OSL – Révision examen 2026

Ce que l'examen attend

Cette fiche couvre les points 1 à 5 et 14 des modalités (VM, processus, comptes et droits, ligne de commande, installation de logiciels, compléments OS), côté Windows puis côté Linux. Les points 6 à 13 (MiniShop : HTML/CSS, JavaScript, PHP/MySQL, sessions, HTTPS, injection SQL) ne sont pas traités ici.

Chaque point des modalités a deux volets : un **savoir** (expliquer, avec le bon vocabulaire) et un **savoir-faire** (refaire la manipulation sur la VM). Les réponses ci-dessous suivent la fiche corrective du formateur quand elle contredit un rendu de TP.

Les six réflexes que la fiche corrective réclame partout :

1. Répondre à **toutes** les parties de la question (qui ? pourquoi ? comparaison ?).
2. Nommer le responsable exact : le noyau, TrustedInstaller, l'hyperviseur, polkit, l'UAC.
3. Écrire ce qu'on a **observé** sur sa machine (valeur, message exact), pas la procédure.
4. Adapter la réponse à son environnement : VM distante sur ESXi, Workstation local, VirtualBox ; Ubuntu ou Debian.
5. Garder des réponses cohérentes entre elles (par exemple droits de System32 en graphique et avec `icacls`).
6. Utiliser le vocabulaire du bon système : « signal » est un mot Linux, Windows « termine » un processus.

Notions communes aux deux systèmes (point 1 des modalités)

Notion	À savoir dire
Hyperviseur de type 1	Installé directement sur le matériel (ESXi). C'est celui du labo.
Hyperviseur de type 2	Installé sur un OS hôte (Workstation ou VirtualBox avec une VM locale).
vSphere Client	Interface d'administration, pas un hyperviseur.
Workstation en « Connect to Server »	Simple client qui affiche la console ; la VM tourne sur le serveur.
Rôle du PC de l'apprenant	Affiche la console, envoie clavier et souris. Hôte seulement si la VM est locale.
Isolation	L'hyperviseur cloisonne : la VM et le PC ne voient pas les processus l'un de l'autre.
Matériel vu par l'OS	Matériel virtuel (vCPU, RAM) décidé par la personne qui a configuré la VM.

Arrêt propre (Shut Down Guest)	Demande à l'OS de s'éteindre, via VMware Tools. Risque faible.
Coupure brutale (Power Off)	Équivaut à débrancher la prise : données non enregistrées perdues, système de fichiers à risque.
Suspension (Suspend)	Met en pause et enregistre la RAM sur le stockage du serveur (.vmss, .vmem). Tout revient à la reprise.
Snapshot	Point de retour sur l'état de la VM. **Pas une sauvegarde** : il dépend du disque de la VM, grossit et ralentit la VM s'il traîne.
VMware Tools	Pilotes et service dans l'invité : arrêt propre, IP visible dans vSphere, résolution automatique, presse-papier.

À distinguer : Suspend reprend l'exécution là où elle était ; un snapshot permet de revenir à un état antérieur, autant de fois qu'on veut.

Résumé Windows 10

Windows protège ses fichiers par des ACL NTFS et un propriétaire spécial (TrustedInstaller), et ne donne les droits d'administrateur qu'après élévation par l'UAC.

1. Processus et services

- **Processus** : programme en cours d'exécution. **PID** : identifiant d'une *instance* de processus ; un programme relancé reçoit un nouveau PID. **Thread** : unité d'exécution à l'intérieur d'un processus.
- **Service** : programme géré par Windows, en arrière-plan, sans interface. Le processus est son exécution. Le Gestionnaire des tâches compte des processus, pas des services.
- **svchost.exe** : hôte générique qui héberge des groupes de services (60 à 70 instances). Deux raisons : économiser les ressources et isoler les pannes. `tasklist /svc` montre les services derrière chaque PID.
- **explorer.exe** : le shell graphique (bureau, barre des tâches, menu Démarrer, Explorateur). Le terminer fait disparaître tout cela, mais Windows continue de tourner. Relance : Gestionnaire des tâches, Fichier, Exécuter une nouvelle tâche, `explorer.exe`.
- **Taskmgr.exe** : le Gestionnaire des tâches lui-même ; le terminer ferme sa fenêtre.
- **Processus critiques** (`wininit.exe`, `csrss.exe`) : les arrêter plante le système (écran bleu).
- **Fin de tâche** : Windows termine le processus immédiatement, sans sauvegarde ni nettoyage (pas de « signal » comme sous Linux).

Compte	Rôle
SYSTEM	Compte système aux privilèges très étendus, supérieurs à ceux d'un administrateur ; utilisé par de nombreux services.

LOCAL SERVICE	Privilèges réduits ; se présente en anonyme sur le réseau.
NETWORK SERVICE	Privilèges réduits ; se présente sur le réseau avec l'identité de l'ordinateur.
DWM-1, UMFD-0	Comptes virtuels du gestionnaire de fenêtres et des pilotes d'affichage.

Savoir-faire : Ctrl + Maj + Échap, onglet Détails, clic droit sur les en-têtes, Sélectionner les colonnes (PID, Nom d'utilisateur, Processeur, Mémoire, Threads, Priorité de base). Cliquer sur l'en-tête CPU ou Mémoire pour trier ; le plus gros consommateur change dans le temps.

2. Comptes, droits et administration

- **Qui refuse ?** TrustedInstaller, propriétaire de `C:\Windows\System32`, de `Boot` et de `chkdsk.exe`. Être administrateur ne suffit pas.
- **Droits sur System32** : TrustedInstaller a le contrôle total ; Administrateurs et Utilisateurs ont Lecture et exécution (RX), ce que confirme `icacls`.
- **Contourner** (zone rouge) : `takeown /f <fichier>` puis `icacls <fichier> /grant Administrateurs:F`. Le droit par défaut n'est donc pas un droit absolu.
- **ACL** : liste qui dit, pour chaque fichier ou dossier, qui a quels droits. `(F)` contrôle total, `(M)` modification (tout sauf changer les permissions ou prendre possession), `(RX)` lecture et exécution.
- **Éléments masqués** : `C:\ProgramData` (propriétaire SYSTEM), `AppData` dans le profil. Masquer évite les erreurs de manipulation ; ce n'est pas une protection, une case à cocher les affiche.
- **Compte standard / administrateur** : l'administrateur appartient au groupe Administrateurs. Mais appartenir au groupe n'active pas les droits.
- **UAC (User Account Control)** : sépare le jeton standard du jeton élevé. Le bouclier signale une action qui demande une élévation. Compte standard : l'UAC demande les identifiants d'un administrateur. Administrateur : simple Oui/Non.
- **Moindre privilège** : ne donner que les droits nécessaires, et les retirer après usage.

Contexte (`whoami /groups`)	**Groupe Administrateurs**	**Niveau d'intégrité**
Compte standard (etudiant)	absent	Moyen
Administrateur, cmd normal	présent mais « refus uniquement »	Moyen
Administrateur, cmd élevé	actif	Élevé

Dossiers personnels :

- Le profil `C:\Users\<nom>` est créé à la **première ouverture de session**, pas à la création du compte.
- Un utilisateur standard n'a aucun droit sur le profil d'un autre. Le propriétaire peut toujours modifier l'ACL de son dossier.
- Avec Lecture et exécution accordé au groupe Utilisateurs, bob peut lire mais pas modifier.

- Après `net user bob /delete`, le dossier reste : Windows supprime l'identité (compte, SID), pas les données. Les ACL pointent vers un SID orphelin ; un nouveau compte « bob » reçoit un autre SID et n'a pas accès aux anciens fichiers.
- Après ajout au groupe Administrateurs, il faut fermer et rouvrir la session (nouveau jeton). Un cmd non élevé répond encore « Accès refusé ».

3. Ligne de commande : cmd et PowerShell

- **cmd** : shell historique, manipule du texte. **PowerShell** : shell moderne, le tube transmet des **objets** avec leurs propriétés.
- Intérêt de la ligne de commande : automatisation, scripts, informations que l'interface masque.
- `ls`, `pwd`, `cat` dans PowerShell sont des alias vers `Get-ChildItem`, `Get-Location`, `Get-Content`, ajoutés pour les habitués d'Unix. Ce n'est pas le `ls` de Linux : `ls -l` ne fonctionne pas.
- L'invite démarre dans le dossier personnel. `dir` et l'Explorateur masquent tous deux les éléments cachés par défaut.

Objectif	cmd	PowerShell
Identité et groupes	`whoami`, `whoami /groups`	idem
Lister, cachés compris	`dir /a`	`Get-ChildItem -Force`
Seulement les cachés	`dir /a:h`	`Get-ChildItem -Force -Hidden`
Propriétaire	`dir /q`	`(Get-Acl .).Owner`
Droits (ACL)	`icacls .`	`Get-Acl . | Format-List`
Lister les processus	`tasklist`, `tasklist /svc`	`Get-Process`
Chercher un processus	`tasklist | find "notepad"`	
Terminer un processus	`taskkill /PID <n>` (demande), `taskkill /PID <n> /F` (forcé, comme Fin de tâche)	
Créer un compte	`net user etudiant Etudiant#2026 /add`	
Supprimer un compte	`net user bob /delete`	

Ajouter aux administrateurs	`net localgroup Administrateurs etudiant /add`
Retirer	`net localgroup Administrateurs etudiant /delete`
Version, matériel	`winver`, `systeminfo`

À la racine, `dir /a` révèle `$Recycle.Bin`, `pagefile.sys`, `System Volume Information`, `Recovery`, `ProgramData`.

4. Installation de logiciels

Méthode	Qui vérifie ?	Risque
Microsoft Store	Contrôles de certification de Microsoft ; source la plus encadrée	Le plus faible
winget	Manifestes validés, hash de l'installateur vérifié ; moins contrôlé que le Store	Faible à moyen
.exe téléchargé sur un site	Seulement l'utilisateur	Le plus élevé

- Commandes : `winget search 7zip`, `winget install --id 7zip.7zip -e`, `winget list`, `winget uninstall <Id>`. L'**Id** identifie un paquet de façon unique, le nom peut être ambigu ; `-e` impose la correspondance exacte.
- **Droits** : 7-Zip va dans `C:\Program Files`, installation pour toute la machine, donc élévation UAC. Un logiciel installé dans le profil (user scope) n'en demande pas.
- **Empreinte SHA-256** (`Get-FileHash .\setup.exe -Algorithm SHA256`) : intégrité, seulement si on la compare à une référence obtenue par un canal fiable.
- **Signature** (`Get-AuthenticodeSignature .\setup.exe`, champs Status et SignerCertificate) : authenticité de l'éditeur. « Valid » veut dire non modifié depuis la signature, pas inoffensif.
- **Deux installations en même temps** : Windows Installer refuse, erreur 1618 « Une autre installation est déjà en cours ». Équivalent du verrou dpkg.
- **Chaîne d'approvisionnement** : un logiciel piégé à la source arrive par un canal de confiance ; d'où le contrôle des sources autorisées en entreprise.

