Plan de formation Linux DSI

PARTIE 1 — Contexte général

1.1 Les trois incidents à l'origine du plan

Incident	Impact et cause
Injection SQL	1 200 fiches clients exfiltrées depuis Monica CRM. Cause : requêtes SQL concaténées sans paramètre préparé, et compte SQL à privilèges d'administration utilisé par l'application.
Panne NetPay	6 heures d'indisponibilité du service Node.js après un reboot. Cause : absence de gestionnaire de processus et de monitoring préventif.
Perte de données	3 semaines de données effacées par une manipulation en production. Cause : pas de backup automatisé, restauration partielle depuis une sauvegarde manuelle de 45 jours.

1.2 Les applications du fil rouge

Monica CRM	Strapi CMS
Laravel (PHP). CRM open source : contacts, relations, rappels. Accès : https://crm.valtrix.local. **Base :** valtrix_crm (MariaDB).	Node.js 20 LTS. CMS headless : API REST, médias, plugins. Accès : https://api.valtrix.local. **Base :** valtrix_strapi (PostgreSQL).

1.3 Environnement local VMware

Réseau 192.168.74.0/24. Certificats auto-signés générés avec OpenSSL.

Domaines locaux : crm.valtrix.local, api.valtrix.local, db.valtrix.local.

Toutes les VM sous Ubuntu Server 24.04 LTS, sauf vm-kali (Kali Linux).

VM	IP	Ressources	Rôle	Services
vm-app	10	4 Go RAM 2 vCPU 40 Go	Serveur applicatif	Nginx, Monica CRM, Strapi, PM2, Docker et Compose, WireGuard
vm-db	20	2 Go RAM 2 vCPU 30 Go	Serveur de données	PostgreSQL 15 (Strapi), MariaDB (Monica)
vm-kali	30	2 Go RAM 1 vCPU 25 Go	Audit offensif (phases 3 et suivantes)	Nmap, OWASP ZAP, SQLMap, Nikto, Wireshark

1.4 Flux réseau autorisés en fin de formation

Flux	Description
vm-app → vm-db (5432)	Strapi se connecte à PostgreSQL
vm-app → vm-db (3306)	Monica CRM se connecte à MariaDB
vm-db → vm-app (SSH)	Envoi des sauvegardes par rsync (séance 15)
vm-kali → vm-app	Tests offensifs (phase 3 uniquement)
Hôte → vm-app (443)	crm.valtrix.local et api.valtrix.local
vm-app ⛔ vm-kali	vm-app ne doit jamais initier de connexion vers vm-kali (règle ufw)

PARTIE 2 — La méthode

2.1 Règles transversales

- **Vérifier avant d'agir** : où suis-je, qu'est-ce qui existe déjà, quel est l'état du service.
- **Diagnostiquer avant de corriger** : une hypothèse écrite vaut mieux qu'une commande qui marche par hasard.

- **Session de secours** : toute modification de SSH, ufw ou Nginx se fait avec une seconde session déjà ouverte.
- **Snapshot VMware** avant les sous-TP à risque : 4b, 12b, 13a, 11b (scan actif), 15b.
- **Dégressivité des indices** : en phase 1, aucune commande ni nom d'outil au-delà du rappel. En phase 2, nom de l'outil et du fichier clé, pas la syntaxe. En phases 3 et 4, outils et fichiers fournis, consignes toujours orientées résultat.

2.2 Rapport

À reprendre pour chaque scénario ou étape d'une mission. Le contenu est **copié depuis le terminal**, jamais réécrit de mémoire. Les erreurs ne sont pas effacées.

```
## Étape X — Titre

État avant d'agir :
$ (commande de vérification)
(résultat réel copié-collé)

Action réalisée :
$ (commande utilisée)

Vérification après :
$ (commande de vérification)
(résultat réel copié-collé)

Si erreur :
- Message obtenu : (copié)
- Mon hypothèse, écrite avant de corriger : …
- Ce que j'ai vérifié pour la confirmer : …
- Correction appliquée et vérification : …
```

À la fin de chaque sous-TP, trois lignes rédigées avec ses propres mots : ce qui a été le plus difficile, ce qui a été compris, ce qui reste flou.

