’URL et ses limites 26
Vérifier l’intégrité d’un message sans inventer une garantie d’identité 28
Relier une revue avancée aux décisions et aux contre-tests 32
Examiner une chaîne de livraison et la provenance de ses composants 34
Construire une détection argumentée et un exercice de réponse 37
Relier droits, groupes et chemins sur Linux 39
Examiner un processus temporaire et une exposition locale 41
Lire une fenêtre de journaux et conserver la provenance 43
Restaurer un fichier et délimiter l’état à récupérer 45
Vérifier la correction, les effets partiels et le retour arrière 47
Constituer un dossier avancé défendable et préparer l’évaluation 49
Dossier pratique et commandes 52
Lecture du code fourni 59
Corrigés des QCM de chapitre 60
Sources et provenance 64

Syllabus et organisation

Parcours gratuit : 18 chapitres, 100 h de travail indicatives, 18 QCM de cinq questions à réussir à 4/5 avant l'examen de 40 questions, réussi à 32/40. Diplôme de réussite interne VLR, non surveillé, sans reconnaissance universitaire revendiquée. Projet autoévalué. Le laboratoire et les captures ont été exécutés sur macOS ; les commandes natives Windows et Linux sont documentées sans exécution native revendiquée.

Volume de travail indicatif : lectures, TP, révisions et projet ; il est distinct de la durée des vidéos. Le tronc commun est partagé entre les trois OS.

Prérequis recommandés

• Avoir étudié les niveaux de base et intermédiaire sur Linux ou démontrer des connaissances équivalentes : HTTP, autorisation, SQL paramétré, permissions et rapport.

• Disposer de Python 3.12 ou ultérieur, d'un navigateur récent et d'un dossier de TP personnel.

• Travailler exclusivement sur le laboratoire local et les données fictives distribuées.

Compétences visées

• Définir des invariants métier et une architecture vérifiable

• Auditer la séparation des responsabilités et les transitions

• Démontrer concurrence, atomicité et déduplication

• Classer archives et destinations avec leurs limites

• Relier intégrité, chaîne logicielle, revue et détection

• Interpréter permissions, exposition et journaux sur Linux

• Produire une recette réversible et un rapport argumenté

Volume de travail et évaluations

Chapitres 1 à 17 : 1,5 h de lecture, 2,5 h de TP et 1 h de révision par chapitre, soit 85 h. Chapitre 18 : 12 h de projet et 3 h de préparation finale, soit 15 h. Total indicatif : 100 h. Les vidéos ont leur durée propre. Le tronc commun est partagé entre les trois parcours ; le changement d'OS ne représente pas trois travaux complets distincts.

Sur le site, cinq questions concluent chaque chapitre : réussite à quatre bonnes réponses. Les dix-huit QCM validés ouvrent le questionnaire final de 40 questions, cinq domaines de huit questions, réussi à 32/40. Le diplôme interne VLR atteste ce parcours non surveillé. Le projet utilise une grille d'autoévaluation distincte. Aucun jury, ECTS, agrément universitaire ou correction humaine du rapport n'est revendiqué.

Articulation des niveaux

Les bases et le niveau intermédiaire préparent ce parcours avancé. Ici, vous auditez des invariants, la concurrence et la reprise, puis reliez provenance, intégrité et détection. Aucune équivalence professionnelle ou universitaire n'est revendiquée.

Parcours débutant recommandé: https://viselareussite.com/formations/hacking-ethique-bases-linux/

CHAPITRE 01 / 5 h indicatives

Concevoir une mission avancée fondée sur des invariants

Objectifs : Définir les propriétés à préserver ; Construire une couverture mesurable ; Distinguer résultat et assurance

Le niveau avancé ne consiste pas à multiplier les commandes. Vous allez vérifier des propriétés qui doivent rester vraies lorsque plusieurs conditions interagissent : une approbation ne peut pas être donnée par l'auteur, une version périmée ne modifie pas l'état, une demande répétée ne crée pas un second effet et une opération concurrente respecte le budget. Atelier Boréal représente un dossier fictif 301, deux organisations et trois identités distribuées. Les montants expriment des unités de TP, sans transaction financière réelle.

Formulez ces propriétés avant les essais. « Claire peut approuver » est incomplet : il faut préciser son organisation, son rôle, l'auteur du dossier, l'état soumis et la version. « Le serveur répond » ne renseigne pas ces conditions. La mission doit relier un besoin, un invariant, un scénario, un résultat attendu et une preuve. Le rapport avancé explique également pourquoi un scénario choisi ne couvre pas toute une famille de situations.

Définir la profondeur et les exclusions

Annoncez les composants accessibles : contrôles Python, interface locale, scripts de tests, transaction SQLite temporaire et modèles documentaires. Le sélecteur d'identité représente un contexte déjà établi ; il n'est pas une authentification à auditer. Les routes de contre-exemple et corrigées utilisent deux états indépendants afin de comparer leurs comportements. Le test de concurrence déclenche seulement deux fils dans votre propre processus. Les URL sont classées sans requête sortante ; les archives sont inspectées sans extraction.

Cette distinction évite une assurance excessive. Le TP permet de confirmer un défaut de transition ou une perte de mise à jour dans son modèle. Il ne démontre pas un défaut identique dans un service tiers. Le verrou Python concerne un processus ; la transaction SQLite concerne la petite base temporaire fournie. Les architectures distribuées, le chiffrement TLS réel, la gestion d'identités de production et la disponibilité à grande échelle restent hors exécution.

Transformer une exigence en décision vérifiable

Pour chaque invariant, construisez un cas autorisé et un cas qui doit être refusé. L'approbation de Claire sur un dossier soumis de A doit réussir. La même action de Bob doit être refusée, même s'il porte le rôle approbateur, car il appartient à B. L'action d'Alice doit être refusée pour conserver la séparation des responsabilités. Une correction qui interdit toute approbation n'a pas satisfait le besoin.

Attribuez une priorité de test à partir des conséquences démontrables et des frontières concernées. N'inventez pas de perte financière, de volume de clients ou de score de vulnérabilité pour ce dossier fictif. Vous pouvez expliquer qu'une transition indue compromettrait une règle de validation dans un système analogue, tout en annonçant que cette conséquence n'a pas été mesurée dans un déploiement réel. La qualité de la mission tient à cette précision.

TP guidé

Télécharger le laboratoire complet

Lire GUIDE-TP.md avant tout essai. Scripts, guides, modèles et grille sont fournis. L'application écoute uniquement sur 127.0.0.1:8878. Aucune extraction d'archive, requête externe ou identité réelle.

Méthode pas à pas

Remplissez mandat.md. Écrivez six invariants et, pour chacun, un scénario positif, un scénario de refus et un contre-test de la correction. Associez les scénarios aux fonctions can_approve, Workflow.change, race_demo, inspect_archive et verify_demo. Indiquez ce qui est réellement exécuté et ce qui restera documentaire.

Exercice à réaliser

Produisez une matrice de couverture avec douze essais minimum, puis justifiez deux limites d’assurance et deux conditions d’arrêt. Préparez une estimation du travail avant l’exécution.

Cas entièrement fictif

Le responsable fictif veut une assurance globale après le seul résultat « Claire approuve ». L’équipe propose de vérifier aussi l’auteur, l’organisation, l’état, la version et la répétition de la demande.

Corrigé raisonné

Le résultat positif établit seulement le cas observé. La couverture ajoute les conditions et leurs contre-tests, sans transformer le laboratoire en validation de toute une architecture. Le mandat conserve les exclusions et la recette préserve l’approbation légitime.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Quel objectif rend une mission vérifiable ?

A. Une propriété et des critères observables

B. Une promesse de sécurité absolue

C. Le plus grand nombre d’outils

2. Quelle action reste autorisée dans ce parcours ?

A. Publier des cookies réels

B. Scanner un tiers sans accord

C. Tester le laboratoire local fictif

3. Quelle conclusion convient à un invariant testé ?

A. Elle garantit tous les déploiements

B. Elle précise la version et le périmètre

C. Elle remplace le consentement

4. Quelle donnée utiliser dans les preuves publiques ?

A. Un journal privé complet

B. Un mot de passe réel

C. Une identité fictive du TP

5. Que faire si une demande sort du mandat ?

A. Étendre le test discrètement

B. Contourner la protection

C. Arrêter et clarifier le périmètre

CHAPITRE 02 / 5 h indicatives

Examiner les frontières de confiance et les données de décision

Objectifs : Cartographier les décisions ; Annoncer les hypothèses d’identité ; Analyser le cycle de vie des états

Dessinez Atelier Boréal avant de lire les résultats. Le navigateur transmet une action et une identité fictive. Le serveur consulte une référence de personnes et un dossier en mémoire. La fonction de changement évalue plusieurs conditions, met à jour le document et conserve un événement. Une table de déduplication associe une clé à une demande précise. Le script de transaction utilise un autre modèle, dans une base SQLite temporaire, pour montrer une mise à jour conditionnelle et son mouvement associé.

Les composants ne sont pas interchangeables. Un bouton réservé à Claire ne prouve pas le contrôle serveur. Un rôle dans la référence ne vaut pas droit global sur toutes les organisations. Une clé de déduplication ne vaut pas secret d’authentification. Une empreinte de fichier ne démontre pas l’identité de son éditeur. Chaque flèche du dessin doit indiquer quelle donnée traverse la frontière, qui peut la modifier et quelle décision en dépend.

Hypothèses assumées du laboratoire

Le TP distribue les identités Alice, Claire et Bob. L’application suppose qu’un mécanisme d’identité réel aurait déjà établi ce contexte, puis illustre les règles qui suivent. Elle n’implémente pas un fournisseur d’identité, un mot de passe, un token fédéré ou un contrôle CSRF de compte réel. Le contrôle d’origine limite les appels de l’interface locale ; il ne transforme pas ce sélecteur en connexion de production.

Le stockage mémoire est également une hypothèse importante. Arrêter le serveur fait disparaître document modifié, événements et clés de déduplication. Une architecture réelle doit choisir les propriétés de persistance, de reprise et de concurrence adaptées à son besoin. Un résultat obtenu dans un seul processus ne prouve pas qu’une flotte de serveurs partage la même référence. Ces écarts figurent dans le modèle, pas dans une note oubliée après le rapport.

Décrire une frontière temporelle

La confiance porte aussi sur le moment de la décision. Une version lue avant une modification peut devenir périmée. Un rôle peut être révoqué entre deux actions. Une autorisation donnée sur le brouillon ne justifie pas nécessairement une approbation après changement de contenu. La question avancée est : quelles informations doivent être revérifiées dans l’opération qui produit l’effet ?

Atelier Boréal relie les conditions de la transition et la mise à jour sous un verrou commun. Le chapitre de concurrence montre pourquoi protéger seulement l’affectation finale peut rester insuffisant. Le projet vous demandera de comparer le dessin, les préconditions et la portée réelle de ce verrou. Un schéma clair doit rendre visible ce que le code ne couvre pas encore.

TP guidé

Méthode pas à pas

Dessinez navigateur, référence users, document, verrou, table approvals et événements. Ajoutez les flux de la transaction SQLite sur un dessin séparé. Identifiez quatre données contrôlées par le client et quatre références serveur. Marquez les états perdus au redémarrage et les décisions à reprendre dans une transition.

Exercice à réaliser

Présentez deux architectures possibles sur papier : un processus local et deux instances partageant une base. Expliquez pourquoi les propriétés du premier ne se transfèrent pas automatiquement au second.

Cas entièrement fictif

Deux serveurs fictifs ont chacun leur propre table de déduplication mémoire. Le commanditaire pense qu'une même clé sera reconnue globalement, alors que chaque instance peut ignorer l'effet réalisé par l'autre.

Corrigé raisonné

Le modèle doit identifier la référence commune nécessaire au besoin de déduplication. Le laboratoire ne prétend pas fournir ce déploiement distribué. La conclusion sépare le contrôle observé en mémoire et les exigences de persistance et de coordination encore à concevoir.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l'annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Quelle frontière étudier pour une décision métier ?

A. La source fiable des données de décision

B. La taille de l'écran

C. La couleur du bouton

2. Que prouvent les acteurs distribués du TP ?

A. Une politique sur des identités fictives

B. Une identité humaine vérifiée

C. Une authentification de production

3. Pourquoi décrire les composants absents ?

A. Pour délimiter la couverture

B. Pour cacher une lacune

C. Pour les déclarer sécurisés

4. Où placer le contrôle d'autorisation ?

A. Avant l'effet qu'il protège

B. Après l'écriture seulement

C. Uniquement dans le navigateur

5. Comment traiter un flux non observé ?

A. Inventer son résultat

B. Le déclarer conforme

C. Le marquer comme inconnu

CHAPITRE 03 / 5 h indicatives

Vérifier une politique combinant rôle, organisation et responsabilité

Objectifs : Décomposer une décision d’accès ; Construire une matrice de conditions ; Préserver la séparation des responsabilités

Une autorisation avancée combine plusieurs informations. Dans Atelier Boréal, Claire est approbatrice de A, Alice est auteure de A et Bob est approbateur de B. Le dossier 301 appartient à A et a été rédigé par Alice. La fonction can_approve exige une organisation identique, le rôle reviewer et une personne distincte de l’auteur. L’état et la version sont contrôlés dans Workflow.change. Ces conditions complémentaires doivent être étudiées comme une politique, pas comme un seul attribut de rôle.

