Checklist Sécurité AUDIT SITE WEB WORDPRESS

Ayi NEDJIMI Consultants
ayinedjimi-consultants.fr
v1.0 — 04/04/2026 · 95 controles

Sommaire

1.0 — INSTALLATION & CONFIGURATION DE BASE

1.1.1 *Version WordPress Actuelle et Supportée*

MITRE ATT&CK : T1190

DESCRIPTION :

WordPress doit être maintenu à jour avec la dernière version stable. Les versions obsolètes contiennent des vulnérabilités critiques exploitées activement par les attaquants. Une mise à jour dans les 30 jours suivant la publication est recommandée.

```
# Vérification avec WPScan
wpscan --url https://example.com --enumerate vp --no-banner
# Vérification via wp-cli
wp core check-update
wp core version
# Vérification manuelle
curl -s https://example.com/ | grep -o 'content="WordPress [0-9.]*"'
```

REMÉDIATION :

1. Effectuer une sauvegarde complète avant mise à jour
2. Tester en environnement de staging

```
wp core update
wp core update-db
```

REMÉDIATION :

1. Activer les mises à jour automatiques mineures :

```
// Dans wp-config.php
define( 'WP_AUTO_UPDATE_CORE', 'minor' );
```

VALEUR PAR DÉFAUT :

Mises à jour automatiques mineures activées depuis WordPress 3.7

1.1.2 *Version PHP Supportée et Sécurisée*

MITRE ATT&CK : T1190

DESCRIPTION :

PHP doit être maintenu dans une version supportée officiellement (PHP 8.0+). Les versions obsolètes de PHP présentent des vulnérabilités de sécurité critiques et des problèmes de performance.

```
# Via wp-cli
wp eval "echo PHP_VERSION;"
# Via WPScan
wpscan --url https://example.com --enumerate vp --no-banner | grep -i php
# Test direct
curl -s https://example.com/wp-content/themes/twentytwentyone/index.php
```

REMÉDIATION :

1. Mettre à niveau vers PHP 8.1+ minimum
2. Configurer php.ini sécurisé :

```
expose_php = Off
allow_url_fopen = Off
allow_url_include = Off
display_errors = Off
log_errors = On
```

VALEUR PAR DÉFAUT :

Dépend de l'hébergeur, souvent PHP 7.4 ou antérieur

1.1.3 *Configuration wp-config.php Sécurisée*

MITRE ATT&CK : T1005

DESCRIPTION :

Le fichier wp-config.php contient les informations de connexion à la base de données et les clés de sécurité. Il doit être protégé contre l'accès web direct et configuré avec les bonnes pratiques de sécurité.

```
# Test accès direct
curl -s https://example.com/wp-config.php
curl -s https://example.com/wp-config.php.bak
# Vérification permissions
ls -la wp-config.php
# Vérification contenu (sur serveur)
grep -E "(DB_PASSWORD|AUTH_KEY)" wp-config.php
```

REMÉDIATION :

1. Déplacer wp-config.php hors de web root ou protéger via .htaccess :

```
<Files wp-config.php>
order allow,deny
deny from all
</Files>
```

REMÉDIATION :

1. Définir les permissions appropriées :

```
chmod 600 wp-config.php
```

REMÉDIATION :

1. Ajouter les constantes de sécurité essentielles

VALEUR PAR DÉFAUT :

Permissions 644, souvent accessible via web

1.1.4 *Clés de Sécurité SALT Configurées*

MITRE ATT&CK : T1552

DESCRIPTION :

Les clés SALT renforcent la sécurité des cookies et des sessions utilisateurs. Elles doivent être uniques, complexes et changées régulièrement pour prévenir les attaques de décryptage.

```
# Vérification présence des clés
grep -E "(AUTH_KEY|SECURE_AUTH_KEY|LOGGED_IN_KEY|NONCE_KEY)" wp-config.php
# Test avec wp-cli
wp config get AUTH_KEY
wp config get SECURE_AUTH_KEY
```

REMÉDIATION :

1. Générer de nouvelles clés via https://api.wordpress.org/secret-key/1.1/salt/
2. Ajouter dans wp-config.php :

```
define('AUTH_KEY',         'votre-clé-unique-ici');
define('SECURE_AUTH_KEY',  'votre-clé-unique-ici');
define('LOGGED_IN_KEY',    'votre-clé-unique-ici');
define('NONCE_KEY',        'votre-clé-unique-ici');
define('AUTH_SALT',        'votre-clé-unique-ici');
define('SECURE_AUTH_SALT', 'votre-clé-unique-ici');
define('LOGGED_IN_SALT',   'votre-clé-unique-ici');
define('NONCE_SALT',       'votre-clé-unique-ici');
```

REMÉDIATION :

1. Renouveler les clés tous les 6 mois

VALEUR PAR DÉFAUT :

Clés par défaut ou vides lors de l'installation

1.1.5 *Préfixe Table Base de Données Personnalisé*

MITRE ATT&CK : T1190

DESCRIPTION :

Le préfixe par défaut "wp_" facilite les attaques d'injection SQL automatisées. Un préfixe personnalisé ajoute une couche d'obscurité contre les attaques génériques.

```
# Via wp-config.php
grep "table_prefix" wp-config.php
# Via wp-cli
wp config get table_prefix
# Test base de données
mysql -e "SHOW TABLES LIKE 'wp_%'" database_name
```

REMÉDIATION :

1. Avant installation, modifier wp-config.php :

```
$table_prefix = 'xyz123_';
```

REMÉDIATION :