5. Cycle de vie de la VM vu depuis Windows

- Observateur d'événements (`eventvwr.msc`, Journaux Windows, Système) : **6008** arrêt précédent inattendu, **41** Kernel-Power (redémarrage sans arrêt correct), **1074** arrêt ou redémarrage demandé par un utilisateur ou un processus.
- **Démarrage rapide** : « Arrêter » conserve une partie du noyau dans `hiberfil.sys`, la Durée d'activité n'est pas forcément remise à zéro. « Redémarrer » recharge tout.

- Presse-papier et glisser-déposer sont des canaux de fuite de données, souvent désactivés. Transferts contrôlés : image ISO, redirection RDP (`mstsc`), partage SMB (`net use`).
- Un dossier partagé n'est pas une copie : un malware dans la VM peut y toucher. L'isolation n'est plus totale.
- Réseau : `ipconfig /all`. NAT (la VM sort par l'hôte), Bridged (la VM est une machine de plus sur le réseau), Host-only (réseau privé hôte et VM, sans Internet).
- Un ping à « 0 % de perte » peut avoir échoué : lire les lignes, « Impossible de joindre l'hôte de destination » venant de sa propre adresse est un échec.

6. Arborescence et compléments

Linux (FHS)	Windows	Rôle
`/home`	`C:\Users`	Fichiers et configuration personnels
`/tmp`	`C:\Windows\Temp` et `%TEMP%`	Fichiers temporaires
`/bin`, `/sbin`, `/lib`	`C:\Windows\System32`	Exécutables et bibliothèques système
`/etc`	`System32\config` (registre), `ProgramData`, `drivers\etc`	Configuration
`/boot`	`C:\Windows\Boot` et partition EFI	Démarrage
`/var` (journaux)	`C:\Windows\Logs`, `System32\winevt\Logs`	Journalisation
`/usr/local`, `/opt`	`C:\Program Files`, `Program Files (x86)`	Logiciels installés
swap	`C:\pagefile.sys`	Mémoire virtuelle

- **hosts** (`System32\drivers\etc\hosts`) : associe des noms à des adresses IP, lu **avant** le DNS. Il ne contient pas l'adresse du serveur DNS. Un attaquant y redirige un nom vers un faux serveur.
- **Program Files** : 64 bits ; **Program Files (x86)** : 32 bits. **System32** : 64 bits ; **SysWOW64** : 32 bits. Nom conservé pour la compatibilité, historique et non logique.
- **pagefile.sys** : prolonge la RAM sur le disque, bien plus lent. Moins la VM a de RAM, plus il est sollicité.
- **Journaux** : `System.evtx`, format EVTX, lu avec l'Observateur d'événements.
- **BCD** (`bcdedit`) : entrées de démarrage (quel système, où, quelles options). Ce n'est pas la partition EFI, qui contient le chargeur.
- **Registre** (`regedit`) : centralisé et administrable à distance, mais point de défaillance unique, binaire, difficile à comparer, et lieu de persistance des malwares (clés Run).

- **sfc** : `sfc /verifyonly` détecte, `sfc /scannow` répare depuis le magasin local `C:\Windows\WinSxS`. Il ne retélécharge pas depuis le Store.
- **Mots de passe Windows** : stockés sous forme de hachages dans la base SAM (`C:\Windows\System32\config\SAM`). Ce point des modalités n'apparaît pas dans tes documents : vérifie-le dans ton cours.

Résumé Linux (Ubuntu / Debian)

Linux protège ses fichiers par les droits rwx que le noyau vérifie à chaque accès, et l'élévation passe par sudo, commande par commande et journalisée.

1. Processus et services

- **PID** : identifiant d'une instance ; chaque lancement en reçoit un nouveau. **PPID** : PID du parent. **NLWP** : nombre de threads. **NI** : valeur nice.
- **PID 1 = systemd** (`/sbin/init` est un lien vers systemd), ancêtre de tous les processus. **PID 2 = kthreadd**, parent des threads du noyau affichés entre crochets (`[kworker/0:1]`).
- **Orphelin** : un processus qui perd son parent est **adopté par systemd** et continue de tourner. **Zombie** : processus terminé dont le parent n'a pas encore lu le code retour.
- **Services** : en général un processus principal par service (avec éventuellement des enfants), sans hôte partagé comme svchost.exe. Liste : `systemctl list-units --type=service --state=running`.
- **root** (UID 0) : superutilisateur. **Comptes de service** (messagebus, syslog, systemd-resolve) : UID inférieur à 1000, shell `nologin`, personne ne s'y connecte ; ils font tourner un service avec des droits limités. Comptes humains : UID à partir de 1000.
- `$$` : PID du shell (stable). `$!` : PID de la dernière commande lancée en arrière-plan avec &.

Action (Moniteur système)	Signal	Effet
Arrêter	SIGSTOP	État T : en pause, pas terminé
Continuer	SIGCONT	État S : reprend
Terminer, `kill <PID>`	SIGTERM (15)	Demande polie ; le programme peut se nettoyer ou l'ignorer
Tuer, `kill -9 <PID>`	SIGKILL (9)	Le noyau termine le processus ; impossible à intercepter

- Équivalents Windows : `taskkill` sans puis avec `/F`.
- **nice** : de -20 (le plus prioritaire) à 19 (le moins prioritaire). Plus petit veut dire plus prioritaire. Sans sudo, on peut seulement augmenter son nice ; le diminuer (même de 10 à 5) demande sudo. `renice -n` fixe la nouvelle valeur, ce n'est pas un ajout.
- `sudo kill -9 1` ne fait rien : le noyau protège le PID 1, parfois sans afficher d'erreur.

Objectif	Commande
Compter les processus	`ps -e --no-headers | wc -l`
Liste triée par mémoire	`ps -eo pid,ppid,user,%cpu,%mem,nlwp,ni,comm --sort=-%mem | head -15`
Retrouver par nom	`pgrep -a <nom>`
Parent et état d'un PID	`ps -o pid,ppid,stat,cmd -p <PID>`
Arbre	`pstree -p`, `ps -ef --forest`
Temps réel	`top` (M mémoire, P CPU, q quitter)
Arrière-plan, priorité	`sleep 600 &`, `nice -n 10 sleep 600 &`, `renice -n 5 -p <PID>`
Liste des signaux	`kill -l`

2. Comptes, droits et administration

- **Qui refuse ?** Le **noyau**, d'après les droits. Message : « Permission denied ».
- **Lecture de `drwxr-xr-x 2 root root`** : `d` dossier (`-` fichier, `l` lien), puis propriétaire `rwx`, groupe `r-x`, autres `r-x`.
- **Octal** : r = 4, w = 2, x = 1. `rwx------` = 700, `rwxr-x---` = 750, `rwxr-xr-x` = 755, `rw-r--r--` = 644.

Droit	Sur un fichier	Sur un dossier
r	Lire le contenu	Lister
w	Modifier le contenu	Créer ou supprimer des fichiers dedans
x	Exécuter	Traverser

- **Règle clé** : modifier un fichier dépend du `w` du **fichier** ; le supprimer ou le créer dépend du `w` du **dossier** qui le contient. C'est pourquoi on ne peut pas supprimer `/usr/sbin/fsck`.
- **setuid** (`s` à la place du `x`, `-rwsr-xr-x`) : le programme s'exécute avec les droits de son propriétaire. `passwd` en a besoin pour écrire dans `/etc/shadow`.
- **Sticky bit** (`t`, `drwxrwxrwt` sur `/tmp`) : tout le monde peut écrire, mais chacun ne supprime que ses propres fichiers.
- **Fichiers cachés** : nom commençant par un point (`.bashrc`, `.profile`). Caché ne veut pas dire protégé : seuls les droits protègent.

- **`/etc/passwd`** : lisible par tous, 7 champs `nom:x:UID:GID:nom complet:dossier:shell`. **`/etc/shadow`** : lisible par root, 2e champ = **hachage** du mot de passe (pas un chiffrement). Séparés pour empêcher les attaques par dictionnaire hors ligne.
- `adduser` demande le mot de passe de façon interactive pour qu'il ne reste pas dans l'historique.
- **sudo** : élève une commande, demande le mot de passe de l'utilisateur lui-même (on prouve qui agit sans partager root) et journalise (`sudo journalctl -t sudo -n 5`).
- **`%sudo ALL=(ALL:ALL) ALL`** : les membres du groupe sudo (%), depuis toute machine, peuvent exécuter en tant que tout utilisateur et tout groupe, toute commande.
- **root** : verrouillé sur Ubuntu (`L`), mot de passe défini sur Debian (`P`). Le noyau ne lui applique pas les contrôles de permissions.
- **polkit** : la fenêtre « Déverrouiller » en graphique, équivalent de l'UAC.
- **Moindre privilège** : une erreur (`rm -rf`) ou un compte compromis aurait un impact total si on restait administrateur par confort.

Dossiers personnels :

- `adduser` crée `/home/<nom>` **immédiatement**, à la création du compte, en copiant `/etc/skel` (Windows : à la première connexion).
- Droits par défaut : 750 sur Ubuntu 24.04, 700 sur Debian. Les « autres » n'ont rien.
- Après `chmod o+rx /home/etudiant` : bob lit le fichier, ne le modifie pas (pas de `w` sur le fichier), ne le supprime pas (pas de `w` sur le dossier).
- `deluser bob` garde `/home/bob` (option `--remove-home` pour le supprimer). `ls -ln` montre un UID sans nom ; un nouvel utilisateur qui reçoit cet UID hérite des fichiers.
- Un changement de groupe n'est pris en compte qu'à la reconnexion.