Commencez par une table de décision. Les lignes décrivent des identités et des dossiers ; les colonnes décrivent chaque condition et la décision attendue. Une décision correcte peut être refusée pour plusieurs raisons à la fois. Pour isoler la cause, construisez un cas où une seule condition change. Dans les tests unitaires, une identité auteur portant le rôle reviewer reste refusée par la séparation des responsabilités. Ce cas est une donnée fictive de test, pas une modification de compte réel.

Expliquer un refus sans le surinterpréter

Alice refusée ne démontre pas encore la séparation des responsabilités, puisqu’elle n’a pas non plus le rôle d’approbatrice dans la référence distribuée. Pour vérifier spécifiquement l’interdiction d’auto-approbation, le test doit fournir un acteur fictif avec le bon rôle, la bonne organisation et l’identifiant de l’auteur. Le refus isole alors la dernière condition. Cette précision distingue une matrice argumentée d’une suite de captures sans hypothèse.

Bob constitue le cas de frontière d’organisation. Il possède le rôle attendu, mais son organisation ne correspond pas au dossier. Un contrôle fondé sur le seul rôle lui donnerait un accès indu. À l’inverse, la seule organisation autoriserait Alice. La conjonction de conditions répond au besoin formulé, à condition que leurs références soient établies côté serveur et qu’un changement ne les contourne pas ailleurs.

Relier la politique aux autres routes

Examinez aussi l’édition et la soumission. La route corrigée exige l’auteur du dossier pour ces actions et vérifie le brouillon avant la modification. Une bonne fonction can_approve ne corrige pas une autre route qui accepte une transition directement. La couverture doit donc relier fonctions de décision, points d’appel et cas d’usage. N’annoncez pas une politique globale après avoir vérifié un seul point d’entrée.

Le laboratoire garde les causes visibles pour l’apprentissage. Dans une application réelle, le contenu d’un message d’erreur se choisit selon le risque de divulgation et le besoin de diagnostic. Le TP ne démontre pas une politique universelle de réponses. Votre rapport doit conserver la règle métier, la méthode de preuve et la portée du constat.

TP guidé

Méthode pas à pas

Construisez au moins huit lignes de matrice rôle/organisation/auteur. Exécutez les tests de can_approve, puis les routes de workflow. Comparez Alice, Bob et Claire avant et après soumission. Vérifiez le cas fictif auteur et reviewer dans la suite unitaire, sans ajouter une nouvelle identité à l’interface de production.

Exercice à réaliser

Rédigez deux constats distincts : règle de rôle isolée et frontière d’organisation. Ajoutez le cas spécifique d’auto-approbation et un usage autorisé qui doit rester possible.

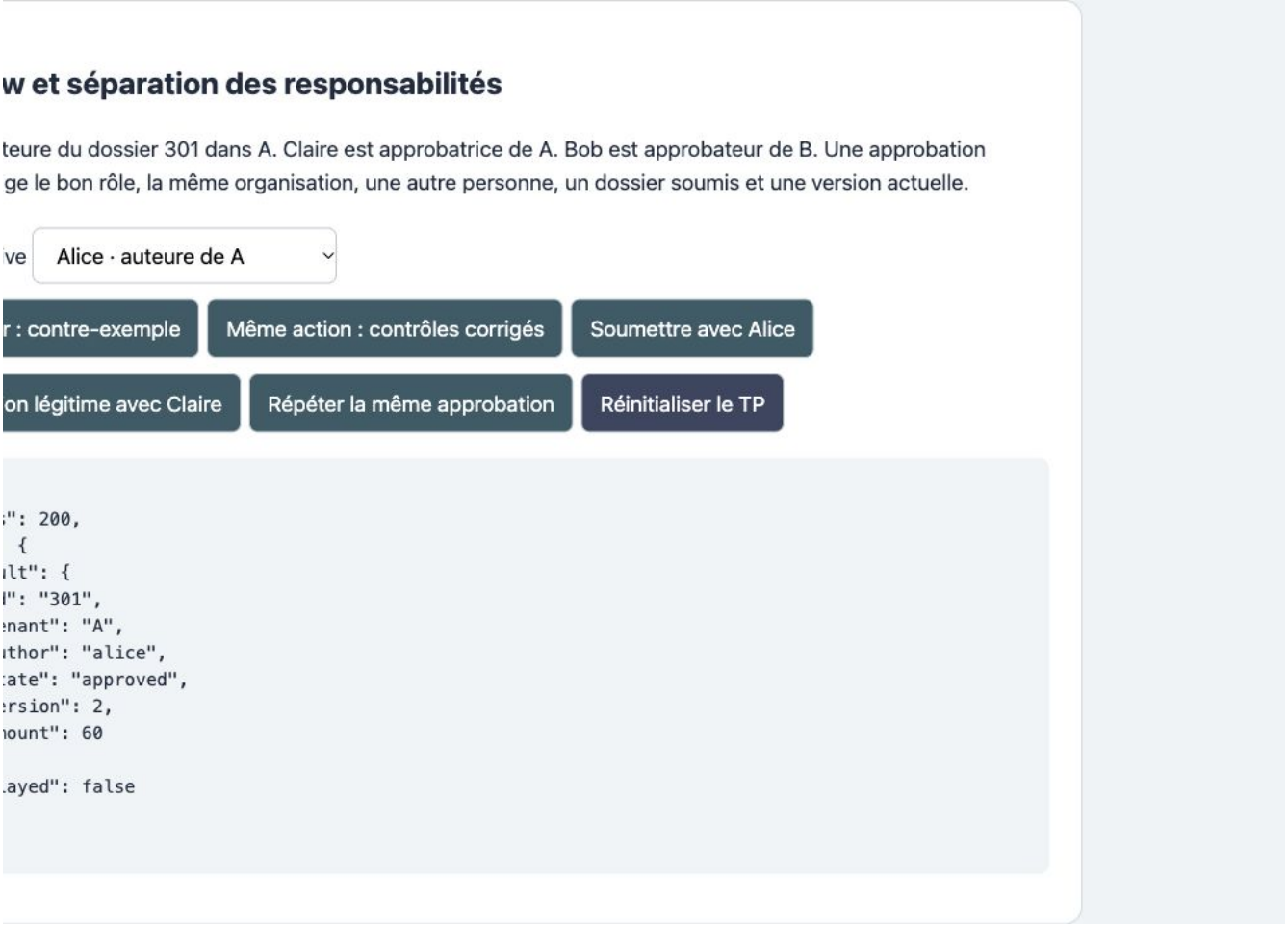
Cas entièrement fictif

Un rapport annonce « auto-approbation empêchée » uniquement parce qu’Alice, qui n’a pas le rôle reviewer, reçoit un refus. Il ne contient aucun cas isolant la condition auteur différent.

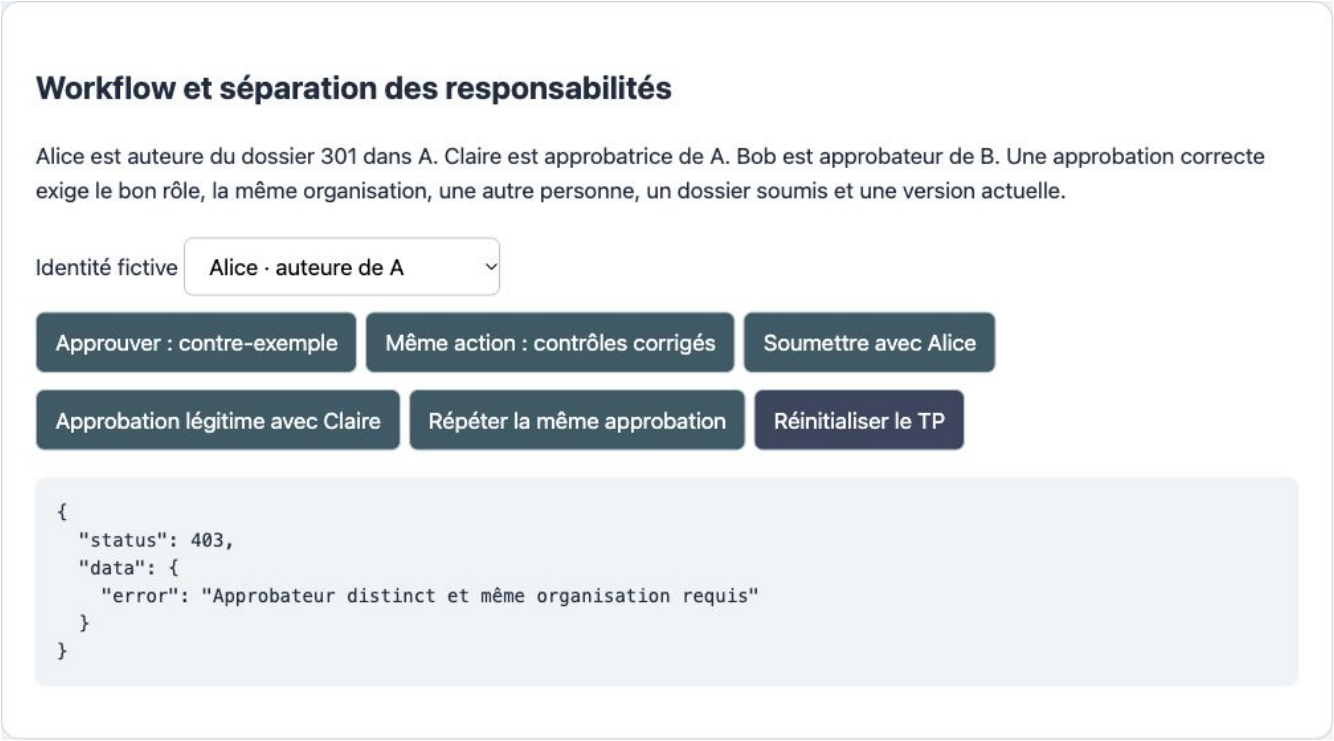
Corrigé raisonné

La conclusion est trop large pour cette preuve. Le test unitaire du rôle fictif adapté isole la séparation des responsabilités. Les routes et transitions sont examinées en complément ; les résultats sont attribués à leurs conditions précises.

Observer le laboratoire réel



Capture réelle du laboratoire local, macOS, 6 octobre 2026. Données fictives. Aucune extraction, requête externe ou identité réelle.



Capture réelle du laboratoire local, macOS, 6 octobre 2026. Données fictives. Aucune extraction, requête externe ou identité réelle.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Qui peut approuver le document après soumission ?

A. Alice, son auteur

B. Bob, d’une autre organisation

C. Claire, approbatrice de la même organisation

2. Un rôle approbateur suffit-il ?

A. Oui si le bouton est visible

B. Oui dans tous les cas

C. Non, organisation et auteur comptent aussi

3. Que vérifier après un refus ?

A. Uniquement la couleur du message

B. Une nouvelle clé secrète

C. L’état et la version inchangés

4. Quel contre-test est indispensable ?

A. Masquer toutes les commandes

B. Refuser tous les acteurs

C. Une approbation légitime réussie

5. Que décrit le succès du mode unsafe ?

A. Une certification officielle

B. Une attaque du site réel

C. Un contre-exemple pédagogique

CHAPITRE 04 / 5 h indicatives

Auditer les transitions et empêcher une décision sur un état périmé

Objectifs : Définir une machine à états ; Vérifier la version d'une demande ; Tester les transitions interdites

Le dossier suit une machine à états simple : draft, submitted, approved. Une modification n'est admise que dans draft ; une soumission part de draft ; une approbation part de submitted. Chaque transition réussie augmente la version. L'identité et l'organisation continuent à être vérifiées. Cette combinaison empêche qu'un client réutilise sans contrôle une décision préparée sur un état devenu ancien.

Écrivez les transitions autorisées avant les tests. Le dossier initial est draft en version 1, montant fictif 60. Alice le soumet avec la version 1 ; le dossier devient submitted en version 2. Claire approuve la version 2 ; le dossier devient approved en version 3. Une demande de Claire utilisant encore la version 1 doit échouer. Le même effet ne peut pas être obtenu en changeant seulement le nom d'un bouton.

Vérifier le moment de la décision

Une interface peut avoir affiché un brouillon valide avant une soumission effectuée ailleurs. Le contrôle serveur doit examiner l'état actuel au moment de la mutation. Dans le laboratoire, le verrou englobe la vérification et la mise à jour de l'objet. Le champ version est un garde supplémentaire pour exprimer que la demande se rapporte à la référence attendue. Un entier transmis par le client ne devient pas autorité : il est comparé à la référence serveur.

L'opération d'édition illustre une autre limite. Le montant fictif doit être un entier entre 1 et 100, et l'auteur doit agir sur un brouillon. Le fait qu'une valeur respecte le type ne justifie pas une modification après soumission. Le rapport distingue donc contrat d'entrée, identité autorisée, état métier et version. Chaque contrôle doit avoir ses cas, au lieu de transformer tous les refus en un résultat unique.

Construire une recette de transitions

Prévoyez une réinitialisation avant chaque séquence indépendante. Les routes de contre-exemple et corrigée utilisent deux documents distincts en mémoire. Si la première route accepte une approbation directement sur le brouillon, elle ne change pas le dossier de la route correcte. Cette séparation évite qu'un essai précédent explique à tort le résultat suivant. La preuve doit annoncer quelle implémentation et quelle version ont été utilisées.

Une machine à états réelle peut être plus complexe : annulation, reprise, expiration, délégation ou correction. Il faut alors définir les invariants de chaque transition, les effets déjà produits et le traitement d'une demande interrompue. Ces cas sont étudiés sur papier ici. Le programme n'implémente ni paiement, notification externe, délégation réelle ni persistance de production.