1. Pour une installation existante, utiliser un plugin ou script :

```
wp search-replace "wp_" "xyz123_" --dry-run
wp db export backup.sql
# Puis renommer les tables manuellement
```

VALEUR PAR DÉFAUT :

wp_

1.1.6 *Mode Debug Désactivé en Production*

MITRE ATT&CK : T1005

DESCRIPTION :

Le mode debug révèle des informations sensibles sur la structure du site et les erreurs système. Il doit être désactivé en production pour éviter la divulgation d'informations.

```
# Vérification wp-config.php
grep "WP_DEBUG" wp-config.php
# Test via wp-cli
wp config get WP_DEBUG
# Forcer une erreur pour tester
curl -s https://example.com/?debug=1
```

REMÉDIATION :

1. Dans wp-config.php :

```
define('WP_DEBUG', false);
define('WP_DEBUG_LOG', false);
define('WP_DEBUG_DISPLAY', false);
define('SCRIPT_DEBUG', false);
```

REMÉDIATION :

1. Supprimer le fichier debug.log existant :

```
rm wp-content/debug.log
```

VALEUR PAR DÉFAUT :

WP_DEBUG false

1.1.7 *Édition de Fichiers Désactivée*

MITRE ATT&CK : T1105

DESCRIPTION :

L'éditeur de fichiers intégré à WordPress permet de modifier les thèmes et plugins depuis l'interface d'administration. Cette fonctionnalité représente un risque majeur en cas de compromission d'un compte administrateur.

```
# Vérification wp-config.php
grep "DISALLOW_FILE_EDIT" wp-config.php
# Test interface admin
# Aller dans Apparence > Éditeur de thème
```

REMÉDIATION :

1. Ajouter dans wp-config.php :

```
define('DISALLOW_FILE_EDIT', true);
```

REMÉDIATION :

1. Vérifier que le menu "Éditeur" disparaît de l'admin

VALEUR PAR DÉFAUT :

Édition activée

1.1.8 *Installation de Plugins/Thèmes Contrôlée*

MITRE ATT&CK : T1505

DESCRIPTION :

Restreindre l'installation de plugins et thèmes depuis l'interface d'administration réduit les risques d'installation de code malveillant en cas de compromission.

```
# Vérification wp-config.php
grep -E "(DISALLOW_FILE_MODS|WP_FILESYSTEM_METHOD)" wp-config.php
# Test interface admin - tentative d'installation
```

REMÉDIATION :

1. Pour désactiver complètement :

```
define('DISALLOW_FILE_MODS', true);
```

REMÉDIATION :

1. Pour contrôler les permissions :

```
define('FS_METHOD', 'direct');
define('WP_FILESYSTEM_METHOD', 'direct');
```

VALEUR PAR DÉFAUT :

Installation libre pour les administrateurs

1.1.9 *Révisions de Posts Limitées*

MITRE ATT&CK : T1005

DESCRIPTION :

WordPress conserve par défaut un historique illimité des révisions d'articles. Limiter ce nombre évite l'accumulation de données sensibles et améliore les performances.

```
# Vérification wp-config.php
grep "WP_POST_REVISIONS" wp-config.php
# Via wp-cli
wp config get WP_POST_REVISIONS
# Compter les révisions existantes
wp post list --post_type=revision --format=count
```

REMÉDIATION :

1. Limiter les révisions dans wp-config.php :

```
define('WP_POST_REVISIONS', 3);
```

REMÉDIATION :

1. Nettoyer les révisions existantes :

```
wp post delete $(wp post list --post_type=revision --format=ids) --force
```

VALEUR PAR DÉFAUT :

Révisions illimitées

1.1.10 *Corbeille Automatique Configurée*

MITRE ATT&CK : T1005

DESCRIPTION :

Les éléments supprimés restent dans la corbeille WordPress indéfiniment par défaut. Configurer une purge automatique évite l'accumulation de contenu supprimé potentiellement sensible.

```
# Vérification wp-config.php
grep "EMPTY_TRASH_DAYS" wp-config.php
# Vérifier contenu corbeille
wp post list --post_status=trash --format=count
```

REMÉDIATION :

1. Configurer la purge automatique :

```
define('EMPTY_TRASH_DAYS', 7);
```

REMÉDIATION :

1. Purger manuellement la corbeille :

```
wp post delete $(wp post list --post_status=trash --format=ids) --force
```

VALEUR PAR DÉFAUT :

30 jours

1.1.11 *Version MySQL/MariaDB Supportée*

MITRE ATT&CK : T1190

DESCRIPTION :

La base de données doit utiliser une version supportée de MySQL (8.0+) ou MariaDB (10.3+). Les versions obsolètes contiennent des vulnérabilités critiques et manquent de fonctionnalités de sécurité modernes.

```
# Via wp-cli
wp db version
# Connexion directe
mysql -V
# Via PHP
wp eval "echo 'MySQL: ' . $wpdb->db_version();"
```

REMÉDIATION :

1. Mettre à niveau vers MySQL 8.0+ ou MariaDB 10.5+
2. Vérifier la compatibilité WordPress après migration
3. Optimiser les paramètres de sécurité MySQL

VALEUR PAR DÉFAUT :

Dépend de l'hébergeur

1.1.12 *Constantes de Sécurité Avancées*

MITRE ATT&CK : T1040

DESCRIPTION :

WordPress offre plusieurs constantes pour renforcer la sécurité : forcer HTTPS, sécuriser les cookies, contrôler les redirections. Ces configurations durcissent la sécurité globale.

```
# Vérifier les constantes de sécurité
grep -E "(FORCE_SSL_ADMIN|COOKIE_DOMAIN|WP_HTTP_BLOCK_EXTERNAL)" wp-config.php
```