Objectif	Commande
Droits et propriétaire	`ls -l`, `ls -ld <dossier>`, `stat <fichier>`
Changer les droits	`chmod 750 <dossier>`, `chmod o+rx <dossier>`
Changer le propriétaire	`sudo chown etudiant:prof <fichier>`
Identité et groupes	`id`, `id etudiant`, `groups`
Créer, supprimer un compte	`sudo adduser etudiant`, `sudo deluser bob`
Ajouter au groupe sudo	`sudo usermod -aG sudo etudiant` (sans `-a`, les autres groupes sont effacés)
Retirer du groupe sudo	`sudo gpasswd -d etudiant sudo`
Membres du groupe	`getent group sudo`

ACL pour un seul utilisateur	`setfacl -m u:bob:rx <dossier>`, lecture avec `getfacl`

3. Terminal et bash

- **bash** : le shell qui interprète les commandes ; il manipule du **texte**.
- Le terminal démarre dans le dossier personnel `/home/<utilisateur>`, abrégé ~ (ce n'est pas `/home`).
- `.` dossier courant, `..` dossier parent.
- **Tube |** : envoie la sortie d'une commande vers l'entrée de la suivante. **>** écrit dans un fichier en l'écrasant ; **>>** ajoute à la fin.
- **Alias** : raccourci, défini dans `~/.bashrc` (`alias ll='ls -alF'`) ; `type ll` le révèle.
- Aide : `man ls` (q pour quitter), `ls --help`.
- `file <fichier>` donne la nature réelle d'un fichier ; `stat` est plus complet que l'interface (octal, UID et GID, inode, dates).

Objectif	Linux	Windows
Où suis-je	`pwd`	`cd`
Lister, cachés compris	`ls -la`	`dir /a`
Seulement les cachés	`ls -d .*`	`dir /a:h`
Propriétaire et droits	`ls -l`, `stat`, `getfacl`	`dir /q`, `icacls`, `Get-Acl`
Identité et groupes	`id`, `groups`	`whoami /groups`
Processus	`ps aux`, `top`	`tasklist`, `Get-Process`
Terminer	`kill`, `kill -9`	`taskkill`, `taskkill /F`
Créer un compte	`sudo adduser`	`net user ... /add`
Rendre administrateur	`sudo usermod -aG sudo`	`net localgroup Administrateurs ... /add`
Élévation	`sudo`, polkit	UAC, Exécuter en tant qu'administrateur
Installer	`sudo apt install`	`winget install --id`

Empreinte	`sha256sum`	`Get-FileHash`
Journaux	`journalctl`	`eventvwr.msc`
Réseau	`ip a`	`ipconfig /all`
Intégrité du système	`dpkg -V`	`sfc /verifyonly`
Matériel et version	`cat /etc/os-release`, `uname -r`, `free -h`, `nproc`, `lscpu`	`winver`, `systeminfo`

4. Installation de logiciels

Source	Qui vérifie ?	Risque
apt (dépôts)	Mainteneurs de la distribution, paquets signés (GPG)	Faible
Snap	Éditeur et magasin Canonical, confinement	Faible à moyen
.deb d'un site	L'éditeur seul, vérification manuelle ; scripts exécutés en root	Moyen à élevé
`curl ... | sudo bash`	Personne, exécution immédiate en root	Le plus élevé

- **`sudo apt update`** : télécharge la liste des paquets disponibles. Il n'installe rien et ne met rien à jour (c'est `apt upgrade`).
- **Droits** : apt écrit dans `/usr` et dans la base des paquets, donc root. Sans sudo : « Could not open lock file /var/lib/dpkg/lock-frontend (13: Permission denied) ».
- **Empreinte SHA-256** : intégrité par rapport à une référence fiable. **Signature GPG** : authenticité de la source. Aucune des deux ne prouve que le logiciel est inoffensif.
- **Scripts `preinst` et `postinst`** d'un .deb : exécutés en root. Un .deb inconnu exécute le code de son auteur en root.
- **Verrou** `/var/lib/dpkg/lock-frontend` : une seule modification de la base à la fois. On ne supprime jamais le fichier ; `sudo lsof /var/lib/dpkg/lock-frontend` montre qui le tient.
- **xz (CVE-2024-3094)** : porte dérobée insérée dans xz 5.6.0 et 5.6.1 par un contributeur devenu mainteneur, découverte en mars 2024. La confiance dans les dépôts officiels n'est jamais aveugle.
- Les fichiers d'un paquet sont répartis selon la FHS (`/usr/bin`, `/usr/share/...`), pas regroupés comme dans Program Files.

Objectif	Commande
Chercher, détails	`apt search htop`, `apt show htop`

Installer	`sudo apt install htop`
Vérifier	`command -v htop`, `dpkg -l htop` (`ii` = installé)
Fichiers d'un paquet	`dpkg -L htop`
Paquet d'un fichier	`dpkg -S /usr/bin/htop`
Désinstaller, config gardée	`sudo apt remove htop` (état `rc`)
Désinstaller, config comprise	`sudo apt purge htop`
Dépendances inutiles	`sudo apt autoremove`
Inspecter un .deb	`sha256sum f.deb`, `dpkg-deb --info f.deb`, `dpkg-deb --contents f.deb`
Installer un .deb	`sudo apt install ./f.deb`

5. Arborescence (FHS) et compléments

La FHS est une norme issue d'un consensus : la structure est la même d'une distribution à l'autre. Windows est uniforme parce qu'un seul éditeur le fait.

Dossier	Rôle
`/etc`	Configuration, en fichiers texte
`/home`	Dossiers personnels
`/tmp`	Temporaires, partagé par tous (sticky bit)
`/bin`, `/sbin`, `/lib`, `/usr`	Exécutables et bibliothèques
`/var/log`	Journaux
`/boot`	Noyau et chargeur de démarrage (GRUB)
`/proc`, `/dev`	« Tout est fichier » : processus et périphériques vus comme des fichiers
swap	Mémoire virtuelle

Mots de passe Linux : hachages dans `/etc/shadow`.

Résumé Web (MiniShop)

La règle qui traverse les points 6 à 13 : tout ce qui vient du visiteur est non fiable, et tout ce qui est envoyé au navigateur est visible et modifiable par lui ; les données et les contrôles sensibles restent côté serveur.

1. Site statique : HTML et CSS (point 6)

- **Client et serveur** : le navigateur (client) demande une page, le serveur la renvoie. Un site statique envoie les fichiers tels quels, sans calcul côté serveur.
- **Limite du statique** : un produit = un fichier. 500 produits = 500 fichiers à la main ; un prix répété dans la fiche, le catalogue et le comparatif finit par diverger. Ce n'est pas impossible, c'est ingérable.
- **HTML** décrit la structure et le contenu ; **CSS** décrit l'apparence, dans un fichier séparé lié à toutes les pages : un changement à un seul endroit s'applique partout.
- **Code source et DOM** : Ctrl + U montre le fichier reçu ; F12, Éléments montre le DOM, la page reconstruite par le navigateur après corrections. Le navigateur est tolérant et corrige en silence un HTML mal écrit.
- **URL** : `https://site.be/catalogue.html?nom=test#haut` = protocole, domaine, chemin, paramètres après `?`, ancre après #.

Squelette à savoir écrire :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>MiniShop - Accueil</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <header><nav><a href="catalogue.html">Catalogue</a></nav></header>
  <main><h1>Bienvenue</h1><p>Texte</p></main>
  <footer>Contact</footer>
</body>
</html>
```

Élément	À retenir
`<!DOCTYPE html>`, `lang`, `charset`	Rendu standard ; langue pour lecteurs d'écran ; accents corrects

`header`, `nav`, `main`, `footer`	Balises sémantiques : elles décrivent le rôle de chaque zone
`h1` à `h6`, `p`, `strong`, `em`	Titres, paragraphe, importance, emphase (un sens, pas seulement une apparence)
`ul`, `ol`, `li`	Liste à puces, liste numérotée, élément
`<img src="images/casque.jpg" alt="Casque audio">`	`alt` : lu par les lecteurs d'écran et affiché si l'image est introuvable ; `alt=""` pour une image décorative
`<a href="...">`	Lien ; le texte affiché n'est pas forcément la destination (hameçonnage)
`table`, `caption`, `thead`, `tbody`, `tr`, `th`, `td`	`th` = en-tête, `td` = donnée ; ligne = enregistrement, colonne = champ
`form`, `label for`, `input`, `select`, `textarea`	`method`, `action`, `name` de chaque champ

- **Chemin relatif** : calculé depuis l'emplacement de la page qui le contient (`images/casque.jpg`, `../images/casque.jpg`). **Chemin absolu** : adresse complète (`https://...`), qui peut viser un autre site ou le sien.
- **Lien externe** : charger une ressource chez un tiers, c'est lui faire confiance. Avec `target="_blank"`, la boutique reste ouverte dans l'onglet d'origine ; on ajoute `rel="noopener noreferrer"` pour que la page ouverte ne puisse pas manipuler la page d'origine (`window.opener`) ni recevoir son adresse.

	GET	**POST**
Où vont les données	Dans l'URL, après `?`	Dans le corps de la requête
Traces	Historique, favoris, journaux du serveur, liens partagés	Pas dans l'URL ni l'historique
Usage	Consultation : recherche, filtres	Action : connexion, commande
Sécurité	Jamais de mot de passe	Pas chiffré pour autant : seul HTTPS chiffre le trajet

- **`type="password"`** masque l'affichage, rien de plus : aucun chiffrement.
- **`required`, `type="email"`** : contrôles du navigateur, pratiques mais contournables (F12, ou requête envoyée sans le formulaire). La vraie validation se fait côté serveur.
- **Envoyer n'est pas traiter** : dans le TP, la page est ouverte depuis le disque (`file://`), aucun serveur ne reçoit le formulaire.
- **Audit (reconnaissance)** : n'importe quel visiteur, sans outil spécial, lit les commentaires HTML, les liens et ressources externes, les noms de dossiers, les champs et la méthode du formulaire, les champs `hidden`, le contenu en `display:none`. On peut masquer, pas

cacher. Jamais de mot de passe, clé d'API, adresse interne ou note de sécurité dans un commentaire.
- CSS introuvable : aucune erreur dans la page, style par défaut ; l'erreur 404 se voit dans F12, Réseau et Console.

2. JavaScript, côté navigateur (point 7)

- **Qui exécute quoi** : JavaScript s'exécute dans le navigateur, sous le contrôle du visiteur. PHP s'exécute sur le serveur ; le visiteur ne reçoit que le HTML produit.
- **DOM** : la page en mémoire, représentée par `document`. **Événement** : une action (clic, saisie) qui déclenche une fonction.
- **Console (F12)** : `document.body.style.background = "yellow"` modifie la page dans mon navigateur seulement. Ni le fichier ni le serveur ne changent, personne d'autre ne le voit, F5 remet tout en état.
- **Le côté client n'est jamais une garantie** : taper `nombre = 500` dans la console puis cliquer donne 501. Un prix, un stock ou un droit d'accès ne se décident pas côté client.
- Un script ne réagit que si la page le charge (`<script src="js/app.js"></script>` avant `</body>`).