PARTIE 3 — Le fil rouge : le dépôt valtrix-dsi

Tout au long, un projet unique et continu est construit : le dépôt Git valtrix-dsi. Ce dépôt accumule tous les fichiers de configuration, scripts, rapports et procédures produits. À la fin, il constitue la documentation opérationnelle complète de l'infrastructure.

```
valtrix-dsi/
├── phase1/   ← rapports, configs SSH, ufw, certificats OpenSSL
├── phase2/   ← configs Nginx, scripts SQL, configs PM2
├── phase3/   ← Dockerfile, docker-compose.yml, rapports Nmap/ZAP, audit Lynis
└── phase4/   ← configs WireGuard, guide 2FA, scripts backup/cron, Plan PCA
```

Tableau de progression

Séance	Thème	Sous-TP	Livrable
1	Prise en main Linux	1 : mission-pont	phase1/arborescence.md
2	Utilisateurs & services	2a comptes/groupes 2b sudo/systemd	phase1/users-services.md
3	SSH & réseau	3a clés SSH 3b ports et HTTP	phase1/ssh-config/
4	SSL & firewall	4a PKI 4b ufw	phase1/ssl-firewall/
5	Nginx & HTTPS	5a vhost http 5b HTTPS et reverse proxy	phase2/nginx-config/
6	Bases de données	6a rôles et bases 6b accès réseau et dump	phase2/sql-scripts/
7	Git & Monica	7a Git 7b déploiement Laravel 7c requêtes SQL	Dépôt, phase2/sql-scripts/
8	PM2 & Strapi	8a déploiement 8b résilience	phase2/pm2-configs/
9	Docker	9a bases 9b Dockerfile Strapi	phase3/Dockerfile-strapi
10	Compose & OWASP	10a Compose 10b injection SQL	phase3/docker-compose.yml
11	XSS, CSRF & ZAP	11a XSS/CSRF 11b scan ZAP	phase3/rapport-zap/
12	Nmap & Fail2ban	12a audit	phase3/security/

Séance	Thème	Sous-TP	Livrable
		12b durcissement SSH	
13	Durcissement	13a Lynis 13b headers et bases	phase3/audit-lynis/
14	VPN & 2FA	14a WireGuard 14b 2FA	phase4/vpn-2fa/
15	Backup & PCA	15a automatisation 15b incident, restauration, PCA	phase4/backup-plan/

PHASE 1 — Fondations Linux et réseau

Objectif global : naviguer sans se perdre, maîtriser fichiers, permissions, utilisateurs, SSH par clé, PKI locale et firewall, sur vm-app et vm-db.

Séance 1 — Prise en main de Linux

Machine : vm-app.

Rappel : support « Rappel Linux, fichiers, chemins, permissions » (système de fichiers, structure d'une commande, chemins absolus et relatifs, lecture d'une ligne de détails de fichier).

Sous-TP 1 — Mission-pont : préparer l'arborescence de vm-app

Objectif : appliquer les acquis des TP précédents à la structure définitive de Valtrix.

Mission Valtrix :

1. Vérifier l'état de départ : où suis-je, que contiennent `/srv`, `/var/log` et la racine `/`.
2. Créer `/srv/valtrix/crm` et `/srv/valtrix/api`, `/var/log/valtrix/{crm,api}`, `/backup/valtrix/{crm,api,db}`.
3. Fixer les droits : `/srv/valtrix` en 755, `/srv/valtrix/crm` et `/srv/valtrix/api` en 750.
4. Vérifier chaque dossier créé et ses droits, sans se fier à l'absence d'erreur.
5. Documenter dans `arborescence.md`, avec l'arborescence complète et les droits.

Livrable : `phase1/arborescence.md`.

Séance 2 — Utilisateurs, groupes et services

Machines : vm-app et vm-db.

Contexte : Monica et Strapi ne tournent jamais sous root. Chaque application a son compte dédié, sans shell interactif.