REMÉDIATION :

1. Ajouter les constantes de sécurité :

```
define('FORCE_SSL_ADMIN', true);
define('COOKIE_DOMAIN', '.example.com');
define('COOKIEHASH', md5('example.com'));
define('WP_HTTP_BLOCK_EXTERNAL', true);
define('WP_ACCESSIBLE_HOSTS', 'api.wordpress.org,*.github.com');
```

VALEUR PAR DÉFAUT :

Non configurées

1.1.13 *Répertoire wp-content Personnalisé*

MITRE ATT&CK : T1083

DESCRIPTION :

Personnaliser le répertoire wp-content ajoute une couche d'obscurité et complique la reconnaissance automatisée de l'architecture WordPress par les attaquants.

```
# Vérification wp-config.php
grep -E "(WP_CONTENT_DIR|WP_CONTENT_URL)" wp-config.php
# Structure répertoires
ls -la | grep -v wp-content
```

REMÉDIATION :

1. Avant installation, dans wp-config.php :

```
define('WP_CONTENT_DIR', ABSPATH . 'assets/');
define('WP_CONTENT_URL', 'https://example.com/assets/');
```

REMÉDIATION :

1. Déplacer le répertoire wp-content vers assets/
2. Mettre à jour les liens dans la base de données

VALEUR PAR DÉFAUT :

wp-content/

1.1.14 *URL d'Administration Personnalisée*

MITRE ATT&CK : T1083

DESCRIPTION :

Modifier l'URL d'accès à l'administration (/wp-admin/) réduit l'exposition aux attaques par force brute automatisées qui ciblent l'URL standard.

```
# Test accès standard
curl -s -I https://example.com/wp-admin/
curl -s -I https://example.com/wp-login.php
# Vérifier plugin de sécurité installé
wp plugin list | grep -i security
```

REMÉDIATION :

1. Utiliser un plugin comme "WPS Hide Login" :

```
wp plugin install wps-hide-login --activate
wp option update whl_page "admin-secure-$(openssl rand -hex 6)"
```

REMÉDIATION :

1. Ou via .htaccess (méthode avancée) :

```
RewriteRule ^admin-secret/?$ /wp-admin/ [L]
RewriteRule ^wp-admin/$ /404/ [L]
```

VALEUR PAR DÉFAUT :

/wp-admin/ et /wp-login.php

1.1.15 *Suppression Fichiers par Défaut*

MITRE ATT&CK : T1083

DESCRIPTION :

WordPress installe des fichiers par défaut qui révèlent des informations sur la version et la configuration : readme.html, license.txt, wp-config-sample.php. Ces fichiers doivent être supprimés.

```
# Test présence fichiers par défaut
curl -s https://example.com/readme.html
curl -s https://example.com/license.txt
curl -s https://example.com/wp-config-sample.php
ls -la readme.html license.txt wp-config-sample.php
```

REMÉDIATION :

1. Supprimer les fichiers sensibles :

```
rm readme.html license.txt wp-config-sample.php
rm wp-admin/install.php wp-admin/upgrade.php
```

REMÉDIATION :

1. Bloquer via .htaccess :

```
<FilesMatch "(readme|license|changelog|install)">
Order allow,deny
Deny from all
</FilesMatch>
```

VALEUR PAR DÉFAUT :

Fichiers présents après installation

1.1.16 *Configuration Memory Limit Appropriée*

MITRE ATT&CK : T1498

DESCRIPTION :

Une limite mémoire insuffisante peut causer des erreurs et des vulnérabilités. Une limite trop élevée peut permettre des attaques par déni de service. La valeur doit être équilibrée selon les besoins.

```
# Via wp-cli
wp eval "echo 'Memory Limit: ' . ini_get('memory_limit');"
# Via WPScan
wpscan --url https://example.com --enumerate vp | grep -i memory
```

REMÉDIATION :

1. Dans wp-config.php :

```
ini_set('memory_limit', '256M');
define('WP_MEMORY_LIMIT', '256M');
define('WP_MAX_MEMORY_LIMIT', '512M');
```

REMÉDIATION :

1. Surveiller l'utilisation mémoire réelle

VALEUR PAR DÉFAUT :

40M ou 128M selon la configuration

1.1.17 *Configuration Timezone Sécurisée*

MITRE ATT&CK : T1070

DESCRIPTION :

Un fuseau horaire correct est essentiel pour la synchronisation des logs, la planification des tâches et la corrélation d'événements de sécurité. Les timestamps incorrects compliquent l'analyse forensique.

```
# Via wp-cli
wp option get timezone_string
wp eval "echo date_default_timezone_get();"
# Interface admin
# Réglages > Général > Fuseau horaire
```

REMÉDIATION :

1. Configurer le fuseau horaire :

```
wp option update timezone_string "Europe/Paris"
```

REMÉDIATION :

1. Dans wp-config.php :

```
date_default_timezone_set('Europe/Paris');
```

VALEUR PAR DÉFAUT :

UTC ou fuseau du serveur

1.1.18 *Limitation Taille Upload Sécurisée*

MITRE ATT&CK : T1105

DESCRIPTION :

Limiter la taille des fichiers uploadés prévient les attaques par déni de service et réduit les risques d'upload de fichiers malveillants volumineux.

```
# Via wp-cli
wp eval "echo 'Upload Max: ' . ini_get('upload_max_filesize');"
wp eval "echo 'Post Max: ' . ini_get('post_max_size');"
# Interface admin
# Médias > Ajouter
```