```
const bouton = document.getElementById("ajouter");

let nombre = 0;

if (bouton) {
  bouton.addEventListener("click", function () {
    nombre = nombre + 1;
    document.getElementById("compteur").textContent = nombre;
  });
}
```

XSS : une saisie du visiteur est interprétée comme du code.

```
message.innerHTML = "Bonjour " + saisie.value;   // dangereux : la saisie est lue comme du HTML

message.textContent = "Bonjour " + saisie.value; // sûr : la saisie reste du texte
```

- Démonstration : saisir `<img src=x onerror="alert('XSS')">`. L'image échoue à charger et le code de `onerror` s'exécute.
- Conséquences d'un XSS : vol de session (cookie), faux formulaire, redirection.
- `textContent` règle le XSS, pas la manipulation d'une valeur dans la console : ce sont deux problèmes distincts.

3. Base de données, SQL et PHP (point 7)

- **XAMPP** : Apache (serveur web, port 80, 443 en HTTPS), MySQL/MariaDB (base, port 3306), PHP (langage qu'Apache exécute, il ne tourne pas seul), phpMyAdmin (application PHP d'administration de la base).
- **`file://` ou `http://localhost`** : avec `file://`, aucun serveur, PHP n'est pas exécuté. Avec `http://localhost`, le navigateur est le client, Apache le serveur.

- **Trajet d'une page .php** : le navigateur demande l'adresse, Apache lance PHP, PHP interroge la base, puis renvoie du HTML.
- **Base, table, colonne (champ), ligne (enregistrement)**. **Clé primaire** : identifie chaque ligne de façon unique et non nulle ; un doublon est refusé (« Duplicate entry »). `AUTO_INCREMENT` laisse la base calculer l'id.
- `DECIMAL(8,2)` : nombre exact à 2 décimales ; permet calculs et tri numérique, refuse le non numérique. `NOT NULL` interdit l'absence de valeur, pas la chaîne vide.
- Chaque action dans phpMyAdmin correspond à une requête SQL.

```
CREATE TABLE produits (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(100) NOT NULL,
  prix DECIMAL(8,2) NOT NULL,
  stock INT NOT NULL
);
INSERT INTO produits (nom, prix, stock) VALUES ('Casque audio', 49.90, 12);
SELECT nom, prix FROM produits WHERE prix < 50 ORDER BY nom;
UPDATE produits SET prix = 44.90 WHERE id = 1;
DELETE FROM produits WHERE id = 1;
```

- **Sans `WHERE`**, `UPDATE` et `DELETE` restent valides et touchent **toutes** les lignes. Réflexe : tester d'abord un `SELECT` avec le même `WHERE`.
- `DELETE FROM t` vide la table (la structure reste) ; `DROP TABLE t` supprime la table.
- **Lu, pas pratiqué** : la **clé étrangère** relie une table à une autre (`produits.categorie_id` vers `categories.id`) et la base refuse une référence inexistante ; le **JOIN** reconstitue l'information répartie sur deux tables ; l'**index** accélère les recherches, au prix d'espace disque et d'écritures plus lentes.

```
<?php
$conn = new mysqli("localhost", "root", "", "minishop");
$resultat = $conn->query("SELECT nom, prix FROM produits ORDER BY nom");
while ($p = $resultat->fetch_assoc()) {
    echo "<h3>" . htmlspecialchars($p["nom"]) . "</h3>";
    echo "<p>" . htmlspecialchars($p["prix"]) . " €</p>";
}
?>
```

- Le visiteur ne reçoit que le HTML : ni le code PHP, ni la requête, ni le mot de passe de la base.
- Une seule page sert 8 ou 500 produits ; un prix modifié dans la base apparaît après F5.
- **`htmlspecialchars`** encode les caractères spéciaux (`<` devient `<`) pour qu'ils s'affichent comme du texte : l'équivalent PHP de `textContent`.

- Une recherche en JavaScript masque seulement l'affichage : tous les produits ont été envoyés et restent dans le code source.

4. Architecture à trois niveaux (point 8)

Niveau	Rôle	Dans MiniShop
Présentation	Interface utilisateur, dans un navigateur ou une application graphique	HTML, CSS, JavaScript
Application	Traitement et règles métier ; authentifie et autorise ; ajoute, modifie, supprime des données	PHP sur Apache
Données	Stockage	MySQL/MariaDB

- Toute communication passe par le niveau application : la présentation ne parle jamais directement à la base.
- Gain de sécurité : le niveau application protège les données contre les attaques venant de la présentation.
- **Utilisateur technique** : le compte avec lequel l'application se connecte à la base (`minishop_app`), distinct des comptes des visiteurs.
- Défaut d'une application où tout est mélangé : affichage, décision et accès à la base dans les mêmes fichiers, donc difficile à maintenir et à sécuriser.

5. Authentification et autorisation (point 9)

	Authentification	Autorisation
Question	Es-tu bien celui que tu prétends être ?	As-tu le droit d'accéder à cette ressource ou fonction ?
Ordre	Première étape, réalisée par l'application sur le serveur	Toujours après l'authentification
Exemples	Mot de passe, code à usage unique, jeton, biométrie	Un client voit ses commandes, pas celles des autres ; seul l'admin modifie les prix

MFA : exiger plus d'un facteur avant d'accorder l'accès. L'autorisation s'appelle aussi contrôle d'accès ou privilège.

6. Sessions et cookies (point 10)

- **Session** : série d'actions d'un visiteur sur un site dans un laps de temps donné (panier puis paiement). Anonyme, ou identifiée dès que l'utilisateur s'authentifie.
- **Session ID** : chaîne aléatoire générée par le serveur. Le navigateur la stocke dans un **cookie** et la renvoie à chaque requête.
- Elle sert de jeton d'accès, lié côté serveur à l'identité : pas besoin de se réauthentifier à chaque requête.
- Le cookie a une durée de validité, celle souhaitée pour la session. Il peut stocker d'autres informations (préférences), en quantité limitée.

- Cookies supprimés après connexion : le navigateur n'envoie plus le session ID, le serveur ne reconnaît plus l'utilisateur, il faut se reconnecter. Navigation privée : les cookies sont effacés à la fermeture, la session ne survit pas.

7. HTTPS et certificats (point 11)

- HTTPS est la version sécurisée de HTTP : tous les échanges entre le navigateur et le serveur sont chiffrés.
- Deux attaques contrées : l'écoute du trafic (man-in-the-middle) et l'usurpation DNS.
- Le **certificat** présenté par le serveur prouve qu'il est bien le propriétaire du nom de domaine.
- Contenu : domaine, titulaire, autorité de certification, signature numérique, sous-domaines, dates d'émission et d'expiration, clé publique.
- L'**autorité de certification** est reconnue par les navigateurs : un acteur malveillant ne peut pas présenter un faux certificat.
- La clé publique sert à chiffrer ; la clé privée reste secrète.

8. Types d'applications et sites dynamiques (point 12)

Type	Définition
Native	Propre à un OS
Web	Utilisée via un navigateur
Serveur	Sans interface, utilisée par d'autres applications via des API
Logiciel système	Tourne en arrière-plan

- **Site dynamique** : la page HTML est construite à la demande par une application serveur, en général depuis une base de données. Exemple : agence immobilière dont les pages reflètent la disponibilité des biens en temps réel.
- Côté navigateur : JavaScript, seul langage compris par tous les navigateurs. Côté serveur : PHP, Java, ASP.NET, Ruby, Python, JavaScript.
- Un site statique est en théorie plus difficile à pirater (pas de base, pas de plugins) mais plus difficile à maintenir.
- Bases relationnelles courantes : MySQL, Oracle, PostgreSQL, Microsoft SQL Server. Une base de données est une application serveur.

9. Injection SQL et moindre privilège (point 13)

L'injection SQL : une donnée envoyée par l'utilisateur modifie la requête au lieu d'être traitée comme du texte. Même principe que le XSS : une donnée utilisateur est interprétée comme du code.

```
// Vulnérable : la valeur est collée dans le texte SQL
$sql = "SELECT nom, prix FROM produits WHERE id = " . $_GET["id"];


// Corrigé : validation + requête préparée
$id = filter_input(INPUT_GET, "id", FILTER_VALIDATE_INT);
$stmt = $conn->prepare("SELECT nom, prix FROM produits WHERE id = ?");
```