TP guidé

Méthode pas à pas

Réinitialisez le TP. Essayez Claire sur un brouillon, puis Alice sur la route corrigée. Soumettez avec Alice, approuvez avec Claire et notez les versions réellement obtenues. Dans les tests, observez l'édition après soumission et la demande d'approbation portant une version périmée. Construisez une table transition/préconditions/effet/contre-test.

Exercice à réaliser

Produisez une séquence de trois états et trois demandes refusées. Expliquez comment une capture d’interface ancienne pourrait induire une conclusion incorrecte sur l’état actuel.

Cas entièrement fictif

Claire prépare une approbation en version 1. Alice soumet ensuite le dossier, qui passe en version 2. Claire transmet sa demande ancienne sans relire la référence.

Corrigé raisonné

La route correcte refuse cette demande périmée. Claire doit agir sur le dossier soumis et sa version actuelle. Les conditions sont vérifiées dans l’opération ; une correction qui se limite à rafraîchir le bouton ne constitue pas la preuve du contrôle serveur.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Quelle transition est légitime ?

A. Approved vers draft sans règle

B. Draft vers approved sans soumission

C. Draft vers submitted par l’auteur

2. Que faire d’une version périmée ?

A. Refuser sans mutation

B. Ignorer le champ

C. Écraser la version actuelle

3. Pourquoi contrôler état et version ensemble ?

A. Pour chiffrer le document

B. Pour remplacer les permissions système

C. Pour relier la décision à l’objet attendu

4. Quel type de version le TP attend-il ?

A. Un entier métier, pas un booléen

B. Toute chaîne

C. Une valeur vraie quelconque

5. Quelle preuve compléter après un statut 200 ?

A. Le document et les événements

B. Un scan distant

C. Un slogan de confiance

CHAPITRE 05 / 5 h indicatives

Démontrer une perte de mise à jour et vérifier l’atomicité

Objectifs : Formuler un invariant numérique ; Expliquer une interleaving contrôlée ; Comparer verrou et transaction

Le budget fictif commence à 100. Deux demandes de 60 arrivent dans le même exercice. Une règle cohérente n’en admet qu’une, avec un solde final de 40. Le contre-exemple lit le solde avant une barrière, vérifie cette ancienne valeur puis écrit le résultat. Les deux fils peuvent lire 100 et admettre chacun 60 ; leurs écritures finissent toutes deux à 40. Le solde affiché paraît plausible, mais il ne correspond plus aux deux opérations acceptées.

Écrivez l’invariant : solde final = solde initial - somme des montants admis. Pour deux demandes admises de 60, cette expression vaut -20 ; le résultat observé 40 montre une perte de mise à jour. Il ne prouve pas un versement, un débit bancaire ou une dette réelle : les unités et opérations sont entièrement fictives. Le nombre d’opérations acceptées est aussi une preuve nécessaire, pas seulement la valeur finale.

Une barrière rend le scénario reproductible

Le programme utilise deux fils et une barrière de rendez-vous. Ils attendent d’avoir lu leur référence avant de continuer. Le TP crée ainsi une interleaving déterminée pour expliquer le défaut, sans dépendre d’un grand nombre de demandes ni d’un hasard de timing. Le test possède des délais bornés ; une exécution incomplète est signalée au lieu d’être transformée en résultat attendu.

Dans le contre-exemple, le verrou protège l’affectation finale et le compteur, mais la lecture et la condition ont déjà eu lieu hors de cette section. Cela suffit à montrer que « il y a un verrou » ne constitue pas une preuve d’atomicité. La version correcte place vérification et décrément ensemble sous le verrou. Une demande passe, l’autre constate le solde devenu insuffisant. L’invariant est alors respecté dans ce processus.

Comparer avec une transaction conditionnelle

transaction_tp.py crée une base SQLite temporaire et un budget fictif. Deux connexions lisent la version initiale. Une transaction prend ensuite la décision par une mise à jour conditionnelle : bonne version et solde suffisant. Le nombre de lignes modifiées indique si la condition a réellement produit un effet. Un mouvement n’est ajouté que si la mise à jour réussit, puis l’ensemble est validé dans la même transaction.

Ce script montre une opération admise, une ligne non modifiée et un mouvement enregistré. Il ne compare pas tous les niveaux d’isolation ni les moteurs de base. Le verrou Python et la transaction SQLite ont des portées différentes. Un système distribué doit préciser ses références partagées, ses garanties de persistance et son comportement après interruption. Un simple délai ajouté entre deux requêtes ne répond pas à ces exigences.

TP guidé

Méthode pas à pas

Lancez les deux boutons de concurrence, puis exécutez verifier.py et transaction_tp.py. Relevez demandes admises, solde, valeur attendue et invariant. Dessinez l’ordre des lectures, de la barrière, des décisions et des écritures. Relisez le périmètre des deux verrous et la mise à jour conditionnelle SQLite.

Exercice à réaliser

Produisez deux diagrammes de séquence et un tableau de résultats réels. Expliquez pourquoi verrouiller seulement l’écriture n’a pas protégé la décision précédente.

Cas entièrement fictif

Deux demandes fictives sont marquées acceptées alors que le solde final n’en reflète qu’une. Une capture ne montre que « 40 » et conclut à tort que le budget est cohérent.

Corrigé raisonné

Il faut rapprocher le solde du nombre d’effets admis. Le contre-exemple perd une mise à jour ; la correction admet une seule demande. La transaction ajoute aussi le mouvement dans l’opération conditionnelle, sans revendication de garantie sur d’autres architectures.

Observer le laboratoire réel



Capture réelle du laboratoire local, macOS, 6 octobre 2026. Données fictives. Aucune extraction, requête externe ou identité réelle.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Deux retraits de 60 acceptés sur 100 laissent 40 : que conclure ?

A. Deux opérations de 30 ont eu lieu

B. L’invariant des mouvements est violé

C. Le total est cohérent

2. Quel résultat attend-on du mode corrigé ?

A. Deux succès et 40

B. Aucun contrôle

C. Un succès et un solde de 40

3. Que garantit le verrou du TP ?

A. Tous les services distribués

B. Une portée dans son processus

C. Une transaction externe

4. Que teste transaction_tp.py ?

A. Un cluster de production

B. Une mise à jour conditionnelle et un mouvement cohérents

C. Un paiement bancaire

5. Pourquoi utiliser une barrière pédagogique ?

A. Pour attaquer un serveur

B. Pour chiffrer les threads

C. Pour rendre le chevauchement observable

CHAPITRE 06 / 5 h indicatives

Relier une clé de déduplication à une demande précise

Objectifs : Distinguer répétition et nouvel effet ; Vérifier une empreinte de demande ; Annoncer la durée de la garantie

Une demande peut être répétée après une réponse perdue. Répondre une seconde fois ne doit pas nécessairement produire un second effet. Atelier Boréal associe une clé de déduplication à l'acteur, au dossier, à l'action, à la version et au montant de la demande. Après une approbation correcte, la répétition exacte renvoie le résultat conservé avec replayed=true. Elle n'ajoute pas une transition ou un événement supplémentaire.

La clé n'est ni mot de passe ni preuve d'identité. Dans ce TP, elle est une étiquette choisie pour rapprocher deux demandes. Les autorisations continuent à être vérifiées. Une personne d'une autre organisation ne devient pas approbatrice parce qu'elle connaît cette étiquette. Le rapport explique la portée de la référence : dans le programme, elle vit en mémoire et disparaît au redémarrage.

La même clé ne signifie pas le même contenu

La comparaison de demande évite qu'une clé serve à masquer une modification. Si une nouvelle demande porte la même clé mais un autre contenu, la route correcte retourne 409. Le programme construit une représentation JSON ordonnée des champs pertinents, puis la compare à celle conservée. Il ne prétend pas fournir une norme universelle de canonicalisation pour tous les protocoles ou langages.

L'ordre des contrôles compte. Une répétition exacte peut porter la version initialement approuvée, devenue ancienne après l'effet. La table de déduplication reconnaît cette demande et restitue le résultat déjà obtenu. Une nouvelle demande avec une autre clé doit, elle, satisfaire la version et l'état actuels. Sans cette distinction, un contrôle de version placé au mauvais endroit pourrait rendre toute reprise exacte impossible après une réponse perdue.

Séparer effet local et effet externe

Le laboratoire n'envoie aucun email, paiement ou message de broker. Dans un système réel, conserver une ligne de déduplication ne garantit pas à lui seul qu'un effet externe a eu lieu une seule fois. Il faut raisonner sur le point de validation, les interruptions et les reprises. Un modèle d'outbox peut relier un changement validé et un événement à traiter, mais le consommateur doit également définir son comportement face à une livraison répétée.

Étudiez ces états sur papier : demande non reçue, effet validé mais réponse perdue, événement produit mais accusé absent. Une issue inconnue appelle une réconciliation de la même demande et du même identifiant, pas une nouvelle mutation aveugle. La garantie dépend de la persistance, de la durée de conservation et de la référence commune entre instances. Ces dimensions restent annoncées comme conception documentaire ici.

TP guidé

Méthode pas à pas

Réinitialisez, soumettez avec Alice et approuvez avec Claire. Répétez la même approbation. Comparez résultat, version, replayed et nombre d'événements. Dans verifier.py, observez la même clé avec contenu différent. Dessinez ensuite un effet local et un effet externe hypothétique avec leurs points d'interruption.

Exercice à réaliser

Rédigez une procédure de reprise pour une réponse perdue. Indiquez les identifiants à conserver, les preuves à relire et la limite du stockage mémoire.

Cas entièrement fictif

Une personne ne reçoit pas la réponse d’approbation et recommence avec une nouvelle clé. Elle interprète chaque réponse réseau comme la preuve d’un nouvel effet, sans examiner l’état du dossier.

Corrigé raisonné

La demande originale doit être réconciliée avec sa clé et son contenu. Le TP restitue l’effet conservé lors d’une répétition exacte, mais ne garantit pas cette mémoire après redémarrage ni un effet externe non implémenté.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Même clé et même demande après succès : résultat attendu ?

A. Effacement du document

B. Résultat réutilisé sans transition supplémentaire

C. Nouvelle approbation obligatoire

2. Même clé avec contenu différent : résultat attendu ?

A. Succès automatique

B. Nouvelle identité

C. Conflit

3. La clé de déduplication est-elle un mot de passe ?

A. Oui si elle est longue

B. Oui toujours

C. Non

4. Que devient la mémoire de déduplication au redémarrage ?

A. Elle disparaît dans ce TP

B. Elle devient une sauvegarde

C. Elle est envoyée à un service

5. Après une réponse perdue, quelle première démarche ?

A. Relire sous les mêmes identifiants

B. Déclarer le refus

C. Créer une nouvelle demande au hasard

CHAPITRE 07 / 5 h indicatives

Contrôler un plan d’archive sans écrire hors du dossier prévu

Objectifs : Séparer nom, contenu et destination ; Analyser des chemins propres aux OS ; Définir des budgets de traitement

Une archive contient des noms qui deviennent souvent des chemins lors d’une extraction. Leur apparence ne suffit pas à établir une destination admissible. Atelier Boréal distribue un nom relatif documents/lecture.txt, une traversée ../hors-perimetre.txt et un chemin Windows fictif. Le contre-exemple accepte ces noms dans un plan. La version corrigée rejette ceux qui ne respectent pas le contrat. Aucune de ces routes n’extrait de fichier.

Cette limite est volontaire : vous démontrez la qualité de la planification sans créer une écriture hors périmètre. Ne décrivez pas le résultat comme une exploitation d’extraction réalisée. Le plan corrigé examine caractères, séparateurs, chemin absolu, segments point et double point, marque de lecteur et doublons normalisés. Les essais associent chaque nom à une raison d’acceptation ou de refus.

Le système de fichiers ajoute des questions

Les chemins ne se comportent pas tous de la même façon selon l’OS. Windows connaît des lecteurs et d’autres formes de noms ; Unix utilise une racine absolue ; des liens peuvent changer la destination réelle d’un chemin. Une normalisation textuelle n’examine pas à elle seule le système de fichiers ni une course entre vérification et utilisation. Si une extraction réelle était conçue, il faudrait définir les opérations et leur confinement avec ces contraintes.

Le TP ne fournit pas cette extraction de production. La fonction inspect_archive lit les métadonnées d’une archive créée en mémoire. Elle rejette les liens symboliques signalés dans les attributs, les entrées chiffrées et les chemins non admis. Le rapport conserve ce niveau de preuve : analyse de noms et de métadonnées, pas couverture de tous les comportements d’un extracteur natif.

Prévoir le budget avant le contenu

Une petite archive peut annoncer un contenu décompressé beaucoup plus important. inspect_archive fixe une taille d’entrée, un nombre d’entrées, une taille totale annoncée, une taille par entrée et un rapport de compression admis. Ces seuils sont pédagogiques, adaptés aux minuscules données distribuées. Ils ne constituent pas un dimensionnement universel de service public.

La vérification doit être effectuée avant une opération coûteuse ou une écriture. Pour un vrai traitement, les tailles annoncées ne remplaceraient pas des limites effectives pendant lecture et extraction. La profondeur des dossiers, les collisions de noms, le stockage temporaire et le nettoyage après interruption demanderaient aussi une décision. Dans votre projet, présentez ces contrôles manquants comme exigences supplémentaires, sans inventer leur exécution.