Sous-TP 2a — Comptes, groupes et propriété

Mission Valtrix :

1. Sur vm-app, créer le groupe `valtrix-grp`.
2. Créer les comptes système `monica-app` et `strapi-app`, sans shell de connexion interactive.

3. Attribuer la propriété : `monica-app:valtrix-grp` sur `/srv/valtrix/crm`, `strapi-app:valtrix-grp` sur `/srv/valtrix/api`. Vérifier le résultat.

4. Créer le compte humain `admin-dsi`. Vérifier son identité complète.

5. Sur vm-db, créer également le compte humain `admin-dsi`, vérifier son identité complète.

6. Vérifier qu'aucun des comptes applicatifs ne peut ouvrir de session interactive locale.

Livrable : première partie de `phase1/users-services.md`.

Sous-TP 2b — sudo limité, systemd, journaux, paquets

Rappel : `visudo` et fichiers dans `/etc/sudoers.d/`, `systemctl`, `journalctl`, `apt`.

Mission Valtrix :

1. Donner à `admin-dsi` un sudo limité à `systemctl`. **Analyser** : que peut-il faire d'autre avec cette autorisation ? Documenter la limite et proposer une restriction plus fine (sous-commandes et unités précises).

2. Installer SSH sur vm-app et vm-db, vérifier qu'il tourne et qu'il est activé au démarrage.

3. Redémarrer vm-app et vérifier que SSH est bien revenu seul.

4. Lire les journaux SSH sur une plage de dates précise.

5. Mettre à jour la liste des paquets, lister les paquets installés, désinstaller un paquet de test.

Livrable : `phase1/users-services.md` complet.

Séance 3 — SSH et réseau

Machines : poste hôte, vm-app, vm-db.

Rappel : support « Module de renforcement SSH » (service, configuration, clés).

Sous-TP 3a — Connexion par clé vers vm-app et vm-db

Mission Valtrix :

1. Sur le poste hôte, générer une paire de clés ED25519 dédiée à Valtrix.

2. Faire accepter la clé publique par vm-app, puis par vm-db.

3. Créer le fichier de configuration SSH avec les alias `vm-app` et `vm-db`.

4. Se connecter aux deux machines par leur alias, sans mot de passe.

5. **Seulement après** validation de la connexion par clé, interdire l'authentification par mot de passe sur les deux serveurs, avec une session de secours ouverte.

6. Vérifier le refus d'une tentative par mot de passe.

Livrable : `phase1/ssh-config/` (fichier de configuration client, procédure documentée pour un collègue).

Sous-TP 3b — Ports, réseau et HTTP

Rappel : ports applicatifs (22, 80, 443, 1337, 5432, 3306), méthodes et codes de statut HTTP.

Mission Valtrix :

1. Lister les ports ouverts sur vm-app, documenter chaque port avec son service.
2. Faire de même sur vm-db.
3. Lancer un serveur HTTP temporaire sur vm-app, l'interroger **depuis le poste hôte**, observer le code de statut et les en-têtes.
4. Provoquer volontairement une erreur (port fermé, mauvaise adresse) et noter la différence de comportement.

Livrable : rapport des ports dans `phase1/ssh-config/`.

Séance 4 — SSL/TLS et firewall

Machines : vm-app et vm-db.

Travail de recherche préalable : prérequis de Let's Encrypt ; raison de son acceptation par les navigateurs ; challenges HTTP-01 et DNS-01 ; CA intermédiaire ; cas d'usage professionnels du certificat auto-signé ; HSTS et ses risques.

Sous-TP 4a — Firewall ufw

Rappel : politique par défaut, règles par port, par IP source, par interface, ordre des règles.

Mission Valtrix :

1. Sur vm-app : tout refuser en entrée par défaut, autoriser SSH, HTTP et HTTPS.
2. Sur vm-db : tout refuser, autoriser SSH, PostgreSQL (5432) et MariaDB (3306) depuis vm-app uniquement.
3. Tester chaque règle depuis l'extérieur : un flux autorisé passe, un flux non autorisé échoue.
4. Documenter chaque règle et son justificatif.

Livrable : règles ufw documentées dans `phase1/ssl-firewall/`.