```
$stmt->bind_param("i", $id);
$stmt->execute();
```

Test	Effet
`?id=1'` ou `?id=abc`	Requête invalide ; l'erreur peut révéler la base, des noms de tables, des chemins
`?id=1 OR 1=1`	Condition toujours vraie : la requête concerne toutes les lignes
`UNION SELECT ...`	Ajoute le résultat d'une autre requête (même nombre de colonnes) : lecture d'une autre table

- **Protections** : requêtes préparées (la structure part avant les valeurs), validation des entrées côté serveur, échappement des caractères spéciaux, erreurs non détaillées.
- **Compte à droits limités** : `CREATE USER 'minishop_app'@'localhost' IDENTIFIED BY '...';` puis `GRANT SELECT ON minishop.* TO 'minishop_app'@'localhost';`. Un `UPDATE` est alors refusé par **MySQL** (« UPDATE command denied »), pas par PHP.
- Si une injection subsiste : avec ce compte l'attaquant lit seulement ; avec root il peut tout modifier et détruire.
- Réglages de laboratoire d'XAMPP (root sans mot de passe, phpMyAdmin ouvert) : acceptables en laboratoire, jamais en production.

Exercices Windows

Réponds par écrit avant de regarder le corrigé. Les exercices « pratique » se font sur ta VM, après un snapshot.

Questions de compréhension

W1. Ta VM tourne sur un serveur ESXi et tu y accèdes avec Workstation en « Connect to Server ». Quel est l'hyperviseur, de quel type, et quel est le rôle de ton PC ? Que change le fait d'utiliser VirtualBox en local ?

W2. Windows affiche 4 Go de RAM et 2 processeurs. Qui a décidé ces valeurs ? Pourquoi le modèle de processeur affiché peut-il être celui du serveur ?

W3. Explique la différence entre un processus et un service, puis le rôle de svchost.exe. Donne les deux raisons pour lesquelles il y en a autant.

W4. Tu termines explorer.exe (PID 4112), puis tu le relances : il porte le PID 4584. Que vois-tu à l'écran entre les deux, et que conclus-tu sur ce qu'est un PID ?

W5. Classe SYSTEM, LOCAL SERVICE, NETWORK SERVICE et un compte administrateur du plus au moins privilégié, et dis comment les deux comptes de service se présentent sur le réseau.

W6. Un administrateur essaie de supprimer `C:\Windows\System32\chkdsk.exe` et obtient un refus. Qui refuse, et pourquoi le fait d'être administrateur ne suffit-il pas ? Quelles deux commandes permettraient de passer outre ?

W7. `whoami /groups` montre le groupe Administrateurs dans un cmd normal ouvert par un administrateur, pourtant `net user test Test#2026 /add` répond « Accès refusé ». Explique avec les notions de jeton et de niveau d'intégrité.

W8. Tu ajoutes etudiant au groupe Administrateurs alors que sa session est ouverte. Peut-il créer un compte tout de suite ? Que doit-il faire, et que lui demandera l'UAC ensuite ?

W9. Après `net user bob /delete`, que devient `C:\Users\bob` ? Si on recrée un compte « bob », a-t-il accès aux anciens fichiers ? Pourquoi ?

W10. Décrypte cette ligne d'`icacls C:\Windows\System32` : `BUILTIN\Utilisateurs:(RX)`. Que signifient aussi `(F)` et `(M)` ?

W11. Pourquoi `ls` fonctionne-t-il dans PowerShell, et pourquoi `ls -l` n'y fonctionne-t-il pas ? Quelle différence de fond sépare le tube de PowerShell de celui de cmd ?

W12. Tu télécharges `npp-installer.exe`. Que t'apprend son empreinte SHA-256, à quelle condition ? Que t'apprend sa signature ? Qu'est-ce qu'aucune des deux ne garantit ?

W13. Compare Microsoft Store, winget et un .exe téléchargé : qui vérifie, lequel est le plus risqué ? Pourquoi `winget install --id 7zip.7zip -e` déclenche-t-il l'UAC alors qu'un petit jeu installé par winget ne le fait pas forcément ?

W14. Un collègue dit : « J'ai pris un snapshot hier, donc ma VM est sauvegardée. » Corrige-le en trois arguments.

W15. Compare Shut Down Guest, Power Off et Suspend : effet sur un Bloc-notes non enregistré, sur la Durée d'activité, et équivalent dans le monde physique. Lequel a besoin de VMware Tools ?

W16. Dans l'Observateur d'événements, tu trouves les événements 41 et 6008, puis plus loin un 1074. Que raconte chacun ?

W17. À quoi sert le fichier `hosts`, et que gagne un attaquant à le modifier ? Quelle confusion faut-il éviter avec le DNS ?

W18. Vrai ou faux, avec justification :

1. System32 contient les composants 32 bits.
2. Un dossier masqué est un dossier protégé.
3. `sfc /scannow` retélécharge les fichiers manquants depuis le Microsoft Store.
4. Un ping qui affiche « 0 % de perte » a forcément réussi.
5. `taskkill /PID 1234` force l'arrêt du processus.
6. Le profil d'un utilisateur est créé au moment de `net user ... /add`.

Exercices pratiques sur la VM

W19. Scénario complet, en ligne de commande uniquement. Note chaque commande et chaque message obtenu.

1. Crée le compte `alice` avec le mot de passe `Alice#2026`.
2. Vérifie qu'elle n'est pas administratrice.
3. Ajoute-la au groupe Administrateurs, vérifie, puis retire-la.
4. Supprime le compte et dis ce qu'il reste sur le disque.

W20. Lance le Bloc-notes, retrouve son PID en ligne de commande, termine-le sans `/F` puis relance-le et termine-le avec `/F`. Refais la même chose avec le Gestionnaire des tâches et compare.

W21. Sans l'interface graphique, donne le propriétaire de `C:\ProgramData` et de `C:\Windows\System32` de deux façons différentes (cmd et PowerShell), puis liste les éléments cachés de `C:\`.

W22. Compte le nombre d'instances de svchost.exe en une commande, puis affiche les services hébergés par l'une d'elles.

W23. Avec winget : cherche Notepad++, installe-le par son Id, vérifie l'installation de trois manières, puis désinstalle-le. À quel moment l'UAC apparaît-il, et pourquoi ?

W24. Retrouve dans Windows puis dans la configuration de la VM : version de Windows, RAM, nombre de processeurs. Prends un snapshot nommé `Avant_test` et retrouve-le dans le gestionnaire de snapshots.

Exercices Linux

Questions de compréhension

L1. Définis PID, PPID et thread. Quels processus portent les PID 1 et 2, et quel est leur rôle ?

L2. Un terminal lance `sleep 600 &`, puis on ferme brutalement le processus parent. Que devient `sleep` ? Quelle est la différence avec un zombie ?

L3. Associe chaque action du Moniteur système (Arrêter, Continuer, Terminer, Tuer) à son signal et à l'état du processus. Que fais-tu si un programme ignore SIGTERM ?

L4. Un processus a un nice de 10. En utilisateur normal, tu tapes `renice -n 15 -p <PID>`, puis `renice -n 5 -p <PID>`. Que se passe-t-il dans chaque cas, et pourquoi ?

L5. Que sont messagebus, syslog et systemd-resolve dans la colonne des utilisateurs ? Donne trois caractéristiques qui les distinguent d'un compte humain.

L6. Décrypte entièrement : `-rwsr-xr-x 1 root root 68208 /usr/bin/passwd`. Pourquoi ce programme a-t-il besoin du `s` ?

L7. Décrypte `drwxrwxrwt 18 root root /tmp`. Qui peut y écrire ? Qui peut y supprimer le fichier d'alice ?

L8. Convertis en octal : `rwxr-x---`, `rw-r--r--`, `rwx------`, `rwxrwxrwt`. Convertis en lettres : 755, 640, 600.

L9. Dans `/home/etudiant` (droits `drwxr-xr-x`), le fichier `note.txt` a les droits `-rw-r--r--` et appartient à etudiant. Bob peut-il le lire, le modifier, le supprimer ? Justifie chaque réponse par le droit précis qui s'applique.

L10. On passe `/home/etudiant` en `drwxrwxrwx`, le fichier reste en `-rw-r--r--`. Bob peut-il maintenant modifier `note.txt` ? Le supprimer ?

L11. Pourquoi les mots de passe sont-ils dans `/etc/shadow` et non dans `/etc/passwd` ? Que contient le 2e champ de chacun ? Pourquoi dit-on « haché » et pas « chiffré » ?

L12. Lis élément par élément : `%sudo ALL=(ALL:ALL) ALL`. Quel mot de passe sudo demande-t-il, et pourquoi ce choix ?

L13. Tu tapes `sudo usermod -G sudo etudiant` (sans `-a`). Quel est le problème ? Et pourquoi `groups` dans la session déjà ouverte d'etudiant ne montre-t-il pas sudo ?

L14. Après `sudo deluser bob`, `ls -ln /home` affiche `1002` comme propriétaire de `/home/bob`. Explique, et décris le risque si on crée ensuite le compte `carla`.

L15. Explique `|`, `>` et `>>` avec un exemple chacun. Écris une commande qui compte les comptes interdits de connexion.

L16. Que fait exactement `sudo apt update` ? Différence entre `remove`, `purge` et `autoremove` ? Que signifie `rc` dans `dpkg -l` ?

L17. Classe du moins risqué au plus risqué : un .deb téléchargé sur un site, `curl ... | sudo bash`, apt, Snap. Dis qui vérifie le logiciel dans chaque cas, puis ce que l'affaire xz de 2024 change à la confiance dans apt.

L18. Le second `apt install` répond « Could not get lock /var/lib/dpkg/lock-frontend ». À quoi sert ce verrou, que ne faut-il jamais faire, et comment trouver qui le tient ?

L19. Vrai ou faux, avec justification :

1. Un fichier dont le nom commence par un point est protégé.
2. `~` désigne le dossier `/home`.
3. Un nice de -5 est moins prioritaire qu'un nice de 5.
4. `sudo kill -9 1` éteint la machine.
5. Le dossier personnel est créé à la première connexion de l'utilisateur.
6. Une empreinte SHA-256 correcte prouve que le logiciel est inoffensif.
7. root a un mot de passe sur Ubuntu.

Exercices pratiques sur la VM

L20. Scénario complet. Note chaque commande et chaque message.

1. Crée les comptes `alice` et `carl`.
2. En alice, crée `~/secret.txt` contenant « test ». Relève les droits du dossier et du fichier.
3. En carl, essaie de lire le fichier. Explique le résultat.
4. Donne à carl seul la lecture du dossier et du fichier, sans toucher aux droits des « autres ».
5. Vérifie, puis supprime les deux comptes avec leurs dossiers.

L21. Lance `yes > /dev/null &`. Retrouve son PID de deux façons, son parent, son pourcentage de CPU. Mets-le en pause, vérifie son état, relance-le, puis termine-le proprement.

L22. En une ligne chacune : (a) le nombre de processus ; (b) les 5 processus les plus gourmands en mémoire avec PID, PPID et utilisateur ; (c) le nombre de processus par utilisateur.

L23. Affiche la généalogie de ton shell jusqu'au PID 1. Donne le PID de ton shell et celui de ta dernière commande en arrière-plan.

L24. Installe `tree` avec apt. Donne son emplacement, la liste de ses fichiers, le paquet propriétaire de `/usr/bin/tree`. Désinstalle-le en gardant la configuration, constate l'état, puis purge.

L25. Télécharge un paquet sans l'installer (`apt download htop`), calcule son empreinte, affiche ses métadonnées et la liste des fichiers qu'il déposerait.

L26. Retrouve dans Linux puis dans la configuration de la VM : distribution, noyau, RAM, nombre de processeurs. Pourquoi `free -h` affiche-t-il un peu moins que la RAM configurée ?

Exercices croisés Windows et Linux

X1. Complète de mémoire le tableau d'équivalences pour : lister les processus, forcer l'arrêt, lister les fichiers cachés, voir le propriétaire, créer un compte, le rendre administrateur, installer un logiciel, calculer une empreinte, lire les journaux.

X2. Compare en un tableau : le moment de création du dossier personnel, ce qui reste après suppression du compte, le mécanisme d'élévation, le mot de passe demandé, l'endroit où sont stockés les mots de passe.

X3. « Qui refuse ? » Réponds pour : la suppression de `/etc/hostname` sans sudo ; la suppression de `C:\Windows\System32\notepad.exe` en administrateur ; `net user test /add` dans un cmd non élevé.

X4. Explique le principe du moindre privilège avec un exemple Windows et un exemple Linux tirés des TP.

Exercices Web

Questions de compréhension

M1. Explique le fonctionnement d'un site statique et sa limite principale, avec l'exemple du prix d'un produit de MiniShop.

M2. Quelle est la différence entre un chemin relatif et un chemin absolu ? Relatif à quoi ? Que valent `images/casque.jpg` et `../images/casque.jpg` ?Résumé Web (MiniShop)

M3. Un lien externe porte `target="_blank"`. Que se passe-t-il pour l'onglet de la boutique ? Pourquoi ajoute-t-on `rel="noopener noreferrer"` ?

M4. Un formulaire de connexion utilise `method="get"`. Décris l'URL obtenue et cite quatre endroits où le mot de passe laisse une trace. Passer en POST suffit-il à sécuriser le formulaire ?

M5. Vrai ou faux, avec justification :

1. `type="password"` chiffre le mot de passe.
2. `required` empêche un attaquant d'envoyer un champ vide.
3. Un commentaire HTML n'est pas envoyé au visiteur.
4. Un élément en `display:none` est invisible dans le code source.
5. Si le fichier CSS est introuvable, la page affiche une erreur.

M6. Tu audites le site d'un binôme avec Ctrl + U et F12. Cite cinq informations qu'un attaquant peut y apprendre, et trois choses qui ne doivent jamais figurer dans un commentaire.

M7. Dans la console, tu tapes `document.body.style.background = "yellow"`. Qu'est-ce qui a changé, où, et qui le voit ? Que se passe-t-il après F5 ?

M8. Le compteur du panier est géré en JavaScript. Tu tapes `nombre = 500` dans la console puis tu cliques sur « Ajouter ». Qu'affiche le compteur, et quelle leçon en tirer pour un prix ou un stock ?

M9. Explique pourquoi `innerHTML` avec une saisie utilisateur ouvre la porte au XSS, et pourquoi `textContent` corrige le problème. Quelle saisie de test utilises-tu, et pourquoi pas une balise `<script>` ?

M10. Quel est le rôle de chacun des quatre composants d'XAMPP ? Que se passe-t-il si tu ouvres `catalogue.php` avec `file://` ?

M11. Définis base, table, colonne, ligne et clé primaire. Que fait la base si on insère deux lignes avec le même id ?

M12. Que font ces deux requêtes : `UPDATE produits SET prix = 1;` et `DELETE FROM produits;` ? Est-ce une erreur de syntaxe ? Différence avec `DROP TABLE produits;` ?

M13. Un visiteur affiche le code source de `catalogue.php`. Que voit-il ? Que ne voit-il pas ? À quoi sert `htmlspecialchars` ?

M14. À quoi servent une clé étrangère, un JOIN et un index ? Une phrase chacun.

M15. Décris les trois niveaux d'une application web et place HTML, PHP et MySQL. Pourquoi la présentation ne parle-t-elle jamais directement à la base ? Qu'est-ce qu'un utilisateur technique ?

M16. Classe chaque situation en authentification ou autorisation : saisir un code reçu par SMS ; un client ne peut pas voir les commandes d'un autre ; empreinte digitale ; seul l'administrateur peut modifier un prix. Laquelle vient toujours en premier ?

M17. Explique le lien entre session, session ID et cookie. Que se passe-t-il si tu supprimes les cookies après t'être connecté ? Et en navigation privée ?

M18. Quelles deux attaques HTTPS contre-t-il ? Que prouve le certificat, que contient-il (cinq éléments), et pourquoi un attaquant ne peut-il pas en fabriquer un faux ?

M19. Donne les quatre types d'applications avec un exemple chacun. Qu'est-ce qu'un site dynamique ? Pourquoi un site statique est-il plus difficile à pirater mais plus difficile à maintenir ?

M20. `produit.php` contient `"... WHERE id = " . $_GET["id"]`. Que se passe-t-il avec `?id=1 OR 1=1` ? Quel est le point commun avec le XSS ? Cite trois protections.

M21. L'application se connecte à la base avec root sans mot de passe. Quel principe est violé ? Que change un compte `minishop_app` limité à `SELECT` si une injection subsiste ? Qui refuse un `UPDATE` lancé avec ce compte ?

Exercices pratiques

M22. Écris de mémoire une page `contact.html` valide : squelette complet, un titre, un lien vers `index.html`, une image avec `alt`, et un formulaire en POST avec nom, e-mail, mot de passe et bouton d'envoi, chaque champ ayant son `label`.

M23. Écris le tableau HTML comparant deux produits (nom, prix, stock) avec légende et en-têtes, puis un fichier `css/style.css` qui met les en-têtes sur fond coloré et fusionne les bordures.

M24. Écris un script qui affiche « Bonjour » suivi du prénom saisi quand on clique sur un bouton. Fais d'abord la version vulnérable, démontre le XSS, puis corrige.

M25. En SQL : crée une table `clients` (id, nom, email), insère deux clients, affiche-les triés par nom, change l'e-mail du premier, supprime le second.

M26. Écris la page PHP qui affiche le nom et l'e-mail de tous les clients dans une liste, sans risque de XSS.

M27. Corrige ce code pour qu'il résiste à l'injection SQL :