TP guidé

Méthode pas à pas

Comparez Plan sans contrôle et Plan avec contrôles. Utilisez Lire l’archive en mémoire pour voir archiveAccepted=false. Relisez archive_plan et inspect_archive. Ajoutez uniquement dans une copie des tests de noms absolus, doublons et segments point ; n’ajoutez aucune extraction. Reliez chaque seuil à un budget écrit dans le mandat.

Exercice à réaliser

Produisez une matrice nom/forme/attendu/observé et une fiche de conception d’extraction sur papier. Séparez les contrôles exécutés des exigences système non testées.

Cas entièrement fictif

Le plan contient un nom traversant le dossier prévu. Le rapport affirme à tort qu’un fichier système a été remplacé, alors que le code n’a effectué aucune extraction.

Corrigé raisonné

Le défaut démontré concerne le plan de destination accepté. La correction rejette les noms non admis et l’archive correspondante. L’effet d’écriture n’a pas été exécuté ; les limites de liens, de confinement et de concurrence restent explicites.

Observer le laboratoire réel



Capture réelle du laboratoire local, macOS, 6 octobre 2026. Données fictives. Aucune extraction, requête externe ou identité réelle.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Le TP extrait-il une archive ?

A. Non, il calcule un plan

B. Oui sur un serveur distant

C. Oui dans le système

2. Comment traiter ../ dans un nom ?

A. Le retirer silencieusement

B. Rejeter selon les règles du plan

C. L’accepter comme sûr

3. Quelle autre borne inspecte le ZIP en mémoire ?

A. Nombre, tailles et ratio de compression

B. La réputation du compte

C. Le solde bancaire

4. Que signifie un doublon normalisé ?

A. Une signature valide

B. Deux noms visant le même nom prévu

C. Une extension sûre

5. Un plan accepté prouve-t-il toute extraction sûre ?

A. Oui sur tous les OS

B. Non, une vraie extraction exige d’autres garanties

C. Oui universellement

CHAPITRE 08 / 5 h indicatives

Analyser une liste fermée d’URL et ses limites

Objectifs : Distinguer chaîne et destination ; Vérifier les composants d’une URL ; Limiter les conclusions sur le réseau

Une adresse peut contenir un nom d’utilisateur, un hôte, un port, une route, une requête et un fragment. Une recherche naïve de texte ne démontre pas la destination visée. Dans le TP, seule l’adresse http://127.0.0.1:8878/api/health est admise. Le classificateur exige le schéma HTTP, l’hôte littéral, le port et la route exacts, sans identifiant utilisateur, query ni fragment. Il ne déclenche aucune requête sortante.

Une fixture contient une adresse qui commence visuellement par la boucle locale mais place exemple.invalid après le caractère @. L’analyse des composants identifie cet hôte comme destination, et la liste fermée la refuse. Les autres fixtures changent schéma, route ou composants. Elles servent à montrer la différence entre lecture humaine d’une chaîne et contrat de destination, sans visiter ces adresses.

Définir le besoin avant la liste

Une liste fermée répond à un besoin étroit : accepter une seule forme distribuée. Elle ne fournit pas un accès arbitraire à Internet. Le classificateur refuse également caractères de contrôle, espaces et séparateurs non admis. Une erreur de parsing est traitée comme un refus. Le résultat indique outgoingRequests=0 afin de préserver la distinction entre classification et exécution réseau.

Cette liste n’est pas une validation générale de tous les liens d’un site. Une application réelle qui résout des noms et suit des redirections doit examiner les destinations effectivement contactées à chaque étape, ainsi que les références réseau accessibles au processus. Une validation initiale de chaîne peut devenir insuffisante si la résolution ou une redirection change la portée. Ces cas sont étudiés sur un diagramme ici, sans sondage de réseau tiers.

Relier le contrôle à la frontière serveur

Le risque de requête serveur induite concerne le service qui reçoit une entrée et choisit une destination avec ses propres accès. Ce n’est pas automatiquement le même comportement qu’un navigateur ouvrant un lien. Votre modèle doit identifier le programme qui ferait la requête, les ressources accessibles et les décisions avant chaque contact. Ne transposez pas le résultat du classificateur en preuve d’une sortie réseau réellement confinée.

Un service réel peut aussi choisir de ne jamais interpréter une URL fournie par le client : un identifiant interne renvoie vers une destination configurée côté serveur. Cette décision réduit certains écarts, mais exige encore de protéger la configuration et les permissions associées. Pour ce niveau, comparez deux conceptions sur papier et justifiez celle adaptée au besoin étroit du TP.

TP guidé

Méthode pas à pas

Cliquez Examiner les URL fictives. Relevez hôte, port, route et décision pour chaque fixture. Relisez destination_allowed, puis les cas du test. Dessinez séparément un service qui résout un nom et suit une redirection ; marquez les contrôles qui ne sont pas exécutés dans ce laboratoire.

Exercice à réaliser

Rédigez une politique de destination pour une unique route locale et une autre pour un service hypothétique sur papier. Annoncez clairement pourquoi les preuves de l'une ne valident pas l'autre.

Cas entièrement fictif

Une capture montre cinq classifications et le rapport annonce « sorties réseau protégées ». Pourtant, aucun appel sortant ni résolution de nom n'a eu lieu.

Corrigé raisonné

La preuve établit les décisions du classificateur sur les fixtures, sans résultat réseau. La conception de sortie doit être documentée séparément. Aucune adresse extérieure n'est ouverte pour enrichir ce TP.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l'annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Quelle destination le TP accepte-t-il ?

A. Tout domaine HTTPS

B. Tout nom contenant localhost

C. HTTP exact vers 127.0.0.1:8878/api/health

2. Le classificateur envoie-t-il une requête ?

A. Non

B. Oui vers Internet

C. Oui vers chaque URL

3. Que faire d'un utilisateur dans l'URL ?

A. Le confondre avec le serveur

B. Le refuser dans ce TP

C. Le publier comme secret

4. Pourquoi parler des redirections ?

A. Elles rendent tout URL sûr

B. Le TP les suit déjà

C. Un futur client pourrait changer de destination

5. Une chaîne localhost présente dans le texte suffit-elle ?

A. Seulement si le texte est long

B. Oui

C. Non

CHAPITRE 09 / 5 h indicatives

Vérifier l’intégrité d’un message sans inventer une garantie d’identité

Objectifs : Interpréter un HMAC ; Définir une représentation canonique ; Distinguer intégrité, identité et rejeu

Un message intact peut rester dangereux s’il est utilisé dans le mauvais contexte. Le TP d’Atelier Boréal signe un petit objet contenant un document, un montant et une version. Il montre une propriété précise : une modification du montant rend la vérification négative lorsque le calcul emploie la même clé et la même représentation. Il ne fournit ni service d’identité ni stockage de secrets.

Partir de la propriété réellement testée

Dans controles.py, canonical produit du JSON UTF-8 avec les clés triées et des séparateurs fixes. sign_demo calcule un HMAC SHA-256. verify_demo compare la valeur recalculée avec compare_digest. Le laboratoire distribue volontairement une clé publique de démonstration. Toute personne disposant du fichier peut produire un autre tag valide. Ce choix rend le mécanisme observable ; il interdit de présenter la démonstration comme une protection contre les participants du cours.

L’intégrité signifie ici que le message correspond au tag sous la clé employée. La confidentialité n’est pas obtenue : le document et le montant restent lisibles. L’identité humaine n’est pas prouvée. La disponibilité du service et la pertinence métier de la décision ne sont pas garanties non plus. Écrire ces distinctions évite de transformer une primitive correcte en promesse de sécurité générale.

Définir ce que l’on signe

Deux représentations d’une même donnée peuvent produire des octets différents. Le TP accepte le changement d’ordre des propriétés grâce à sa canonicalisation, mais ne constitue pas un standard universel de JSON canonique. Les nombres, Unicode, champs facultatifs et données binaires nécessitent une spécification adaptée aux systèmes qui communiquent. Une revue vérifie aussi les champs omis : signer un montant sans destinataire pourrait autoriser une substitution de contexte.

Dans le dossier fictif, le groupe propose une enveloppe comprenant version de format, identifiant de document, opération et échéance. Cette proposition n’est pas implémentée dans l’atelier. Elle doit être testée avec des messages expirés, des versions non reconnues et un contexte différent avant d’être considérée comme une correction opérationnelle.

Examiner le cycle de vie et les répétitions

Une clé de production devrait être gérée par un dispositif adapté, avec droits minimaux, rotation et traitement de compromission. Le cours n’en crée aucune. Sur papier, distinguer identifiant de clé, période d’acceptation, événement de retrait et conséquences pour les anciens messages. Ne pas copier une clé privée réelle dans un exercice ou un rapport.

Un HMAC valide n’empêche pas de soumettre deux fois le même message. La protection contre le rejeu repose sur un protocole et un état cohérents : expiration vérifiée, identifiant unique ou association à une transition autorisée. Le chapitre sur la déduplication apporte une autre propriété, sans remplacer cette conception.

TP guidé

Méthode pas à pas

Cliquer Comparer original et modification ; relever le message, les deux résultats et l’indication de clé publique ; exécuter les tests de canonicalisation ; annoter chaque propriété démontrée et chaque propriété absente.

Exercice à réaliser

Produire une matrice de six lignes : message original, montant modifié, ordre des clés modifié, contexte omis, seconde soumission et clé connue. Indiquer pour chaque ligne ce que le TP teste réellement.

Cas entièrement fictif

L’équipe fictive obtient originalValid=true et alteredValid=false puis annonce une authentification forte. La clé distribuée et l’absence de vérification de l’émetteur contredisent cette conclusion.

Corrigé raisonné

Attendre une vérification positive pour l’original et négative pour le montant modifié. L’ordre des propriétés ne change pas le tag de ce TP. Le rejeu et l’identité de l’émetteur demeurent hors de sa garantie. Ne proposer aucun algorithme cryptographique maison.

Observer le laboratoire réel



Capture réelle du laboratoire local, macOS, 6 octobre 2026. Données fictives. Aucune extraction, requête externe ou identité réelle.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Que démontre alteredValid=false ?

A. La modification du montant ne correspond plus au tag

B. L’auteur humain est vérifié

C. Le message est chiffré

2. Pourquoi la clé distribuée ne protège-t-elle pas des participants ?

A. Ils peuvent calculer un tag

B. SHA-256 est absent

C. Le message disparaît

3. Que change l’ordre des propriétés dans ce TP ?

A. Toujours le montant

B. Rien après canonicalisation prévue

C. Toujours l’identité

4. Un HMAC empêche-t-il seul le rejeu ?

A. Oui toujours

B. Non

C. Oui sans état

5. Quelle proposition est correcte ?

A. Inventer un chiffrement

B. Copier la clé du TP en production

C. Prévoir une gestion de clés adaptée en production

CHAPITRE 10 / 5 h indicatives

Relier une revue avancée aux décisions et aux contre-tests

Objectifs : Suivre une décision jusqu'à l'écriture ; Construire une matrice de propriétés ; Écarter les conclusions fondées sur un mot isolé

La présence d'un contrôle dans un fichier n'indique pas qu'il s'applique à chaque chemin. Une revue avancée suit les données et les décisions depuis la requête jusqu'à l'effet observable. Pour Atelier Boréal, le point d'entrée reconnaît une action et une identité fictive, choisit un mode pédagogique puis appelle Workflow.change. La lecture du code complète les observations HTTP ; aucune des deux méthodes ne remplace l'autre.

Cartographier les chemins plutôt que les fonctions seules

Dessiner le parcours du document : validation du corps, sélection du workflow, autorisation, état attendu, version, écriture et événement. Associer à chaque étape une condition et une preuve. Le mode unsafe sert de contre-exemple annoncé. Le qualifier de vulnérabilité cachée du site VLR serait faux : il appartient au laboratoire volontairement imparfait et n'est pas son service de production.

Un contrôle doit précéder l'effet qu'il protège. Vérifier une version après avoir modifié le document arrive trop tard. Une autorisation calculée sur une copie ancienne peut devenir insuffisante si les données changent. Dans le TP, un verrou encadre la décision et l'écriture en mémoire ; la portée de ce verrou reste limitée au processus en cours.

Écrire des tests qui peuvent contredire l'hypothèse

Chaque propriété reçoit un cas autorisé, un cas interdit et une observation de l'état final. Pour la séparation des responsabilités, Claire peut approuver après soumission, Alice ne le peut pas, Bob appartient à une autre organisation. Vérifier seulement le statut HTTP ne suffit pas : comparer aussi version, état et nombre d'événements.

La matrice doit couvrir les bornes pertinentes. Une version booléenne ne doit pas être acceptée comme un entier métier. Une clé de déduplication réutilisée avec une demande différente doit produire un conflit. Un échec ne doit pas laisser une écriture partielle. Ces conditions sont explicitement présentes dans verifier.py et restent lisibles sans outil d'analyse payant.

Interpréter les outils et les modifications

Un avertissement automatique est un candidat à examiner. Il faut connaître la version analysée, le chemin accessible et les conditions nécessaires. Le cours n'installe aucun scanner ni dépendance tierce. Pour illustrer une régression, travailler sur une copie du dossier, modifier une condition pédagogique, lancer les tests puis expliquer celui qui détecte la différence. Ne jamais modifier le service public pour provoquer un défaut.

Le test qui suit exactement l'implémentation peut partager son erreur. Une assertion métier indépendante, comme « une approbation ne peut pas venir de l'auteur », aide à éviter ce problème. Le groupe peut aussi calculer séparément l'invariant du budget à partir des mouvements acceptés, plutôt que de recopier la valeur retournée par le programme.