REMÉDIATION :

1. Dans wp-config.php :

```
ini_set('upload_max_filesize', '10M');
ini_set('post_max_size', '10M');
ini_set('max_execution_time', 60);
```

REMÉDIATION :

1. Via .htaccess :

```
php_value upload_max_filesize 10M
php_value post_max_size 10M
```

VALEUR PAR DÉFAUT :

2M à 32M selon l'hébergeur

1.1.19 *Configuration Cron Sécurisée*

MITRE ATT&CK : T1053

DESCRIPTION :

Le système cron WordPress (wp-cron) peut être exploité pour des attaques par déni de service. Le désactiver et utiliser le cron système améliore la sécurité et les performances.

```
# Vérifier wp-config.php
grep "DISABLE_WP_CRON" wp-config.php
# Tester wp-cron
curl https://example.com/wp-cron.php
wp cron event list
```

REMÉDIATION :

1. Désactiver wp-cron dans wp-config.php :

```
define('DISABLE_WP_CRON', true);
```

REMÉDIATION :

1. Configurer cron système :

```
# Crontab
*/15 * * * * curl -s https://example.com/wp-cron.php >/dev/null 2>&1
```

VALEUR PAR DÉFAUT :

wp-cron activé

1.1.20 *Configuration Error Reporting Sécurisée*

MITRE ATT&CK : T1005

DESCRIPTION :

Les erreurs PHP affichées peuvent révéler des informations sensibles sur la structure du serveur et les chemins de fichiers. L'affichage doit être désactivé en production.

```
# Forcer une erreur PHP
curl -s "https://example.com/wp-content/themes/active-theme/functions.php"
# Vérifier configuration
wp eval "echo 'Display Errors: ' . ini_get('display_errors');"
```

REMÉDIATION :

1. Dans wp-config.php :

```
ini_set('display_errors', 0);
ini_set('log_errors', 1);
ini_set('error_log', ABSPATH . 'wp-content/debug.log');
```

REMÉDIATION :

1. Protéger le fichier de log :

```
<Files debug.log>
Order allow,deny
Deny from all
</Files>
```

VALEUR PAR DÉFAUT :

Dépend de la configuration PHP

2.0 — AUTHENTIFICATION & GESTION DES UTILISATEURS

2.1.1 *Politique de Mots de Passe Forte*

MITRE ATT&CK : T1110.001

DESCRIPTION :

WordPress ne impose pas de politique de mots de passe par défaut. Des mots de passe faibles facilitent les attaques par force brute et par dictionnaire. Une politique stricte est essentielle pour la sécurité des comptes.

```
# Énumérer les utilisateurs
wp user list --field=user_login
wpscan --url https://example.com --enumerate u --no-banner
# Tester force brute (avec autorisation)
wp user check-password admin password123
```

REMÉDIATION :

1. Installer un plugin de politique de mots de passe :

```
wp plugin install force-strong-passwords --activate
```

REMÉDIATION :

1. Ou ajouter dans functions.php :

```
function enforce_strong_passwords($errors, $update, $user) {
    $password = $_POST['pass1'];
    if (!empty($password) && strlen($password) < 12) {
        $errors->add('weak_password', 'Le mot de passe doit contenir au moins 12 caractères.');
    }
    return $errors;
}
add_action('user_profile_update_errors', 'enforce_strong_passwords', 10, 3);
```

VALEUR PAR DÉFAUT :

Aucune politique appliquée

2.1.2 *Authentification Multi-Facteurs (2FA/MFA)*

MITRE ATT&CK : T1110

DESCRIPTION :

L'authentification à deux facteurs est cruciale pour protéger les comptes administrateurs. Elle ajoute une couche de sécurité même en cas de compromission du mot de passe.

```
# Vérifier plugins 2FA installés
wp plugin list | grep -iE "(two|2fa|authenticator|otp)"
# Tester connexion admin sans 2FA
curl -d "log=admin&pwd=password" https://example.com/wp-login.php
```

REMÉDIATION :

1. Installer un plugin 2FA réputé :

```
wp plugin install two-factor --activate
# ou
wp plugin install google-authenticator --activate
# ou
wp plugin install wordfence --activate
```

REMÉDIATION :

1. Configurer pour tous les administrateurs
2. Forcer l'activation pour les rôles sensibles

VALEUR PAR DÉFAUT :

Non activée

2.1.3 *Limitation Tentatives de Connexion*

MITRE ATT&CK : T1110.001

DESCRIPTION :

WordPress permet un nombre illimité de tentatives de connexion par défaut, facilitant les attaques par force brute. Limiter les tentatives et implémenter un délai progressif est essentiel.

```
# Tester attaque force brute
curl -d "log=admin&pwd=wrong1" https://example.com/wp-login.php
curl -d "log=admin&pwd=wrong2" https://example.com/wp-login.php
curl -d "log=admin&pwd=wrong3" https://example.com/wp-login.php
# Vérifier si compte bloqué
```

REMÉDIATION :

1. Plugin de limitation des tentatives :

```
wp plugin install limit-login-attempts-reloaded --activate
wp option update limit_login_allowed_retries 3
wp option update limit_login_lockout_duration 1800
```

REMÉDIATION :

1. Ou via Wordfence :

```
wp plugin install wordfence --activate
```

VALEUR PAR DÉFAUT :

Tentatives illimitées

2.1.4 *Protection reCAPTCHA*

MITRE ATT&CK : T1110.003

DESCRIPTION :