```
$nom = $_GET["nom"];
$sql = "SELECT nom, prix FROM produits WHERE nom = '" . $nom . "'";
$resultat = $conn->query($sql);
```

Corrigés

Windows

W1. L'hyperviseur est ESXi, de type 1 (installé directement sur le matériel du serveur). Workstation n'exécute pas la VM ici : c'est un client. Le PC affiche la console et transmet clavier et souris. Avec VirtualBox en local, l'hyperviseur est de type 2 (installé sur l'OS du PC), la VM tourne sur le PC, qui devient l'hôte ; on voit alors `VirtualBoxVM.exe` dans le Gestionnaire des tâches du PC.

W2. La personne qui a configuré la VM, dans la limite des ressources de l'hôte. Windows voit du matériel virtuel : le nom du processeur est dérivé du processeur physique du serveur, mais le nombre de cœurs est celui des vCPU alloués.

W3. Un processus est un programme en cours d'exécution ; un service est un programme géré par Windows qui tourne en arrière-plan, et son exécution est un processus. svchost.exe est un hôte générique qui héberge des groupes de services. Raisons : économiser les ressources, et isoler les pannes (un groupe qui plante n'emporte pas les autres).

W4. Le bureau, la barre des tâches et le menu Démarrer disparaissent, mais Windows continue de tourner. Le PID identifie une instance de processus, pas le programme : chaque lancement en reçoit un nouveau.

W5. SYSTEM, puis administrateur (après élévation), puis NETWORK SERVICE et LOCAL SERVICE, tous deux à privilèges réduits. LOCAL SERVICE se présente en anonyme sur le réseau, NETWORK SERVICE avec l'identité de l'ordinateur.

W6. TrustedInstaller, propriétaire du fichier, qui seul a le contrôle total ; le groupe Administrateurs n'a que Lecture et exécution. Pour passer outre : `takeown /f <fichier>` (prendre possession) puis `icacls <fichier> /grant Administrateurs:F` (modifier l'ACL).

W7. Avec l'UAC, un administrateur reçoit deux jetons. Le cmd normal utilise le jeton standard, niveau d'intégrité Moyen : le groupe Administrateurs y figure, mais marqué « refus uniquement ». Il faut un cmd élevé (Exécuter en tant qu'administrateur) pour obtenir le jeton complet, niveau Élevé.

W8. Non : le jeton est créé à l'ouverture de session. Il doit fermer puis rouvrir sa session. Ensuite, dans un cmd élevé, l'UAC demande une simple confirmation Oui/Non, sans mot de passe. Dans un cmd non élevé, c'est encore « Accès refusé ».

W9. Le dossier reste : Windows supprime l'identité (compte et SID), pas les données. Un nouveau « bob » reçoit un nouveau SID ; les ACL de l'ancien dossier pointent vers le SID orphelin, donc pas d'accès automatique.

W10. Le groupe intégré Utilisateurs a Lecture et exécution : il peut lancer les programmes système, pas les modifier. `(F)` = contrôle total ; `(M)` = modification, c'est-à-dire tout sauf changer les permissions ou prendre possession.

W11. `ls` est un alias de `Get-ChildItem`, ajouté pour faciliter la transition des utilisateurs Unix. Ce n'est pas le programme `ls` de Linux, donc ses options n'existent pas. Le tube de PowerShell transmet des objets avec leurs propriétés ; celui de cmd (et de bash) transmet du texte.

W12. L'empreinte vérifie l'intégrité, à condition de la comparer à une valeur de référence obtenue par un canal fiable (si le hash et le fichier viennent de la même page piratée, cela ne prouve rien). La signature vérifie l'authenticité de l'éditeur et que le fichier n'a pas changé depuis la signature. Aucune ne garantit que le logiciel est inoffensif.

W13. Store : contrôles de certification de Microsoft, source la plus encadrée. winget : manifestes validés et hash vérifié, moins contrôlé que le Store. .exe d'un site : seul l'utilisateur vérifie, c'est le plus risqué. 7-Zip s'installe dans `C:\Program Files`, pour toute la machine, d'où l'élévation ; un logiciel installé dans le profil (user scope) n'en a pas besoin.

W14. (1) Le snapshot dépend du disque et du stockage de la VM : si le stockage casse, tout est perdu, alors qu'une sauvegarde est une copie indépendante, ailleurs. (2) Conservé longtemps, il grossit et consomme du stockage. (3) Il dégrade les performances. On le supprime ou on le consolide après le test.

W15.

Mode	Bloc-notes	Durée d'activité	Monde physique
Shut Down Guest	Texte perdu (Windows prévient)	Repart de zéro, sauf démarrage rapide	Bouton Arrêter logiciel
Power Off	Texte perdu	Remise à zéro	Débrancher la prise
Suspend	Texte toujours là	Continue	Mise en pause

Shut Down Guest a besoin de VMware Tools : c'est par lui que l'hyperviseur demande à Windows de s'éteindre.

W16. 41 (Kernel-Power) : le système a redémarré sans arrêt correct. 6008 : Windows constate que l'arrêt précédent était inattendu. Les deux signalent une coupure brutale. 1074 : un utilisateur ou un processus a demandé un arrêt ou un redémarrage (arrêt propre). Ce sont des indices à interpréter, pas une preuve absolue.

W17. Il associe localement des noms à des adresses IP et il est consulté avant le DNS. Un attaquant peut y rediriger un nom (une banque) vers un faux serveur : hameçonnage, DNS spoofing local. Confusion à éviter : il ne contient pas l'adresse du serveur DNS.

W18.

1. Faux : System32 contient le 64 bits, SysWOW64 le 32 bits.
2. Faux : masquer évite les erreurs, une case à cocher suffit à afficher ; seuls les droits protègent.
3. Faux : il utilise le magasin local `C:\Windows\WinSxS`.
4. Faux : « Impossible de joindre l'hôte de destination » est compté comme une réponse ; il faut lire les lignes.
5. Faux : sans `/F`, c'est une demande ; `/F` force.
6. Faux : il est créé à la première ouverture de session.

W19. Dans un cmd élevé :

```
net user alice Alice#2026 /add
net localgroup Administrateurs
net localgroup Administrateurs alice /add
net localgroup Administrateurs
net localgroup Administrateurs alice /delete
net user alice /delete
```

Dans un cmd non élevé, la première commande répond « Accès refusé » (erreur système 5). À la fin : si alice n'a jamais ouvert de session, il ne reste rien, car le profil n'a jamais été créé ; sinon `C:\Users\alice` reste.

W20.