TP guidé

Méthode pas à pas

Lire change et can_approve ; tracer deux chemins jusqu'à l'écriture ; relier chaque branche aux tests existants ; compléter une propriété sans changer l'application publique.

Exercice à réaliser

Livrer une matrice condition, cas légitime, cas interdit, état avant, état après et limite. Vérifier un chemin de succès complet et deux refus sans mutation.

Cas entièrement fictif

Une recherche trouve compare_digest et conclut que toutes les décisions sont authentifiées. La fonction ne protège que le tag de démonstration et ne vérifie pas la responsabilité de l'approbateur.

Corrigé raisonné

Distinguer la fonction cryptographique, la politique métier et le routage effectif. Une revue convaincante décrit les chemins exercés et les chemins absents, et ne transforme pas un signal de recherche en résultat d'audit.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l'annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Que doit suivre la revue ?

A. Un mot isolé

B. Le chemin jusqu'à l'effet

C. La décoration

2. Quel ensemble de tests est utile ?

A. La syntaxe seule

B. Tous les refus seulement

C. Succès légitime, refus et état final

3. Où essayer une régression pédagogique ?

A. Sur une copie personnelle du TP

B. Sur le site public

C. Sur un tiers

4. Un avertissement automatique est-il un verdict ?

A. Oui

B. Oui si rouge

C. Non, un candidat à vérifier

5. Pourquoi calculer séparément un invariant ?

A. Pour éviter de partager l'erreur d'implémentation

B. Pour masquer une limite

C. Pour supprimer les tests

CHAPITRE 11 / 5 h indicatives

Examiner une chaîne de livraison et la provenance de ses composants

Objectifs : Lire un inventaire fictif ; Différencier hash et provenance ; Préparer une livraison reproductible et limitée

Un logiciel peut être correct dans son dépôt et livré sous une autre forme. Le niveau avancé ajoute donc une revue de la chaîne de fabrication : sources, dépendances, construction, artefact, approbation et installation. Le cours utilise une bibliothèque standard Python et un inventaire pédagogique fictif. Il ne publie aucune dépendance et n'appelle aucun fournisseur de compilation.

Séparer les objets de preuve

Le manifeste de fichiers du laboratoire permet de vérifier des octets. Un hash SHA-256 correspondant indique qu'un fichier correspond au manifeste consulté. Si un tiers remplace à la fois le fichier et un manifeste non signé, cette concordance ne prouve pas son origine. Le dossier précise donc que son manifeste est une preuve de cohérence de la livraison, sans signature d'éditeur indépendante.

Un inventaire de composants décrit ce qui est annoncé dans un produit. Il ne garantit pas que chaque composant est effectivement chargé. Le fichier inventaire-fictif.json présente trois composants inventés, leurs usages et leurs propriétaires fictifs. Aucun numéro de vulnérabilité réel n'y est associé. L'avis BOREAL-DEMO-01 est lui aussi fictif ; il sert à poser une question de présence et de configuration, pas à alerter sur un fournisseur réel.

Relier un avis à un usage

La fiche doit distinguer composant inventorié, version, chemin d'appel et précondition. Un parseur archivé mais jamais importé n'a pas le même statut qu'un composant accessible depuis une entrée. Une configuration absente du périmètre reste inconnue. On ne clôture pas un risque sur une simple supposition d'inaccessibilité.

Les élèves rédigent une décision pour chaque composant : présent et exercé, présent mais usage non démontré, absent du paquet inspecté ou inconnu. La décision doit indiquer les fichiers et commandes qui la soutiennent. Elle peut être révisée si une nouvelle preuve révèle un autre chemin.

Utiliser les référentiels sans promettre une conformité totale

Le SSDF décrit des pratiques de développement sécurisé. ASVS propose des exigences vérifiables pour les applications. Le cours cite SSDF 1.1 et ASVS 5.0.0 comme repères datés, avec des correspondances partielles explicites. Une poignée de tests de TP ne suffit pas à déclarer une conformité complète à ces référentiels. Ils ne délivrent pas le diplôme interne de VLR.

Le pipeline fictif propose contrôle de la source, tests, construction isolée, inventaire, vérification d'intégrité, approbation distincte et recette. Il reste un scénario sur papier. Les permissions, protections et configurations réelles d'un hébergeur ne doivent pas être modifiées pour « vérifier » ce scénario.

TP guidé

Méthode pas à pas

Lire inventaire-fictif.json et pipeline-fictif.md ; calculer le hash de preuve-locale.txt ; distinguer la concordance de fichiers de l’authentification de l’éditeur ; faire une correspondance limitée avec les sources primaires citées.

Exercice à réaliser

Construire une fiche de provenance comprenant entrée, transformation, sortie, propriétaire fictif, vérification, condition d’arrêt et preuve absente. Traiter un composant déclaré mais non exécuté sans inventer son comportement.

Cas entièrement fictif

Un manifeste copié depuis le même paquet que l’artefact est présenté comme une signature indépendante. Il n’existe pourtant ni clé de confiance ni mécanisme de signature dans l’exercice.

Corrigé raisonné

Le hash sert à comparer les octets à une référence. La provenance authentifiée demande une référence et une chaîne de confiance adaptées. Indiquer clairement que l’inventaire et l’avis sont fictifs, et que le pipeline n’a pas été déployé.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Que prouve un hash correspondant ?

A. L’identité humaine

B. La sûreté complète

C. La concordance avec une référence

2. Que vaut le manifeste non signé du même paquet ?

A. Une cohérence interne limitée

B. Une signature indépendante

C. Un agrément

3. Que représente BOREAL-DEMO-01 ?

A. Un certificat

B. Une vulnérabilité réelle annoncée

C. Un avis entièrement fictif

4. Un composant inventorié est-il forcément exécuté ?

A. Oui à chaque requête

B. Oui

C. Non

5. Quelques TP prouvent-ils une conformité ASVS complète ?

A. Oui

B. Non

C. Oui après le QCM

CHAPITRE 12 / 5 h indicatives

Construire une détection argumentée et un exercice de réponse

Objectifs : Interpréter les événements du laboratoire ; Définir une règle et ses limites ; Conduire une réponse fictive sans accusation

Une détection produit une hypothèse à examiner. Elle doit distinguer faits observés, règle de corrélation et décision humaine. Le laboratoire affiche les événements des deux modes de workflow et signale les événements du contre-exemple. Ce signal décrit un contexte pédagogique volontaire. Il ne démontre aucune attaque contre Vise la Réussite et ne désigne aucun utilisateur réel.

Comprendre ce qui est enregistré

Les événements comportent un index, l'acteur fictif, l'action, la version et le mode. Ils sont en mémoire et disparaissent à la réinitialisation ou au redémarrage. Les refus de transition ne créent pas d'événement dans cette version. Il faut donc conserver leur observation HTTP dans une fiche séparée. Un journal vide ne prouve pas que personne n'a tenté une action.

L'index fournit un ordre local, pas une horloge synchronisée entre machines. Le programme ne collecte ni adresses personnelles ni cookies réels. Lorsqu'une enquête a besoin d'une chronologie distribuée, elle doit prévoir horodatage, provenance et incertitude. Ces éléments ne sont pas ajoutés rétroactivement aux captures du TP.

Écrire une règle testable

La fonction detect retourne les événements du mode contre-exemple comme candidats. Les élèves peuvent proposer une autre règle sur une copie : plusieurs approbations successives d'un même document, décision venant de l'auteur ou transition inattendue. Chaque règle exige une définition, un jeu positif, un jeu négatif et une discussion des faux positifs.

Le seuil « deux événements » choisi pour un exercice n'a aucune validité universelle. Une opération légitime répétée après une coupure peut produire plusieurs observations. Une déduplication bien conçue peut retourner le même résultat sans événement supplémentaire. L'analyste doit donc relier les événements à la sémantique du service et aux identifiants de demande.

Répondre dans le périmètre autorisé

Le scénario fictif commence par une approbation inattendue. Le groupe préserve les fichiers de TP, note la version et décrit les décisions de confinement proposées sur papier. Il ne coupe aucun service réel, ne révoque aucun compte public et ne publie aucune accusation. Les étapes sont adaptées à la mission : qualification, préservation minimale, correction, recette et enseignements.

Un rapport de réponse distingue données absentes et données inexistantes. L'absence de collecte des refus est une limite connue du TP. Elle peut donner lieu à une proposition de journalisation future, avec minimisation des données et contrôle d'accès, mais pas à une preuve inventée.

TP guidé

Méthode pas à pas

Réinitialiser ; produire une soumission légitime puis une approbation corrigée ; lire les événements ; produire un contre-exemple indépendant ; comparer les candidats et le nombre d'événements ; conserver une capture minimale.

Exercice à réaliser

Écrire une règle avec ses cas positifs et négatifs, puis une chronologie du scénario fictif distinguant événements conservés, refus observés et lacunes de collecte. Ajouter un faux positif plausible.

Cas entièrement fictif

Un apprenant observe un refus HTTP mais ne trouve aucune ligne correspondante dans le journal. Il conclut que la requête n'a jamais existé. Le programme ne journalise pourtant que les changements réussis.

Corrigé raisonné

Le refus observé et l'absence de journal sont compatibles avec l'implémentation. Le dossier doit expliquer cette couverture limitée et ne pas attribuer au journal une exhaustivité qu'il ne possède pas.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l'annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Quels événements métier sont enregistrés ici ?

A. Toutes les activités du poste

B. Tous les refus

C. Les changements réussis

2. Que fournit l'index local ?

A. Une horloge distribuée

B. Un ordre local

C. Une identité certaine

3. Un candidat de détection est-il une accusation ?

A. Oui si répété

B. Oui

C. Non

4. Pourquoi tester un faux positif ?

A. Pour préciser la règle

B. Pour éviter les sources

C. Pour supprimer le mandat

5. Que devient le journal après réinitialisation ?

A. Il est archivé durablement

B. Il disparaît avec l'état mémoire

C. Il est envoyé par e-mail

CHAPITRE 13 / 5 h indicatives

Relier droits, groupes et chemins sur Linux

Objectifs : Lire les droits d'un dossier personnel ; Examiner les composants d'un chemin ; Séparer permission système et politique métier

Sur Linux, un accès dépend notamment de l'identité effective, des groupes, des permissions et du chemin parcouru. Le rôle Claire d'Atelier Boréal ne correspond pas à un utilisateur Linux. Le TP conserve ces deux plans séparés afin de ne pas attribuer à chmod une garantie d'autorisation métier.

Observer le contexte sans le modifier

Dans votre dossier de laboratoire, id et ls -ld . preuve-locale.txt permettent d'examiner le contexte et les modes classiques. Un nom de compte ou de groupe réel peut être personnel ; le modèle de rapport remplace ces éléments par des valeurs fictives. Aucune commande sudo n'est nécessaire.

La traversée d'un dossier demande un droit distinct de la lecture de son contenu. Un fichier lisible peut rester inaccessible si un composant du chemin n'est pas traversable. namei -l peut aider lorsque cette commande est disponible ; elle n'est pas installée par le cours. Si elle manque, noter cette limite et examiner les seuls composants autorisés avec ls -ld.

Distinguer ce que le mode n'explique pas

Les bits classiques ne décrivent pas toujours toute la décision : ACL, montage, politiques de sécurité et contexte du processus peuvent intervenir. getfacl, s'il est déjà présent, fournit une autre observation sur le dossier personnel. Son absence ne justifie pas une installation automatique ni une désactivation de politique système.

L'exercice lit et modifie une copie de preuve-locale.txt appartenant à l'apprenant. Il ne demande pas de rendre un dossier mondialement modifiable, de manipuler un binaire privilégié ou d'examiner des répertoires privés d'autres comptes. Un refus doit rester une observation valide.

Examiner le chemin et les liens

Un lien symbolique peut orienter une ouverture vers un autre objet. Le plan d'archive du laboratoire rejette les liens déclarés dans les métadonnées ZIP, mais ne réalise aucune extraction. Il ne démontre donc pas une protection complète contre tous les changements de liens sur un système de fichiers concurrent.

La revue avancée décrit cette différence : une règle sur des chaînes de chemins n'est pas l'équivalent d'une ouverture sûre dans un environnement hostile. La conception d'une vraie extraction exigerait des garanties supplémentaires propres aux API et au périmètre. Le cours s'arrête au plan et à ses limites explicites.

TP guidé

Méthode pas à pas

Relever id et les modes du dossier ; identifier les composants accessibles ; lire une copie du fichier ; comparer ces droits à la politique auteur/approbateur ; annoter les limites du plan d'archive.

Exercice à réaliser

Rédiger une fiche chemin, objet, identité, droit attendu, effet observé et données absentes. Ne modifier aucune permission pour obtenir un résultat spectaculaire.

Cas entièrement fictif

Un plan d’archive refuse ../ puis le rapport annonce une extraction sûre contre tous les liens symboliques. Aucune extraction n’a pourtant eu lieu.

Corrigé raisonné

Conclure sur la classification réalisée et le dossier effectivement examiné. Les fiches Linux sont documentaires dans la livraison initiale, sans exécution Linux native revendiquée.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Quel objet lire avec ls -ld dans le TP ?