L'intégration de reCAPTCHA sur les formulaires de connexion, d'inscription et de contact prévient les attaques automatisées et le spam.

```
# Vérifier présence reCAPTCHA sur wp-login
curl -s https://example.com/wp-login.php | grep -i recaptcha
curl -s https://example.com/wp-login.php | grep -i "google.com/recaptcha"
```

REMÉDIATION :

1. Installer un plugin reCAPTCHA :

```
wp plugin install google-captcha --activate
```

REMÉDIATION :

1. Configurer les clés API Google reCAPTCHA
2. Activer sur : connexion, inscription, commentaires, contact

VALEUR PAR DÉFAUT :

Non configuré

2.1.5 *Gestion Roles Utilisateurs Restrictive*

MITRE ATT&CK : T1078

DESCRIPTION :

Appliquer le principe du moindre privilège en assignant les rôles minimaux nécessaires. Éviter la prolifération des comptes administrateurs et auditer régulièrement les permissions.

```
# Lister utilisateurs par rôle
wp user list --role=administrator --format=table
wp user list --role=editor --format=table
wp user list --format=table --fields=user_login,roles
# Énumération externe
wpscan --url https://example.com --enumerate u1-100 --no-banner
```

REMÉDIATION :

1. Auditer et réduire les administrateurs :

```
wp user set-role user2 editor
wp user set-role user3 author
```

REMÉDIATION :

1. Créer des rôles personnalisés si nécessaire :

```
wp plugin install members --activate
```

REMÉDIATION :

1. Supprimer les comptes inutiles :

```
wp user delete unused_user --reassign=1
```

VALEUR PAR DÉFAUT :

Rôle admin souvent sur-utilisé

2.1.6 *Prévention Énumération Utilisateurs*

MITRE ATT&CK : T1087

DESCRIPTION :

WordPress permet l'énumération des utilisateurs via diverses méthodes (?author=1, /wp-json/wp/v2/users/, etc.). Cette information facilite les attaques ciblées et doit être bloquée.

```
# Test énumération classique
curl -s "https://example.com/?author=1" | grep -o "author/[^/]*"
curl -s "https://example.com/wp-json/wp/v2/users/" | jq '.[].slug'
# Via WPScan
wpscan --url https://example.com --enumerate u --no-banner
```

REMÉDIATION :

1. Bloquer via .htaccess :

```
# Bloquer énumération ?author=
RewriteCond %{QUERY_STRING} ^author=([0-9]+) [NC]
RewriteRule ^(.*)$ /? [R=301,L]

# Bloquer API users
RewriteRule ^wp-json/wp/v2/users /404 [R=404,L]
```

REMÉDIATION :

1. Ou via functions.php :

```
// Bloquer énumération utilisateurs
add_action('wp', 'prevent_user_enumeration');
function prevent_user_enumeration() {
    if (is_author() || (isset($_GET['author']) && !is_admin())) {
        wp_redirect(home_url(), 301);
        exit;
    }
}
```

VALEUR PAR DÉFAUT :

Énumération possible

2.1.7 *Désactivation XML-RPC*

MITRE ATT&CK : T1190

DESCRIPTION :

XML-RPC permet l'accès distant à WordPress mais est souvent exploité pour des attaques par amplification DDoS et force brute. Il doit être désactivé si non utilisé.

```
# Test XML-RPC actif
curl -X POST https://example.com/xmlrpc.php \
  -d "<?xml version='1.0'?><methodCall><methodName>system.listMethods</methodName></methodCall>"
# Test attaque amplification
curl -X POST https://example.com/xmlrpc.php \
  -d "<?xml version='1.0'?><methodCall><methodName>system.multicall</methodName><params><param><value><array><data><value><struct><
```

REMÉDIATION :

1. Désactiver via functions.php :

```
add_filter('xmlrpc_enabled', '__return_false');
remove_action('wp_head', 'rsd_link');
```

REMÉDIATION :

1. Bloquer via .htaccess :

```
<Files xmlrpc.php>
Order allow,deny
Deny from all
</Files>
```

REMÉDIATION :

1. Via plugin de sécurité :

```
wp plugin install disable-xml-rpc --activate
```

VALEUR PAR DÉFAUT :

XML-RPC activé

2.1.8 *Sécurisation wp-login.php*

MITRE ATT&CK : T1110

DESCRIPTION :

La page de connexion WordPress est une cible privilégiée des attaques. Implémenter des mesures de protection spécifiques : HTTPS forcé, headers de sécurité, masquage des erreurs.

```
# Tester page de connexion
curl -s -I https://example.com/wp-login.php
curl -s https://example.com/wp-login.php | grep -i "motdepasse\|username\|erreur"
# Test redirection HTTPS
curl -s -I http://example.com/wp-login.php
```

REMÉDIATION :

1. Forcer HTTPS pour l'admin :

```
define('FORCE_SSL_ADMIN', true);
```

REMÉDIATION :

1. Personnaliser les erreurs de connexion :

```
function custom_login_error() {
    return 'Identifiants incorrects.';
}
add_filter('login_errors', 'custom_login_error');
```

REMÉDIATION :

1. Ajouter headers de sécurité :

```
<FilesMatch "wp-login.php">
Header always set X-Frame-Options "DENY"
Header always set X-Content-Type-Options "nosniff"
</FilesMatch>
```

VALEUR PAR DÉFAUT :

Protection basique uniquement

2.1.9 *Audit Sessions Utilisateurs*

MITRE ATT&CK : T1078

DESCRIPTION :

Surveiller et auditer les sessions utilisateurs actives permet de détecter les accès non autorisés et les sessions concurrentes suspectes.

```
# Lister sessions actives
wp user session list --all
# Vérifier metadata utilisateurs
wp user meta list 1 | grep session
```