```
notepad
tasklist | find "notepad"
taskkill /PID <PID>
taskkill /PID <PID> /F
```

Sans `/F`, le Bloc-notes reçoit une demande de fermeture (il peut proposer d'enregistrer). Avec `/F`, il disparaît immédiatement, comme avec Fin de tâche. Les deux outils agissent sur les mêmes processus ; la ligne de commande peut en plus servir dans un script.

W21.

```
dir /q C:\
dir /a C:\
dir /a:h C:\
```

En PowerShell : `(Get-Acl C:\ProgramData).Owner`, `(Get-Acl C:\Windows\System32).Owner`, `Get-ChildItem C:\ -Force -Hidden`. ProgramData

: SYSTEM (affiché `AUTORITE NT\Système`). System32 : TrustedInstaller (affiché `NT SERVICE\TrustedInstaller`). Le libellé diffère de l'interface graphique, le propriétaire est le même.

W22.

```
tasklist | find /c "svchost"
tasklist /svc | find "svchost"
```

W23.

```
winget search notepad++
winget install --id Notepad++.Notepad++ -e
winget list
winget uninstall --id Notepad++.Notepad++
```

Trois vérifications : menu Démarrer ; Paramètres, Applications ; `winget list`. L'UAC apparaît à l'installation, car le logiciel va dans `C:\Program Files`.

W24. `winver` ; Paramètres, Système, À propos de ; ou `systeminfo`. Côté VM : VM, Settings, Hardware (ou Edit Settings dans vSphere). Snapshot : VM, Snapshot, Take Snapshot, puis Snapshot Manager (Ctrl + M). Les valeurs sont identiques des deux côtés.

Linux

L1. PID : identifiant d'une instance de processus. PPID : PID du processus parent. Thread : unité d'exécution à l'intérieur d'un processus (colonne NLWP). PID 1 : systemd, ancêtre de tous les processus. PID 2 : kthreadd, parent des threads du noyau (entre crochets).

L2. Il devient orphelin, est adopté par systemd et continue de tourner. Un zombie est l'inverse : un processus déjà terminé dont le parent n'a pas encore lu le code retour.

L3. Arrêter : SIGSTOP, état T (en pause). Continuer : SIGCONT, état S. Terminer : SIGTERM (15), le processus disparaît après s'être éventuellement nettoyé. Tuer : SIGKILL (9), le noyau le termine. Si SIGTERM est ignoré, le programme continue : on passe à `kill -9`.

L4. 15 : accepté, on augmente le nice, donc on baisse la priorité. 5 : refusé (« Permission denied »), car on diminuerait le nice, donc on augmenterait la priorité, ce qui demande sudo.

L5. Des comptes de service. UID inférieur à 1000 ; shell `nologin`, donc personne ne peut s'y connecter ; créés pour faire tourner un service (D-Bus, journaux, résolution DNS) avec des droits limités.

L6. `-` : fichier ordinaire. `rws` : le propriétaire (root) lit, écrit, exécute, et le `s` est le bit setuid. `r-x` pour le groupe root, `r-x` pour les autres. Le programme s'exécute avec les droits de son propriétaire, root : il peut ainsi écrire dans `/etc/shadow`, qu'un utilisateur ordinaire ne peut pas modifier.

L7. Dossier, `rwx` pour tous, et `t` = sticky bit (1777). Tout le monde peut y écrire. Seuls alice (propriétaire du fichier), le propriétaire du dossier et root peuvent supprimer le fichier d'alice.

L8. 750, 644, 700, 1777. Puis `rwxr-xr-x`, `rw-r-----`, `rw-------`.

L9. Lire : oui, `r` sur le fichier pour les autres et `x` sur le dossier pour le traverser. Modifier : non, pas de `w` sur le **fichier** pour les autres. Supprimer : non, pas de `w` sur le **dossier** pour les autres.

L10. Modifier : toujours non, le fichier n'a pas changé. Supprimer : oui, il a maintenant `w` sur le dossier, et il n'y a pas de sticky bit.

L11. `/etc/passwd` doit rester lisible par tous (correspondance entre UID et noms) ; son 2e champ vaut `x`. Le 2e champ de `/etc/shadow` est le hachage du mot de passe, lisible par root seulement, pour empêcher une attaque par dictionnaire hors ligne. Un hachage ne se déchiffre pas : il n'y a pas d'opération inverse.

L12. `%sudo` : les membres du groupe sudo. Premier `ALL` : depuis toute machine. `(ALL:ALL)` : en tant que tout utilisateur et tout groupe. Dernier `ALL` : toute commande. sudo demande le mot de passe de l'utilisateur lui-même : on prouve qui agit sans partager celui de root.

L13. Sans `-a`, `-G` remplace la liste : etudiant perd ses autres groupes supplémentaires. Les groupes sont lus à la connexion : il faut fermer et rouvrir la session.

L14. Le compte est supprimé, pas le dossier : il appartient à l'UID 1002, qui n'a plus de nom. `adduser` attribue le premier UID libre : carla peut recevoir 1002 et hériter des fichiers de bob. Pour l'éviter : `deluser --remove-home`.

L15. `ps aux | head -5` : la sortie de `ps` devient l'entrée de `head`. `ls -l /etc > liste.txt` : écrit dans le fichier en l'écrasant. `date >> liste.txt` : ajoute à la fin. Comptage : `grep -c nologin /etc/passwd`.

L16. Il télécharge la liste à jour des paquets disponibles ; il n'installe rien et ne met rien à jour. `remove` enlève le programme et garde la configuration ; `purge` enlève aussi la configuration ; `autoremove` retire les dépendances devenues inutiles. `rc` : paquet supprimé (r), configuration conservée (c).

L17. apt (mainteneurs de la distribution, paquets signés), Snap (éditeur et magasin Canonical, confinement), .deb d'un site (l'éditeur seul, scripts en root), `curl ... | sudo bash` (personne, exécution immédiate en root). xz : une porte dérobée est passée par le circuit open source normal ; une source vérifiée n'est pas infaillible, et les mainteneurs ne relisent pas tout le code amont.

L18. Il empêche deux modifications simultanées de la base des paquets (corruption, fichiers écrasés, scripts qui se superposent). Ne jamais supprimer le fichier de verrou. `sudo lsof /var/lib/dpkg/lock-frontend` montre le processus qui le tient.

L19.

1. Faux : caché n'est pas protégé, seuls les droits protègent.
2. Faux : le dossier personnel, `/home/<utilisateur>`.
3. Faux : plus petit veut dire plus prioritaire.
4. Faux : le noyau protège le PID 1, la commande ne fait rien.
5. Faux : `adduser` le crée à la création du compte, depuis `/etc/skel`.
6. Faux : elle prouve l'intégrité par rapport à une référence fiable, rien de plus.
7. Faux : il est verrouillé (`L`) sur Ubuntu ; il en a un sur Debian.

L20.

```
sudo adduser alice
sudo adduser carl
su - alice
echo test > ~/secret.txt
ls -ld ~ ; ls -l ~/secret.txt
exit
su - carl
cat /home/alice/secret.txt
exit
sudo setfacl -m u:carl:rx /home/alice
sudo setfacl -m u:carl:r /home/alice/secret.txt
getfacl /home/alice
sudo deluser --remove-home alice
sudo deluser --remove-home carl
```

Carl obtient « Permission denied » : le dossier est en 750 (Ubuntu) ou 700 (Debian), les autres ne peuvent pas le traverser. L'ACL donne l'accès à carl seul, sans ouvrir aux autres. Si `setfacl` manque : `sudo apt install acl`.

L21.

```
yes > /dev/null &
echo $!
pgrep yes
ps -o pid,ppid,stat,%cpu,cmd -p <PID>
kill -STOP <PID>
ps -o pid,stat,cmd -p <PID>
kill -CONT <PID>
kill <PID>
```

Le parent est bash, le CPU est proche de 100 % d'un cœur, l'état passe à T pendant la pause. `kill` sans option envoie SIGTERM.

L22.

```
ps -e --no-headers | wc -l
ps -eo pid,ppid,user,%mem,comm --sort=-%mem | head -6
ps -eo user= | sort | uniq -c
```

L23. `pstree -p | head -30` ou `ps -ef --forest` ; généalogie typique : systemd, gestionnaire de session, gnome-terminal-server, bash. `echo $$` puis `echo $!`.

L24.

```
sudo apt update
sudo apt install tree
command -v tree
dpkg -L tree
dpkg -S /usr/bin/tree
sudo apt remove tree
dpkg -l tree
sudo apt purge tree
```

Après `remove`, `dpkg -l` peut afficher `rc` si le paquet avait des fichiers de configuration ; après `purge`, plus rien.

L25.