A. Un dossier privé de tiers

B. Le dossier personnel du laboratoire

C. Tout le système

2. Le rôle Claire est-il un compte système ?

A. Non, une identité métier fictive

B. Oui, un administrateur

C. Oui, un compte créé

3. Une permission locale remplace-t-elle la politique métier ?

A. Oui toujours

B. Oui si lecture autorisée

C. Non

4. Que faire si une observation est refusée ?

A. Noter la limite sans élargir les protections

B. Désactiver les protections

C. Accorder tous les droits

5. Quelle conclusion vaut pour les droits de Linux ?

A. Elle porte sur les objets et le contexte examinés

B. Elle couvre tous les comptes

C. Elle garantit tous les services

CHAPITRE 14 / 5 h indicatives

Examiner un processus temporaire et une exposition locale

Objectifs : Relier une socket à son propriétaire ; Lire le contexte d’exécution ; Délimiter l’étude documentaire d’un service

Le laboratoire tourne comme un processus temporaire dans le dossier personnel. Il n’ajoute aucune unité systemd et ne modifie aucun démarrage automatique. L’objectif est de relier une observation réseau locale au programme lancé par l’apprenant et à sa durée d’exécution.

Vérifier la résolution du lanceur

command -v python3 et python3 --version indiquent le lanceur résolu et sa version. Un alias, un environnement virtuel ou un PATH différent peut modifier le résultat. Noter ces éléments avant de comparer deux essais. Le cours utilise Python 3.12 ou ultérieur comme environnement recommandé et ne télécharge aucun paquet pour fonctionner.

Après python3 application.py, ss -ltnp peut afficher l’écoute 127.0.0.1:8878. Les informations de processus dépendent des droits et de la disponibilité de ss. Si le propriétaire n’apparaît pas, ne pas utiliser sudo pour contourner cette limite ; relier l’observation au processus lancé dans votre propre terminal et signaler ce que la sortie ne montre pas.

Interpréter l’exposition

Une écoute sur la boucle locale ne crée pas une cible distante publique. Elle ne signifie pas qu’aucun autre processus local ne peut envoyer une requête. Le serveur pédagogique vérifie Host et Origin pour ses mutations et emploie des acteurs fictifs distribués. Ces contrôles de laboratoire ne constituent pas une authentification de production.

Arrêter votre instance avec Ctrl+C et constater la disparition de son écoute. Ne pas tuer un PID trouvé au hasard. Si le port reste présent, rechercher uniquement le processus de votre exercice et documenter la différence, sans intervenir sur un service extérieur au périmètre.

Étudier un service sur papier

pipeline-fictif.md décrit une future livraison. Ajouter sur papier une identité de service limitée, un dossier de travail, des chemins explicites et une procédure d’arrêt. Discuter les droits d’écriture sur les sources et les conditions de redémarrage. Aucune unité n’est installée et aucune protection Linux n’est désactivée.

Une configuration lue dans un fichier n’est pas nécessairement active. Le rapport distingue définition, chargement, exécution et observation. Les états non examinés restent inconnus. Cette prudence est essentielle lorsque plusieurs versions du programme coexistent.

TP guidé

Méthode pas à pas

Relever le lanceur ; démarrer votre instance ; observer l’écoute et le contexte accessible ; arrêter votre processus ; constater la fin ; analyser ensuite l’identité du service fictif sur papier.

Exercice à réaliser

Produire une fiche version, PID connu, adresse, dossier et durée. Ajouter une distinction entre unité proposée et service réellement installé.

Cas entièrement fictif

Une unité systemd rédigée dans le rapport est présentée comme une configuration déjà active. Le TP n’a pourtant créé aucun service.

Corrigé raisonné

Décrire l’exécution temporaire et maintenir la proposition documentaire séparée. La limitation de l’écoute ne vaut pas revue globale de la machine.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Que relier après ss -ltnp ?

A. L’écoute à votre processus

B. Un PID inventé

C. L’écoute à tout le site

2. Quelle écoute attend le laboratoire ?

A. Tous les ports

B. 127.0.0.1:8878

C. Une adresse publique

3. Quel service permanent crée le TP ?

A. Un agent système

B. Aucun

C. Un service de production

4. Quel processus arrêter ?

A. Tout processus sur le port

B. Un service inconnu

C. Votre propre instance identifiée

5. Une proposition de service sur papier est-elle active ?

A. Non

B. Oui après lecture

C. Oui automatiquement

CHAPITRE 15 / 5 h indicatives

Lire une fenêtre de journaux et conserver la provenance

Objectifs : Définir une collecte limitée ; Conserver fuseau et producteur ; Distinguer lacune et absence d’activité

Linux peut héberger plusieurs mécanismes de journalisation. journalctl existe sur les systèmes utilisant le journal systemd ; d’autres installations peuvent employer des fichiers ou un autre service. Le cours ne suppose pas une configuration universelle et ne demande aucun accès privilégié.

Poser une question de collecte

Sur un poste où journalctl est déjà disponible, une commande documentaire borne une fenêtre courte, le nombre de lignes et éventuellement le contexte utilisateur. Les paramètres exacts doivent être adaptés à la version du poste. Lire le message d’erreur lorsqu’aucun journal utilisateur n’est accessible ; ne pas modifier les groupes ou politiques pour élargir l’accès.

La commande date -Iseconds aide à noter une heure avec décalage de fuseau. La fiche doit préciser si la collecte couvre votre application ou seulement un journal système sans événement du TP. Une requête large peut contenir des informations d’autres logiciels : elles ne doivent pas être copiées dans le dossier partagé.

Relier la trace à son producteur

Les événements d’Atelier Boréal proviennent de Workflow.change, restent en mémoire et comportent un index local. Ils ne sont pas automatiquement intégrés à journalctl. Le terminal du serveur affiche des requêtes HTTP, mais n’est pas une preuve d’identité de l’utilisateur. Les acteurs sont des noms fictifs présents dans le scénario.

Une règle de détection doit indiquer la source qu’elle consomme. Une corrélation entre PID, version de fichier et résultat métier nécessite des champs communs ou une observation clairement datée. Le TP ne fournit pas une horloge distribuée ni un stockage inviolable des événements.

Évaluer les faux positifs

Un événement du mode unsafe est attendu dans l’atelier. Il devient un candidat pédagogique, pas une accusation. Une répétition identique de l’approbation corrigée retourne le même résultat et ne crée pas une nouvelle transition. Le nombre de lignes seul peut donc être trompeur.

Les refus ne figurent pas dans les événements métier de cette version. Conserver leur réponse HTTP dans une preuve séparée. Le dossier propose éventuellement une journalisation future avec minimisation, rétention et accès limités, mais ne prétend pas que ces fonctions sont déjà présentes.

TP guidé

Méthode pas à pas

Noter heure et fuseau ; lire la fiche de collecte documentaire ; produire une action autorisée et un refus dans le TP ; relier chacun à sa source ; documenter la trace absente sans l’inventer.

Exercice à réaliser

Construire une chronologie avec producteur, heure connue ou index, observation et limite. Ajouter un faux positif et la vérification qui l'écarte.

Cas entièrement fictif

Le journal système est silencieux sur une approbation. Le rapport déclare cette approbation impossible, alors que l'application n'exporte pas ses événements vers ce journal.

Corrigé raisonné

La couverture des sources commande la conclusion. Une collecte limitée ou absente doit être explicitement présentée comme telle.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l'annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Que préciser avant une collecte ?

A. Une accusation

B. Source, fenêtre et limite

C. Le maximum de données privées

2. Un journal vide prouve-t-il aucune activité ?

A. Oui sans filtre

B. Oui toujours

C. Non

3. Les événements du TP vont-ils automatiquement au journal système ?

A. Non

B. Oui après chaque clic

C. Oui sur tous les OS

4. Que conserve la fiche d'un refus ?

A. Son observation séparée

B. Un événement inventé

C. Un mot de passe

5. Quel terme convient à un acteur du TP ?

A. Utilisateur réel identifié

B. Acteur fictif

C. Suspect certain

CHAPITRE 16 / 5 h indicatives

Restaurer un fichier et délimiter l’état à récupérer

Objectifs : Préserver une copie locale ; Vérifier un hash ; Séparer restauration de code et effet métier

La récupération du TP s’appuie sur des fichiers personnels et un état en mémoire. Elle n’exige ni snapshot de système ni modification du stockage de production. Le chapitre enseigne une procédure vérifiable et ses limites plutôt qu’une promesse de retour arrière universel.

Préserver avant de modifier

Dans le dossier du laboratoire, sha256sum preuve-locale.txt calcule le hash du fichier d’exercice. cp -- preuve-locale.txt preuve-locale.bak crée une copie sur les systèmes compatibles avec cette syntaxe. Lire la commande avant de l’exécuter et vérifier qu’aucun fichier de référence existant ne sera écrasé sans intention.

Modifier uniquement preuve-locale.txt, recalculer son hash, puis restaurer depuis votre copie. Une égalité finale indique le retour aux mêmes octets. Elle ne garantit pas que les permissions, attributs étendus ou autres métadonnées ont été conservés par cette méthode simple. Le rapport borne sa conclusion au contenu étudié.

Examiner la confiance de la référence

Un manifeste non signé copié avec le paquet donne un point de comparaison. Il ne fournit pas une provenance authentifiée à lui seul. Si un attaquant hypothétique modifie le paquet et le manifeste ensemble, une comparaison interne peut rester cohérente. Le TP ne distribue aucune clé de signature privée et ne prétend pas apporter cette garantie.

Pour une vraie chaîne de livraison, une référence indépendante et des contrôles de confiance seraient nécessaires. Le chapitre sur la chaîne logicielle décrit cette proposition sur papier. Il n’installe aucun agent de vérification ni dispositif de déploiement.

Distinguer les états

La réinitialisation du workflow remet les documents fictifs à leur état initial. La transaction SQLite travaille dans un dossier temporaire qui est supprimé après l’essai. Ces comportements sont annoncés afin que l’apprenant ne cherche pas une base persistante inexistante.

Un changement déjà envoyé à un service externe ne serait pas annulé par la restauration d’un fichier Python. Aucun service externe n’est appelé ici. La recette doit néanmoins mentionner cette limite conceptuelle et prévoir la réconciliation avant toute nouvelle mutation lorsque le résultat est incertain.

TP guidé

Méthode pas à pas

Préserver la copie ; noter les hashes ; modifier le seul fichier d’exercice ; restaurer ; vérifier ; exécuter transaction_tp.py ; distinguer ses données temporaires de la mémoire du workflow.

Exercice à réaliser

Livrer une procédure avec précondition, action, preuve de succès et limite. Préciser les métadonnées non examinées et l’absence d’effet externe dans le TP.

Cas entièrement fictif

Un rapport annonce une récupération complète de l'entreprise après restauration de preuve-locale.txt. L'essai porte seulement sur quelques octets dans un dossier personnel.

Corrigé raisonné

Une restauration valide reste proportionnée à son objet. Les garanties de stockage durable et de compensation externe ne sont pas démontrées.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l'annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Que compare sha256sum ?

A. L'état métier

B. Tous les droits

C. Les octets à une référence

2. Quel fichier peut être restauré ?

A. Votre fichier personnel d'exercice

B. Un fichier système

C. Une donnée de tiers

3. Une concordance de hash prouve-t-elle la sûreté ?

A. Oui

B. Non

C. Oui si le fichier est court

4. Quel effet une restauration de fichier n'annule-t-elle pas forcément ?

A. Une décision déjà appliquée

B. Son texte restauré

C. Les octets du fichier

5. Que faire des protections du poste ?

A. Les désactiver pour le TP

B. Les contourner

C. Les conserver

CHAPITRE 17 / 5 h indicatives

Vérifier la correction, les effets partiels et le retour arrière

Objectifs : Définir des critères de recette ; Réconcilier une opération incertaine ; Préparer un retour arrière limité au TP

La correction n’est complète que si elle protège la propriété visée tout en préservant le parcours autorisé. Une restriction trop large peut empêcher toute approbation sans corriger proprement le service. La recette avancée relie politique, état, version, concurrence et reprise après interruption. Elle reste exécutée dans le laboratoire ou sur ses fichiers personnels.

Définir les résultats avant l’essai

Pour le workflow, la recette exige que l’auteur soit refusé, que l’autre organisation soit refusée, que Claire réussisse après la soumission et qu’une répétition identique rende le résultat connu sans nouvelle transition. Une version périmée doit être rejetée. Les critères portent sur le document et les événements, pas uniquement sur le code HTTP.

Pour le budget, le résultat corrigé doit accepter une opération de soixante sur un total de cent et conserver quarante. La table SQLite utilise une condition sur version et solde dans la même transaction que l’écriture du mouvement. Un simple verrou Python ne prouve pas la sûreté d’une architecture distribuée. Le rapport délimite les deux essais séparément.

Traiter une réponse perdue

Le scénario d’une réponse perdue est distinct d’un refus confirmé. Si l’état final est inconnu, relire l’identifiant et la version avant de tenter une nouvelle mutation. Dans le TP, une répétition de la même approbation avec sa même clé fournit un résultat réutilisé. Une demande différente sous la même clé produit un conflit.

Cette garantie disparaît au redémarrage du workflow en mémoire. Une conception durable devrait enregistrer demande, résultat et effet de façon cohérente. Aucun paiement, e-mail ou appel externe n’existe dans l’exercice. Le groupe ne peut donc pas revendiquer une garantie d’exécution unique pour un service externe.

Préparer un retour arrière observable

Le TP possède une commande de réinitialisation volontaire qui remet les deux workflows à l’état initial. Pour les fichiers propres à l’apprenant, la procédure prend une copie avant changement, note les hashes, effectue une modification locale et compare après restauration. Elle ne sauvegarde aucun secret réel et ne touche pas les fichiers système.