REMÉDIATION :

1. Installer plugin d'audit des sessions :

```
wp plugin install user-session-control --activate
```

REMÉDIATION :

1. Configurer expiration des sessions :

```
// Limiter durée des sessions
function custom_session_expire($expiration, $user_id, $remember) {
    return $remember ? DAY_IN_SECONDS * 7 : HOUR_IN_SECONDS * 2;
}
add_filter('auth_cookie_expiration', 'custom_session_expire', 10, 3);
```

VALEUR PAR DÉFAUT :

Sessions illimitées dans le temps

2.1.10 *Protection Comptes Inactifs*

MITRE ATT&CK : T1078

DESCRIPTION :

Les comptes utilisateurs inactifs représentent un risque de sécurité. Ils doivent être identifiés, désactivés ou supprimés selon une politique définie.

```
# Utilisateurs sans connexion récente
wp user list --format=csv --fields=user_login,user_registered | head
# Via plugin d'audit
wp plugin list | grep -i "activity\|audit\|log"
```

REMÉDIATION :

1. Auditer l'activité utilisateur :

```
wp plugin install wp-security-audit-log --activate
```

REMÉDIATION :

1. Script de désactivation automatique :

```
function disable_inactive_users() {
    $inactive_days = 90;
    $users = get_users(array('meta_key' => 'last_activity', 'meta_compare' => '<', 'meta_value' => date('Y-m-d', strtotime("-{$inac
    foreach ($users as $user) {
        wp_update_user(array('ID' => $user->ID, 'user_status' => 1));
    }
}
```

VALEUR PAR DÉFAUT :

Comptes inactifs conservés indéfiniment

2.1.11 *Sécurisation Mot de Passe Administrateur*

MITRE ATT&CK : T1110

DESCRIPTION :

Le compte administrateur principal est la cible prioritaire des attaquants. Son mot de passe doit être extrêmement robuste et changé régulièrement.

```
# Identifier le compte admin principal
wp user list --role=administrator --field=user_login | head -1
# Tenter connexion avec mots de passe communs (avec autorisation)
```

REMÉDIATION :

1. Générer mot de passe fort :

```
wp user update admin --user_pass="$(openssl rand -base64 32)"
```

REMÉDIATION :

1. Renommer le compte admin :

```
wp user update admin --user_login=newadminname
```

REMÉDIATION :

1. Implémenter rotation régulière des mots de passe

VALEUR PAR DÉFAUT :

Souvent "admin/admin" ou mots de passe faibles

2.1.12 *Configuration Cookies Sécurisés*

MITRE ATT&CK : T1539

DESCRIPTION :

Les cookies d'authentification doivent être configurés avec les attributs de sécurité appropriés : Secure, HttpOnly, SameSite pour prévenir les attaques de détournement.

```
# Analyser les cookies
curl -c cookies.txt -b cookies.txt -s https://example.com/wp-login.php
cat cookies.txt
# Vérifier attributs sécurité
curl -s -I -D headers.txt https://example.com/wp-login.php
grep -i "set-cookie" headers.txt
```

REMÉDIATION :

1. Configurer cookies sécurisés :

```
// Dans wp-config.php
ini_set('session.cookie_secure', 1);
ini_set('session.cookie_httponly', 1);
ini_set('session.cookie_samesite', 'Strict');
define('COOKIE_DOMAIN', '.example.com');
```

REMÉDIATION :

1. Via .htaccess :

```
<IfModule mod_headers.c>
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure;SameSite=Strict
</IfModule>
```

VALEUR PAR DÉFAUT :

Cookies sans attributs de sécurité

2.1.13 *Restriction Accès par IP*

MITRE ATT&CK : T1078

DESCRIPTION :

Limiter l'accès administratif à des adresses IP spécifiques ajoute une couche de sécurité, particulièrement efficace pour les équipes avec IPs fixes.

```
# Tester accès depuis IP non autorisée
curl -s -I https://example.com/wp-admin/ --interface 8.8.8.8
# Vérifier .htaccess ou configuration serveur
grep -A 10 -B 2 "allow\|deny" .htaccess
```

REMÉDIATION :

1. Protection via .htaccess :

```
<Directory "/wp-admin">
Order deny,allow
Deny from all
Allow from 192.168.1.0/24
Allow from 203.0.113.5
</Directory>
```

REMÉDIATION :

1. Via plugin de sécurité :

```
wp plugin install all-in-one-wp-security-and-firewall --activate
```

VALEUR PAR DÉFAUT :

Accès libre depuis toutes les IPs

2.1.14 *Audit Trail Authentifications*

MITRE ATT&CK : T1078

DESCRIPTION :

Enregistrer toutes les tentatives d'authentification (succès et échecs) permet de détecter les attaques et d'analyser les patterns suspects d'accès.

```
# Vérifier logs d'activité
wp plugin list | grep -iE "(log|audit|activity)"
# Examiner fichiers de log
ls -la wp-content/uploads/wp-security-audit-log/
tail -f wp-content/debug.log | grep -i login
```

REMÉDIATION :

1. Installer plugin d'audit complet :

```
wp plugin install wp-security-audit-log --activate
```

REMÉDIATION :

1. Configurer logs personnalisés :

```
function log_login_attempts($user_login, $user = null) {
    $ip = $_SERVER['REMOTE_ADDR'];
    $user_agent = $_SERVER['HTTP_USER_AGENT'];
    error_log("Login attempt: {$user_login} from {$ip} - {$user_agent}");
}
add_action('wp_login_failed', 'log_login_attempts');
add_action('wp_login', 'log_login_attempts', 10, 2);
```