```
apt download htop
sha256sum htop_*.deb
dpkg-deb --info htop_*.deb
dpkg-deb --contents htop_*.deb
```

L26. `cat /etc/os-release`, `uname -r`, `free -h`, `nproc`, `lscpu`, comparés à VM, Settings, Hardware. `free -h` affiche un peu moins : une partie de la mémoire est réservée au noyau, et il compte en Gio alors que vSphere parle en Go.

Exercices croisés

X1. Voir le tableau d'équivalences de la section Terminal et bash.

X2.

Point	Windows	Linux
Création du dossier personnel	À la première ouverture de session	À la création du compte par adduser (`/etc/skel`)
Après suppression du compte	Dossier conservé, SID orphelin	Dossier conservé, UID sans nom
Élévation	UAC : élève un jeton	sudo : élève une commande ; polkit en graphique
Mot de passe demandé	Aucun pour un administrateur (Oui/Non) ; celui d'un administrateur pour un compte standard	Celui de l'utilisateur lui-même
Stockage des mots de passe	Hachages dans la base SAM	Hachages dans `/etc/shadow`

X3. Le noyau Linux, d'après les droits du dossier `/etc`. TrustedInstaller, propriétaire du fichier (les administrateurs n'ont que RX). L'absence d'élévation : le jeton du cmd n'est pas élevé, « Accès refusé », erreur système 5.

X4. N'accorder que les droits nécessaires, le temps nécessaire. Windows : retirer etudiant du groupe Administrateurs après le test (`net localgroup Administrateurs etudiant /delete`). Linux : `sudo gpasswd -d etudiant sudo` ; les services tournent sous des comptes dédiés sans connexion possible. But : limiter les dégâts d'une erreur ou d'un compte compromis.

Corrigés Web

M1. Les pages sont des fichiers envoyés tels quels par le serveur, sans calcul ni base de données. Limite : un produit = un fichier. Le prix figure dans la fiche, le catalogue et le comparatif : un changement doit être répété partout, avec le risquExercices Webe d'afficher deux prix différents. C'est ingérable à 500 produits.

M2. Un chemin relatif se calcule depuis l'emplacement de la page qui le contient ; un chemin absolu est une adresse complète (`https://...`). `images/casque.jpg` : sous-dossier `images` du dossier de la page. `../images/casque.jpg` : on remonte d'un dossier, puis `images`.

M3. Le lien s'ouvre dans un nouvel onglet, la boutique reste ouverte dans l'onglet d'origine. `noopener` empêche la page ouverte d'accéder à la page d'origine par `window.opener` et de la rediriger vers un faux site ; `noreferrer` n'envoie pas l'adresse de la page d'origine.

M4. `connexion.html?nom=bob&mdp=abc123` : les données sont dans l'URL, après `?`, séparées par `&`. Traces : historique, favoris, journaux du serveur et des proxys, liens partagés. POST les place dans le corps de la requête, mais ne chiffre rien : il faut HTTPS pour le trajet et une validation côté serveur.

M5.

1. Faux : il masque seulement l'affichage ; c'est HTTPS qui chiffre le trajet.
2. Faux : contrôle du navigateur, contournable dans F12 ou en envoyant la requête sans le formulaire.
3. Faux : il n'est pas affiché, mais il est envoyé et visible dans le code source.
4. Faux : il reste dans le code envoyé. On peut masquer, pas cacher.
5. Faux : la page s'affiche avec le style par défaut ; l'erreur 404 n'apparaît que dans F12 (Réseau, Console).

M6. Commentaires HTML ; liens et ressources externes ; noms de dossiers et de fichiers ; champs et méthode du formulaire, champs `hidden` ; adresses e-mail, noms ou versions d'outils. Jamais dans un commentaire : mots de passe, clés d'API, adresses de serveurs internes (ou notes de sécurité, chemins de sauvegarde).

M7. Le DOM, c'est-à-dire la page en mémoire dans mon navigateur. Ni le fichier HTML ni le serveur ne changent ; personne d'autre ne le voit, car chaque visiteur a sa propre copie. F5 recharge la page depuis le serveur : retour à l'état d'origine.

M8. 501 : j'ai modifié la variable en mémoire, le clic ajoute 1. Le visiteur contrôle son navigateur ; un prix, un stock ou un droit d'accès se décident côté serveur.

M9. `innerHTML` fait lire la saisie comme du HTML : le navigateur ne distingue pas ce que j'ai écrit de ce que le visiteur a saisi, et exécute le code. `textContent` l'insère comme du texte : `<` et `>` s'affichent tels quels. Test : `<img src=x onerror="alert('XSS')">`. Une balise `<script>` insérée par `innerHTML` ne s'exécute pas, d'où ce détour.

M10. Apache : serveur web, envoie les fichiers et lance PHP. PHP : langage exécuté sur le serveur pour générer les pages. MySQL/MariaDB : stockage des données. phpMyAdmin : application PHP pour administrer la base. Avec `file://`, aucun serveur n'intervient : PHP n'est pas exécuté.

M11. Base : ensemble de tables. Table : ensemble de données de même nature. Colonne : un champ (prix). Ligne : un enregistrement (un produit). Clé primaire : identifie chaque ligne de façon unique et non nulle. Deux ids identiques : la base refuse l'insertion (« Duplicate entry »).

M12. Sans `WHERE`, la première met le prix de **toutes** les lignes à 1, la seconde supprime **toutes** les lignes. Ce n'est pas une erreur de syntaxe. `DELETE` laisse la table et sa structure ; `DROP TABLE` supprime la table entière.

M13. Il voit uniquement le HTML produit. Il ne voit ni le code PHP, ni la requête SQL, ni le mot de passe de la base. `htmlspecialchars` encode les caractères spéciaux (`<` devient `<`) pour que les données s'affichent comme du texte, ce qui empêche l'exécution d'un script injecté.

M14. Clé étrangère : relie une ligne à une ligne d'une autre table, et la base refuse une référence inexistante. JOIN : réunit dans un résultat les données de deux tables liées. Index : accélère les recherches sur une colonne, au prix d'espace disque et d'écritures plus lentes.

M15. Présentation (interface : HTML, CSS, JavaScript dans le navigateur), application (traitement et règles métier, authentifie et autorise : PHP), données (stockage : MySQL). Tout passe par l'application, qui protège les données contre les attaques venant de la présentation. Utilisateur technique : le compte avec lequel l'application accède à la couche de données.

M16. Code SMS : authentification. Commandes d'un autre : autorisation. Empreinte : authentification. Modifier un prix : autorisation. L'authentification vient toujours en premier.

M17. La session est la série d'actions du visiteur ; le serveur lui attribue un session ID aléatoire ; le navigateur le stocke dans un cookie et le renvoie à chaque requête, ce qui évite de se réauthentifier. Cookies supprimés : le serveur ne reconnaît plus le visiteur, il faut se reconnecter. Navigation privée : les cookies disparaissent à la fermeture, la session aussi.

M18. L'écoute du trafic (man-in-the-middle) et l'usurpation DNS. Le certificat prouve que le serveur est bien le propriétaire du nom de domaine. Contenu : domaine, titulaire, autorité de certification, signature numérique, dates d'émission et d'expiration, clé publique, sous-domaines. Il est signé par une autorité de certification reconnue par les navigateurs : un faux ne serait pas reconnu.

M19. Native : application propre à un OS (Bloc-notes). Web : via un navigateur (webmail). Serveur : sans interface, utilisée par d'autres via des API (MySQL). Logiciel système : en arrière-plan (un service). Site dynamique : page construite à la demande par une application serveur, en général depuis une base. Statique : pas de base ni de plugins à attaquer, mais chaque changement se fait à la main.

M20. La requête devient `WHERE id = 1 OR 1=1` : condition toujours vraie, toutes les lignes sont concernées ; avec une table de clients, tout serait exposé. Point commun : une donnée utilisateur est interprétée comme du code. Protections : requêtes préparées, validation des entrées côté serveur, échappement des caractères spéciaux.

M21. Le moindre privilège. Avec `minishop_app`, l'attaquant peut lire mais ni modifier ni détruire ; avec root il peut tout. C'est MySQL, le serveur de base, qui refuse (« UPDATE command denied »), ni PHP ni phpMyAdmin.

M22.

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>MiniShop - Contact</title>
```

```
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <h1>Contact</h1>
  <a href="index.html">Accueil</a>
  <img src="images/logo.png" alt="Logo MiniShop">
  <form method="post" action="contact.html">
    <label for="nom">Nom</label>
    <input type="text" id="nom" name="nom" required>
    <label for="email">E-mail</label>
    <input type="email" id="email" name="email" required>
    <label for="mdp">Mot de passe</label>
    <input type="password" id="mdp" name="mdp">
    <button type="submit">Envoyer</button>
  </form>
</body>
</html>
```

M23.

```
<table>
  <caption>Comparatif</caption>
  <thead><tr><th>Produit</th><th>Prix</th><th>Stock</th></tr></thead>
  <tbody>
    <tr><td>Casque</td><td>49,90 €</td><td>12</td></tr>
    <tr><td>Souris</td><td>19,90 €</td><td>30</td></tr>
  </tbody>
</table>
```

```
table { border-collapse: collapse; }
th, td { border: 1px solid #333; padding: 8px; }
th { background-color: #9A6B00; color: white; }
```

M24.

```
<input type="text" id="saisie">
<button id="bouton">Valider</button>
<p id="message"></p>
<script src="js/app.js"></script>
```

```
const bouton = document.getElementById("bouton");
const saisie = document.getElementById("saisie");
const message = document.getElementById("message");
bouton.addEventListener("click", function () {
  message.textContent = "Bonjour " + saisie.value;
});
```

Version vulnérable : `message.innerHTML = ...` ; la saisie `<img src=x onerror="alert('XSS')">` déclenche l'alerte. Avec `textContent`, elle s'affiche comme du texte.

M25.

```
CREATE TABLE clients (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(100) NOT NULL,
  email VARCHAR(150) NOT NULL
);
INSERT INTO clients (nom, email) VALUES ('Dupont', 'dupont@example.be');
INSERT INTO clients (nom, email) VALUES ('Martin', 'martin@example.be');
SELECT * FROM clients ORDER BY nom;
```

```
UPDATE clients SET email = 'd.dupont@example.be' WHERE id = 1;
DELETE FROM clients WHERE id = 2;
```

M26.

```
<?php
$conn = new mysqli("localhost", "minishop_app", "mot-de-passe", "minishop");
if ($conn->connect_error) {
    die("Service momentanément indisponible");
}
$conn->set_charset("utf8mb4");
$resultat = $conn->query("SELECT nom, email FROM clients ORDER BY nom");
echo "<ul>";
while ($c = $resultat->fetch_assoc()) {
    echo "<li>" . htmlspecialchars($c["nom"]) . " - " .
htmlspecialchars($c["email"]) . "</li>";
}
echo "</ul>";
?>
```

M27.

```
$nom = $_GET["nom"] ?? "";
$stmt = $conn->prepare("SELECT nom, prix FROM produits WHERE nom = ?");
$stmt->bind_param("s", $nom);
$stmt->execute();
$resultat = $stmt->get_result();
```

Le `?` remplace la valeur, sans guillemets autour ; `"s"` indique une chaîne (`"i"` pour un entier). La structure de la requête part avant la valeur, qui ne peut donc plus la modifier.