Un retour arrière de code ne supprime pas nécessairement un effet métier déjà produit. Le dossier fictif distingue restauration du programme, restauration de données et éventuelle compensation. La compensation n’est pas simulée par une suppression silencieuse d’un événement. Elle doit être explicitement conçue et approuvée dans un système réel.

TP guidé

Méthode pas à pas

Exécuter verifier.py et transaction_tp.py ; relever les résultats avant/après ; réaliser le parcours Claire autorisé ; répéter sa même demande ; réinitialiser ; vérifier l’état initial ; restaurer une copie de preuve-locale.txt.

Exercice à réaliser

Livrer une recette avec critères mesurables, preuve de succès légitime, preuve de refus, scénario de réponse perdue, limite au redémarrage et retour arrière propre au dossier du laboratoire.

Cas entièrement fictif

L'équipe fictive propose de relancer une approbation avec une nouvelle clé après une réponse réseau absente. Sans lecture de l'état, cela ne permet pas de savoir si le premier effet a déjà eu lieu.

Corrigé raisonné

Relire le résultat et l'état sous les mêmes identifiants. Préserver une trace de l'incertitude. Les tests du TP valident une portée locale, sans garantir tous les effets d'une future architecture distribuée.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l'annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Quelle recette préserve le service ?

A. Refus de tous

B. Refus interdits et succès de Claire

C. Suppression des contrôles

2. Que faire d'une mutation ambiguë ?

A. La déclarer absente

B. Changer tous les identifiants

C. Réconcilier avant de retenter

3. Une restauration de code annule-t-elle un effet externe ?

A. Oui si le hash revient

B. Pas nécessairement

C. Oui toujours

4. Quel objet modifier pour le TP de retour arrière ?

A. Le fichier personnel d'exercice

B. Un compte public

C. Un fichier système

5. Quelle portée annoncer pour SQLite ?

A. Toute architecture distribuée

B. Un paiement réel

C. Le test transactionnel temporaire

CHAPITRE 18 / 15 h indicatives

Constituer un dossier avancé défendable et préparer l’évaluation

Objectifs : Produire quatre constats argumentés ; Écarter un faux positif ; Délimiter la portée du diplôme interne

Le projet final rassemble les propriétés étudiées dans une mission fictive d’Atelier Boréal. Le livrable n’est pas une collection de captures sans commentaire. Chaque constat présente une question, des préconditions, un résultat, un impact limité, une correction et une recette. Le lecteur doit pouvoir distinguer ce qui a été exécuté de ce qui a été proposé sur papier.

Organiser un dossier indépendant des outils

Commencer par le mandat, la version des fichiers, le schéma des frontières et une matrice de couverture. Ajouter quatre constats issus du laboratoire : séparation des responsabilités, transition sur version périmée, concurrence et plan d’archive. Un autre choix est possible si les preuves et les limites sont aussi explicites. Tous les noms, documents et budgets du scénario sont fictifs.

Pour chaque constat, conserver une preuve minimale. Le montant de soixante n’est pas un chiffre financier réel ; il permet de calculer un invariant. Une capture du plan d’archive ne démontre aucune écriture hors dossier : l’exercice n’extrait rien. Le rapport doit maintenir cette précision jusque dans son titre et sa conclusion.

Démontrer un faux positif

Choisir un signal qui pourrait être mal interprété. La clé HMAC publique n’est pas une fuite d’une clé de production puisqu’elle est distribuée pour le TP. Un composant fictif inventorié n’est pas un avis de sécurité sur un fournisseur réel. Un résultat unsafe n’indique pas que le mode corrigé possède le même chemin.

Le faux positif est écarté par une vérification, pas par une préférence. Indiquer le fichier ou le test qui contredit l’hypothèse et ce qui reste inconnu. La capacité à réduire une conclusion fait partie de la qualité du travail d’audit.

Réviser les limites et la présentation

Relire les formulations causales : le test produit-il effectivement l’effet annoncé ? Vérifier les versions, la lisibilité des captures, les liens des sources primaires et l’absence de données personnelles. Les commandes Windows et Linux sont des fiches documentaires lorsqu’aucune exécution native n’a été effectuée. Le laboratoire portable a été vérifié sur macOS ; cette observation ne vaut pas validation de tous les postes des apprenants.

Le parcours propose cent heures de travail indicatives, comprenant dix-sept chapitres de cinq heures et quinze heures de projet. Ce volume inclut lecture, manipulations et révision. Il ne correspond ni à la durée des vidéos ni à une présence surveillée. La grille fournit une autoévaluation ; aucun correcteur humain n’est promis.

Comprendre les conditions de réussite

Les dix-huit QCM de chapitre demandent quatre bonnes réponses sur cinq. L’examen final devient accessible lorsque ces QCM sont validés. Il contient quarante questions et exige trente-deux bonnes réponses. La plateforme peut alors délivrer un diplôme de réussite interne VLR gratuit. Ce questionnaire non surveillé ne constitue ni habilitation professionnelle ni diplôme universitaire reconnu.

TP guidé

Méthode pas à pas

Utiliser le modèle de rapport et la grille ; vérifier quatre constats avec leurs contre-tests ; écarter un faux positif ; distinguer preuves exécutées et propositions ; relire les limites avant de passer les QCM.

Exercice à réaliser

Remettre votre propre dossier : mandat, architecture, matrice, quatre constats, faux positif, recette, retour arrière, provenance et synthèse. Évaluer chaque critère de la grille avec un lien vers une preuve réelle du TP.

Cas entièrement fictif

Un dossier annonce une extraction arbitraire et une fraude de production alors qu'il ne contient qu'un plan d'archive en mémoire et un budget fictif. Il doit être réécrit pour correspondre aux effets réellement observés.

Corrigé raisonné

Un dossier satisfaisant reste reproductible, proportionné et honnête sur ses limites. Le projet est autoévalué ; les QCM déterminent la réussite au questionnaire et l'émission du diplôme interne selon les règles de la plateforme.

QCM de chapitre : entraînement

Répondez avant de consulter les corrigés de l’annexe. Pour valider la progression, passez ce QCM sur le site. Cinq questions ; seuil quatre bonnes réponses.

1. Combien de constats demande le dossier avancé ?

A. Aucun

B. Un slogan

C. Quatre et un faux positif écarté

2. Quelle condition ouvre l’examen ?

A. Le paiement

B. Les dix-huit QCM validés

C. Le seul téléchargement du PDF

3. Quel seuil final est requis ?

A. 32 sur 40

B. Une seule réponse

C. 20 sur 40

4. Que représente le diplôme ?

A. Une habilitation de test de tiers

B. Une réussite interne VLR non surveillée

C. Un diplôme universitaire reconnu

5. Qui corrige le projet proposé ?

A. Une université

B. L’apprenant avec la grille

C. Un jury promis

Dossier pratique et commandes

Télécharger les scripts, données fictives et modèles du TP:
https://viselareussite.com/images/academy/hacking-ethique-avance-laboratoire/2026.10.6.1/laboratoire.zip

Atelier Boréal : guide complet du laboratoire local

Périmètre et prérequis

Python 3.12 ou ultérieur, navigateur récent, dossier personnel. Aucun paquet externe, compte administrateur ou service payant. Les données sont fictives. Ne tester aucune cible tierce. Aucun service externe, e-mail ou paiement n'est appelé.

Démarrage

Lire application.py, controles.py et verifier.py. Ouvrir un terminal dans ce dossier puis lancer :

```
python3 application.py
```

Sur Windows, employer le lanceur disponible et vérifié, par exemple :

```
py -3 application.py
```

Ouvrir http://127.0.0.1:8878 dans le navigateur. Ne changer aucune adresse d'écoute. Si le port appartient à un autre programme, ne pas le tuer. Arrêter votre propre instance avec Ctrl+C.

Politique et transitions

Alice est l'auteur du document 301 de l'organisation A. Claire est reviewer de A. Bob est reviewer de B. Ces noms ne sont pas des comptes authentifiés. Le mode contre-exemple et le mode corrigé ont deux états indépendants.

1. Réinitialiser le TP.

2. Essayer l'approbation de l'auteur dans le contre-exemple : succès pédagogique.

3. Essayer la même action corrigée : refus 403 et état inchangé.

4. Soumettre avec Alice : succès, version 2.

5. Approuver avec Claire : succès, version 3.

6. Répéter la même approbation : résultat réutilisé sans nouvelle transition.

7. Lire les événements ; les refus ne sont pas enregistrés dans ce journal métier.

Concurrence et transaction

Comparer les deux boutons de budget. Deux décisions de 60 sur un solde de 100 dans le contre-exemple peuvent être acceptées avec un solde final de 40 ; l'invariant calculé est faux. La section critique corrigée accepte une seule décision et conserve 40. Elle est limitée à un processus. Exécuter aussi le cas transactionnel :

```
python3 transaction_tp.py
```

Il crée une base SQLite dans son propre dossier temporaire, teste une mise à jour conditionnelle et un mouvement dans la même transaction, puis nettoie ce dossier. Résultat attendu : un succès, 40 restants, une ligne de mouvement. Aucun argent réel.

Archives et destinations

Comparer le plan sans contrôle et le plan corrigé. L'inspection ZIP lit des métadonnées en mémoire : nombre, tailles, ratio, chiffrement et liens déclarés. Aucune extraction ni écriture dans un chemin d'archive. Le classificateur d'URL compare uniquement une destination HTTP locale exacte. outgoingRequests reste à zéro ; résolution DNS et redirections sont étudiées sur papier.

Intégrité et journal

Le bouton intégrité compare un objet original à un montant modifié. La clé HMAC est publique, volontairement pédagogique. Cette démonstration ne chiffre rien, n'authentifie pas une personne et ne protège pas seule contre le rejeu. Le journal est en mémoire ; un index n'est pas une horloge distribuée.

Vérification automatique

```
python3 verifier.py
```

Dix tests couvrent politique, transitions, déduplication, contre-exemple, concurrence, SQLite, archives, URL, intégrité et routes HTTP. Ils ouvrent un serveur local temporaire et le ferment. resultats-http.json ne contient que des données fictives.

Dossier et limites

Utiliser mandat.md, rapport-modele.md et grille-projet.md. Quatre constats, un faux positif, contrôles légitimes, recette et limites. Les captures livrées sont réelles sur macOS, avec données fictives. Les commandes Windows/Linux sont documentaires. Un TP réussi ne vaut ni audit d'un site réel ni habilitation à tester un tiers.

Fiches de commandes : lire avant de lancer

Exécuter dans votre dossier personnel seulement. Remplacer tout PID par celui effectivement observé. Les sorties contiennent parfois des données personnelles : ne pas les publier. Aucun privilège élevé ni changement de protection.

Windows : PowerShell, fiche documentaire

```
$PSVersionTable.PSVersion
whoami
Get-Acl -LiteralPath .
Get-Command py
py -3 --version
Get-NetTCPConnection -LocalPort 8878 -State Listen
Get-FileHash -LiteralPath .\preuve-locale.txt -Algorithm SHA256
```

Après attribution du PID, Get-Process -Id permet de lire votre processus. La lecture éventuelle de journaux doit être bornée à une source autorisée ; aucune collecte native Windows n'a été effectuée pour cette livraison. Get-AuthenticodeSignature sur un fichier compatible ne remplace pas une analyse de sûreté.

Linux : shell, fiche documentaire

```
id
ls -ld . preuve-locale.txt
command -v python3
python3 --version
ss -ltnp
sha256sum preuve-locale.txt
```

ss et getfacl peuvent être absents. Ne pas installer ni utiliser sudo pour compléter le TP. Une collecte journalctl nécessite une source, une fenêtre et une limite ; aucune collecte native Linux n'a été effectuée. Ne créer aucune unité systemd.

macOS : observations locales et limites

```
id
ls -lde . preuve-locale.txt
ls -l@ preuve-locale.txt
command -v python3
python3 --version
lsof -nP -iTCP:8878 -sTCP:LISTEN
shasum -a 256 preuve-locale.txt
```

L'écoute attendue est 127.0.0.1:8878. Les journaux privés du poste ne sont pas collectés. codesign et spctl sont présentés comme outils documentaires ; ne contourner aucune protection et ne créer aucun LaunchAgent.

Copie et restauration du fichier d'exercice

Vérifier la destination avant copie. Conserver un hash avant modification, modifier seulement preuve-locale.txt puis restaurer votre copie et recalculer le hash. Une égalité finale concerne les octets, pas toutes les métadonnées ni les décisions métier.

```
cp preuve-locale.txt preuve-locale.bak
cp preuve-locale.bak preuve-locale.txt
```

Sur Windows, utiliser Copy-Item avec les chemins personnels correspondants. Le rapport doit indiquer les commandes réellement exécutées ; ne recopier aucune sortie d'exemple comme observation.

Mandat fictif Atelier Boréal

Responsable et apprenant : valeurs fictives à compléter. Version du paquet et date de votre essai : à relever. Périmètre : scripts distribués, fichiers personnels du TP, http://127.0.0.1:8878. Exclusions : site public, comptes réels, tiers, réseau externe, modifications système et protections.

Propriétés : séparation auteur/approbateur, organisation, transitions et versions, budget cohérent, déduplication, plan d'archive borné, classification hors réseau, intégrité pédagogique. Les états sont en mémoire ; aucune extraction ou opération externe.