VALEUR PAR DÉFAUT :

Logs minimaux dans les logs serveur

2.1.15 *Protection Contre Attaques Timing*

MITRE ATT&CK : T1110

DESCRIPTION :

Les différences de temps de réponse entre utilisateur valide/invalide peuvent révéler des informations. Uniformiser les temps de réponse prévient les attaques par analyse temporelle.

```
# Mesurer temps de réponse
time curl -d "log=validuser&pwd=wrongpass" https://example.com/wp-login.php
time curl -d "log=invaliduser&pwd=wrongpass" https://example.com/wp-login.php
# Comparer les différences
```

REMÉDIATION :

1. Uniformiser les temps de réponse :

```
function uniform_login_response() {
    // Délai fixe pour toutes les réponses
    usleep(rand(500000, 1000000)); // 0.5-1 seconde
}
add_action('login_init', 'uniform_login_response');
```

REMÉDIATION :

1. Utiliser des hashes constants :

```
function secure_login_check($user, $password) {
    $stored_hash = '$2y$10$dummy.hash.for.timing.protection';
    password_verify($password, $stored_hash); // Calcul constant
    return $user;
}
```

VALEUR PAR DÉFAUT :

Temps de réponse variables

2.1.16 *Gestion Expiration Mots de Passe*

MITRE ATT&CK : T1110

DESCRIPTION :

Forcer le renouvellement périodique des mots de passe, particulièrement pour les comptes privilégiés, réduit l'impact d'une compromission de credentials.

```
# Vérifier plugins de gestion des mots de passe
wp plugin list | grep -iE "(password|expir)"
# Examiner métadonnées utilisateurs
wp user meta list 1 | grep -iE "(password|expir)"
```

REMÉDIATION :

1. Plugin d'expiration des mots de passe :

```
wp plugin install force-password-reset --activate
```

REMÉDIATION :

1. Code personnalisé :

```
function check_password_age($user_login, $user = null) {
    $last_change = get_user_meta($user->ID, 'password_last_change', true);
    $max_age = 90 * DAY_IN_SECONDS; // 90 jours

    if ($last_change && (time() - $last_change > $max_age)) {
        wp_redirect(admin_url('profile.php?password_expired=1'));
        exit;
    }
}
add_action('wp_login', 'check_password_age', 10, 2);
```

VALEUR PAR DÉFAUT :

Mots de passe sans expiration

2.1.17 *Protection Reset Mot de Passe*

MITRE ATT&CK : T1110

DESCRIPTION :

Le processus de réinitialisation des mots de passe peut être exploité pour des attaques par énumération et déni de service. Il doit être sécurisé et limité.

```
# Tester reset de mot de passe
curl -d "user_login=admin" https://example.com/wp-login.php?action=lostpassword
curl -d "user_login=nonexistent" https://example.com/wp-login.php?action=lostpassword
# Vérifier rate limiting
```

REMÉDIATION :

1. Limiter les demandes de reset :

```
function limit_password_reset() {
    $ip = $_SERVER['REMOTE_ADDR'];
    $transient = 'pwd_reset_' . md5($ip);

    if (get_transient($transient)) {
        wp_die('Trop de demandes. Réessayez dans 15 minutes.');
    }

    set_transient($transient, true, 15 * MINUTE_IN_SECONDS);
}
add_action('lostpassword_post', 'limit_password_reset');
```

REMÉDIATION :

1. Masquer les messages d'erreur :

```
function generic_reset_message() {
    return 'Si cet utilisateur existe, un email a été envoyé.';
}
add_filter('lostpassword_errors', 'generic_reset_message');
```

VALEUR PAR DÉFAUT :

Reset illimité avec messages révélateurs

2.1.18 *Authentification Applications Tierces*

MITRE ATT&CK : T1078

DESCRIPTION :

Les applications tierces utilisant l'API WordPress doivent utiliser des méthodes d'authentification sécurisées : Application Passwords, OAuth, JWT plutôt que les credentials directs.

```
# Vérifier Application Passwords
wp user application-password list admin
# Tester authentification API
curl -u "admin:password" https://example.com/wp-json/wp/v2/users/me
```

REMÉDIATION :

1. Configurer Application Passwords :

```
wp user application-password create admin "Mon App Mobile" --porcelain
```

REMÉDIATION :

1. Désactiver authentification basique pour API :

```
add_filter('rest_authentication_errors', function($result) {
    if (!empty($result)) {
        return $result;
    }

    if (!is_user_logged_in() && !isset($_SERVER['PHP_AUTH_USER'])) {
        return new WP_Error('rest_not_logged_in', 'Authentification requise.', array('status' => 401));
    }

    return $result;
});
```

VALEUR PAR DÉFAUT :

Authentification basique possible

2.1.19 *Séparation Comptes Fonctionnels*

MITRE ATT&CK : T1078

DESCRIPTION :

Séparer les comptes techniques/fonctionnels des comptes utilisateurs humains améliore la traçabilité et permet un contrôle d'accès granulaire.

```
# Identifier les comptes de service
wp user list --format=table --fields=user_login,user_email,roles
# Vérifier les dernières connexions
wp plugin install wp-security-audit-log --activate
```

REMÉDIATION :

1. Créer comptes dédiés :

```
wp user create api-service api@example.com --role=editor --user_pass="$(openssl rand -base64 32)"
wp user create backup-service backup@example.com --role=subscriber --user_pass="$(openssl rand -base64 32)"
```