Arrêt : dépassement de périmètre, données personnelles imprévues, processus inconnu, résultat de mutation ambigu. Préserver la preuve minimale et réconcilier. Aucun partage de secret. Livrables : quatre constats, faux positif, recette et limites.

Rapport professionnel à compléter

Synthèse et périmètre

Question, autorisation, dates, version, systèmes effectivement exécutés. Distinguer résultats, hypothèses et propositions.

Architecture et couverture

Entrées, données de décision, frontières, stockage, événements, effets. Tableau propriété, cas légitime, cas interdit, preuve et limite. Indiquer les chemins non testés.

Fiche de constat à répéter quatre fois

Titre proportionné. Préconditions. Procédure locale. État avant et après. Résultat attendu et observé. Preuve minimale. Effet et invariant. Portée. Correction proposée. Contre-test. Recette. Incertitude et retour arrière.

Faux positif écarté

Hypothèse, vérification contradictoire, preuve et information encore absente. Ne qualifier aucun fournisseur réel à partir de l'inventaire fictif.

Reprise et livraison

Même identifiant après résultat ambigu. Limite de mémoire volatile. Référence de fichiers et hash. Pipeline sur papier non déployé. Distinction entre restauration de code, données et effet externe.

Conclusion

Ce qui a été exécuté, ce qui est documentaire et ce qui est inconnu. Autoévaluation avec la grille. Aucun audit global ou diplôme externe revendiqué.

Grille d'autoévaluation du projet avancé

Aucun correcteur humain promis. Le projet n'attribue pas le diplôme à lui seul. Pour chaque critère : absent, partiel ou étayé, avec lien vers votre preuve.

1. Mandat borné et condition d'arrêt.

2. Architecture et données de décision.

3. Quatre constats reproductibles et proportionnés.

4. Contrôles légitimes préservés et refus sans mutation.

5. Calcul indépendant d'un invariant.

6. Faux positif écarté par vérification.

7. Réconciliation sous les mêmes identifiants.

8. Recette et restauration personnelle vérifiée.

9. Provenance, versions et limites documentaires.

10. Présentation lisible et données minimisées.

Réviser les critères absents. Les dix-huit QCM de chapitre sont validés à 4/5 ; l'examen final de quarante questions à 32/40 ouvre le diplôme de réussite interne VLR non surveillé.

Lecture du code fourni

Ces extraits sont tirés du laboratoire distribué. Les fonctions complètes, le serveur et les tests se trouvent dans l’archive ZIP. Lisez les préconditions et utilisez uniquement votre copie locale.

Fonction destination_allowed

```
def destination_allowed(raw):
 if not isinstance(raw,str)or any(ord(c)<33 for c in raw)or '\\'in raw:return False
 try:
  u=urlsplit(raw)
  return u.scheme=='http'and u.hostname=='127.0.0.1'and u.port==8878 and u.username
is None and u.password is None and u.path=='/api/health'and not u.query and not u.
fragment
 except ValueError:return False
```

Fonction canonical

```
def canonical(payload):return json.dumps(payload,sort_keys=True,separators=(',',':'),
ensure_ascii=False).encode('utf-8')
```

Corrigés des QCM de chapitre

Ces corrigés servent à la révision. Le questionnaire final est passé dans la plateforme après les dix-huit validations de chapitre. Le score du projet ne remplace pas le QCM final.

Chapitre 01

1. Réponse A : Une propriété et des critères observables. Le mandat relie une propriété à des preuves et à des limites.
2. Réponse C : Tester le laboratoire local fictif. Le périmètre est votre laboratoire et ses données fictives.
3. Réponse B : Elle précise la version et le périmètre. La preuve ne dépasse pas l'environnement observé.
4. Réponse C : Une identité fictive du TP. La preuve doit minimiser les données.
5. Réponse C : Arrêter et clarifier le périmètre. L'autorisation doit couvrir l'action.

Chapitre 02

1. Réponse A : La source fiable des données de décision. Une décision dépend de données et de frontières de confiance.
2. Réponse A : Une politique sur des identités fictives. Le TP ne fournit pas de service d'identité réel.
3. Réponse A : Pour délimiter la couverture. Une absence de modèle limite la conclusion.
4. Réponse A : Avant l'effet qu'il protège. Le contrôle doit empêcher l'effet interdit.
5. Réponse C : Le marquer comme inconnu. La couverture repose sur des preuves.

Chapitre 03

1. Réponse C : Claire, approbatrice de la même organisation. La politique combine rôle, organisation et séparation.
2. Réponse C : Non, organisation et auteur comptent aussi. La politique comporte plusieurs conditions.
3. Réponse C : L'état et la version inchangés. Un refus doit empêcher la mutation.
4. Réponse C : Une approbation légitime réussie. Une correction doit préserver l'usage autorisé.
5. Réponse C : Un contre-exemple pédagogique. Les deux modes appartiennent au TP fictif.

Chapitre 04

1. Réponse C : Draft vers submitted par l'auteur. Le workflow impose des transitions explicites.
2. Réponse A : Refuser sans mutation. La version protège la décision sur l'état attendu.
3. Réponse C : Pour relier la décision à l'objet attendu. Les conditions portent sur la même opération.
4. Réponse A : Un entier métier, pas un booléen. Le TP exclut le booléen malgré son rapport aux entiers Python.
5. Réponse A : Le document et les événements. Le statut seul ne décrit pas tout l'effet.

Chapitre 05

1. Réponse B : L'invariant des mouvements est violé. Le solde attendu après deux succès serait -20.
2. Réponse C : Un succès et un solde de 40. La section critique évite les deux décisions sur le même solde.
3. Réponse B : Une portée dans son processus. La portée du verrou est locale.
4. Réponse B : Une mise à jour conditionnelle et un mouvement cohérents. SQLite est utilisé dans un dossier temporaire.
5. Réponse C : Pour rendre le chevauchement observable. La barrière synchronise les deux lectures du contre-exemple.

Chapitre 06

1. Réponse B : Résultat réutilisé sans transition supplémentaire. Le TP conserve le résultat de la demande.
2. Réponse C : Conflit. La clé est associée au contenu précis.
3. Réponse C : Non. Elle identifie une demande, sans authentifier l'acteur.
4. Réponse A : Elle disparaît dans ce TP. Le TP ne stocke pas durablement le workflow.
5. Réponse A : Relire sous les mêmes identifiants. Une réponse absente peut cacher un effet déjà appliqué.

Chapitre 07

1. Réponse A : Non, il calcule un plan. Aucune écriture d'extraction n'est réalisée.
2. Réponse B : Rejeter selon les règles du plan. Le plan rejette les segments de traversée.
3. Réponse A : Nombre, tailles et ratio de compression. Les métadonnées sont bornées sans extraction.
4. Réponse B : Deux noms visant le même nom prévu. Le plan doit éviter une destination ambiguë.
5. Réponse B : Non, une vraie extraction exige d'autres garanties. Le TP délimite sa propriété à la classification.

Chapitre 08

1. Réponse C : HTTP exact vers 127.0.0.1:8878/api/health. La liste du TP est fermée et exacte.
2. Réponse A : Non. Le TP est une classification hors réseau.
3. Réponse B : Le refuser dans ce TP. La syntaxe est analysée avant la comparaison.
4. Réponse C : Un futur client pourrait changer de destination. Le cas réseau réel reste étudié sur papier.
5. Réponse C : Non. La comparaison porte sur les composants analysés.

Chapitre 09

1. Réponse A : La modification du montant ne correspond plus au tag. Le HMAC porte sur l'intégrité sous la clé utilisée.
2. Réponse A : Ils peuvent calculer un tag. La clé du TP est volontairement publique.
3. Réponse B : Rien après canonicalisation prévue. Les clés sont triées pour le calcul.
4. Réponse B : Non. Le protocole doit traiter répétition et contexte.
5. Réponse C : Prévoir une gestion de clés adaptée en production. Le cours ne crée aucun secret de production.

Chapitre 10

1. Réponse B : Le chemin jusqu'à l'effet. Un contrôle doit être relié à son appel effectif.
2. Réponse C : Succès légitime, refus et état final. Les tests doivent pouvoir contredire l'hypothèse.
3. Réponse A : Sur une copie personnelle du TP. Le périmètre reste local.
4. Réponse C : Non, un candidat à vérifier. La condition et le chemin doivent être examinés.
5. Réponse A : Pour éviter de partager l'erreur d'implémentation. L'assertion métier doit garder son indépendance.

Chapitre 11

1. Réponse C : La concordance avec une référence. Un hash compare des octets.
2. Réponse A : Une cohérence interne limitée. La confiance dans la référence reste à traiter.
3. Réponse C : Un avis entièrement fictif. L'exercice ne vise aucun fournisseur réel.
4. Réponse C : Non. L'usage effectif demande une preuve.
5. Réponse B : Non. Le cours propose des correspondances partielles datées.

Chapitre 12

1. Réponse C : Les changements réussis. La couverture est limitée.
2. Réponse B : Un ordre local. Le TP ne synchronise pas de machines.
3. Réponse C : Non. La qualification exige des preuves.
4. Réponse A : Pour préciser la règle. Une règle doit distinguer les situations légitimes.
5. Réponse B : Il disparaît avec l'état mémoire. Le TP n'a pas de stockage durable du workflow.

Chapitre 13

1. Réponse B : Le dossier personnel du laboratoire. Le périmètre reste personnel.
2. Réponse A : Non, une identité métier fictive. Aucun compte système n'est créé.
3. Réponse C : Non. Les objets et décisions sont distincts.
4. Réponse A : Noter la limite sans élargir les protections. Un refus reste une preuve de couverture limitée.
5. Réponse A : Elle porte sur les objets et le contexte examinés. La portée doit correspondre à l'observation.

Chapitre 14

1. Réponse A : L'écoute à votre processus. Le PID doit être réellement observé.
2. Réponse B : 127.0.0.1:8878. La cible est locale.
3. Réponse B : Aucun. Le processus est temporaire.
4. Réponse C : Votre propre instance identifiée. L'action doit rester autorisée.
5. Réponse A : Non. Conception et exécution sont distinctes.

Chapitre 15

1. Réponse B : Source, fenêtre et limite. La collecte répond à une question bornée.
2. Réponse C : Non. La couverture du producteur compte.
3. Réponse A : Non. Le workflow écrit en mémoire.
4. Réponse A : Son observation séparée. Les refus ne sont pas journalisés par ce workflow.
5. Réponse B : Acteur fictif. Le scénario ne prouve pas une identité humaine.

Chapitre 16

1. Réponse C : Les octets à une référence. Un hash a une portée sur le contenu.
2. Réponse A : Votre fichier personnel d'exercice. La restauration est bornée.
3. Réponse B : Non. Intégrité de référence et sûreté sont distinctes.
4. Réponse A : Une décision déjà appliquée. Un effet métier a son propre état.
5. Réponse C : Les conserver. Le cours ne demande aucun affaiblissement.

Chapitre 17

1. Réponse B : Refus interdits et succès de Claire. La correction doit conserver le parcours légitime.
2. Réponse C : Réconcilier avant de retenter. Une réponse perdue ne décrit pas l'effet.
3. Réponse B : Pas nécessairement. Les effets ont leur propre état.
4. Réponse A : Le fichier personnel d'exercice. La restauration est bornée au laboratoire.
5. Réponse C : Le test transactionnel temporaire. La recette précise le dispositif effectivement exécuté.

Chapitre 18

1. Réponse C : Quatre et un faux positif écarté. Le dossier relie constats, preuves et recette.
2. Réponse B : Les dix-huit QCM validés. Chaque chapitre demande quatre réponses correctes sur cinq.
3. Réponse A : 32 sur 40. Le seuil est de 80 pour cent.
4. Réponse B : Une réussite interne VLR non surveillée. Le cours ne revendique aucune reconnaissance externe.
5. Réponse B : L’apprenant avec la grille. Le projet est explicitement autoévalué.

Sources et provenance

Les leçons et scénarios sont originaux. Les références ci-dessous soutiennent les concepts et les observations documentaires. L'avis BOREAL-DEMO-01 et les composants boreal-demo sont entièrement fictifs. Les sources ont été consultées le 6 octobre 2026. Les commandes natives Windows et Linux restent documentaires ; le laboratoire Python a été testé sur macOS.

OWASP ASVS 5.0.0 : exigences versionnées: https://owasp.org/projects/asvs

OWASP : autorisation de transaction: https://cheatsheetseries.owasp.org/cheatsheets/Transaction_Authorization_Cheat_Sheet.html

OWASP : prévention SSRF: https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html

NIST SSDF 1.1: https://csrc.nist.gov/pubs/sp/800/218/final

Python : ZIP: https://docs.python.org/3/library/zipfile.html

Python : threads: https://docs.python.org/3/library/threading.html

Python : HMAC: https://docs.python.org/3/library/hmac.html

Python : SQLite: https://docs.python.org/3/library/sqlite3.html

OWASP : journaux: https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

OWASP : chaîne logicielle: https://cheatsheetseries.owasp.org/cheatsheets/Software_Supply_Chain_Security_Cheat_Sheet.html

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

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

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

Apple : sécurité de plateforme: https://support.apple.com/guide/security/welcome/web

Cours sur Vise la Réussite: https://viselareussite.com/formations/hacking-ethique-avance-linux/

Vidéos : supports originaux avec narration synthétique française ; deux voix au choix et sous-titres avec découpage estimé. Captures : interface locale réelle, données fictives, macOS. Schémas et couvertures : créations originales sans carte.