REMÉDIATION :

1. Restreindre les permissions :

```
// Rôle personnalisé pour services
function create_service_role() {
    add_role('api_service', 'Service API', array(
        'read' => true,
        'edit_posts' => true,
        'publish_posts' => false
    ));
}
add_action('init', 'create_service_role');
```

VALEUR PAR DÉFAUT :

Comptes mixtes humains/services

2.1.20 *Monitoring Escalade Privilèges*

MITRE ATT&CK : T1078

DESCRIPTION :

Surveiller les changements de rôles et permissions utilisateurs permet de détecter les escalades de privilèges malveillantes ou accidentelles.

```
# Vérifier historique des changements de rôles
wp plugin list | grep -iE "(audit|security|log)"
# Examiner les logs d'activité
```

REMÉDIATION :

1. Monitoring automatisé :

```
function monitor_role_changes($user_id, $role, $old_roles) {
    $user = get_userdata($user_id);
    $new_roles = $user->roles;

    if ($old_roles != $new_roles) {
        error_log("SECURITY: Role change for user {$user->user_login}: " .
                  implode(',', $old_roles) . " -> " . implode(',', $new_roles));

        // Alerter l'administrateur
        wp_mail('admin@example.com', 'Changement de rôle détecté',
                "L'utilisateur {$user->user_login} a changé de rôle.");
    }
}
add_action('set_user_role', 'monitor_role_changes', 10, 3);
```

REMÉDIATION :

1. Plugin d'audit complet :

```
wp plugin install wp-security-audit-log --activate
```

VALEUR PAR DÉFAUT :

Changements non surveillés

2.1.21 *Protection Contre User Enumeration API*

MITRE ATT&CK : T1087

DESCRIPTION :

L'API REST WordPress expose par défaut les informations utilisateurs. Cette exposition doit être contrôlée pour empêcher la reconnaissance et le profilage des utilisateurs.

```
# Test exposition utilisateurs via API
curl -s https://example.com/wp-json/wp/v2/users/ | jq '.[] | {id, name, slug}'
curl -s https://example.com/wp-json/wp/v2/users/1 | jq '{id, name, slug, roles}'
```

REMÉDIATION :

1. Restreindre l'accès à l'API utilisateurs :

```
// Bloquer complètement l'endpoint users
add_filter('rest_endpoints', function($endpoints) {
    if (isset($endpoints['/wp/v2/users'])) {
        unset($endpoints['/wp/v2/users']);
    }
    if (isset($endpoints['/wp/v2/users/(?P<id>[\d]+)'])) {
        unset($endpoints['/wp/v2/users/(?P<id>[\d]+)']);
    }
    return $endpoints;
});
```

REMÉDIATION :

1. Ou limiter aux utilisateurs connectés :

```
add_filter('rest_user_query', function($prepared_args, $request) {
    if (!is_user_logged_in()) {
        return new WP_Error('rest_user_cannot_view', 'Accès non autorisé.', array('status' => 401));
    }
    return $prepared_args;
}, 10, 2);
```

VALEUR PAR DÉFAUT :

API utilisateurs publique

2.1.22 *Gestion Sessions Concurrentes*

MITRE ATT&CK : T1078

DESCRIPTION :

Limiter le nombre de sessions simultanées par utilisateur et détecter les connexions concurrentes suspectes peut révéler des compromissions de comptes.

```
# Examiner les sessions actives
wp user session list --all --format=table
# Compter sessions par utilisateur
wp user list --format=ids | xargs -I {} wp user session list {} --format=count
```

REMÉDIATION :

1. Limiter les sessions concurrentes :

```
function limit_concurrent_sessions($user_login, $user) {
    $sessions = WP_Session_Tokens::get_instance($user->ID);
    $all_sessions = $sessions->get_all();

    if (count($all_sessions) > 2) { // Max 2 sessions
        // Détruire les sessions anciennes
        $sessions->destroy_others($_COOKIE[LOGGED_IN_COOKIE]);
    }
}
add_action('wp_login', 'limit_concurrent_sessions', 10, 2);
```

REMÉDIATION :

1. Alertes de connexions multiples :

```
function alert_concurrent_login($user_login, $user) {
    $sessions = WP_Session_Tokens::get_instance($user->ID);
    if (count($sessions->get_all()) > 1) {
        wp_mail($user->user_email, 'Nouvelle connexion détectée',
                'Une nouvelle connexion a été détectée sur votre compte.');
    }
}
add_action('wp_login', 'alert_concurrent_login', 10, 2);
```

VALEUR PAR DÉFAUT :

Sessions multiples illimitées

2.1.23 *Protection Mot de Passe en Transit*

MITRE ATT&CK : T1040

DESCRIPTION :

Tous les échanges de mots de passe doivent être chiffrés via HTTPS. HTTP doit être complètement désactivé pour les pages d'authentification.

```
# Tester redirection HTTPS
curl -I http://example.com/wp-login.php
curl -I http://example.com/wp-admin/
# Vérifier force SSL
grep "FORCE_SSL_ADMIN" wp-config.php
```

REMÉDIATION :

1. Forcer HTTPS pour l'administration :

```
// wp-config.php
define('FORCE_SSL_ADMIN', true);
```

REMÉDIATION :

1. Redirection .htaccess :

```
RewriteCond %{HTTPS} off
RewriteRule ^(wp-admin|wp-login\.php).*$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

REMÉDIATION :

1. Headers de sécurité HTTPS :

```
<IfModule mod_headers.c>
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
</IfModule>
```

VALEUR PAR DÉFAUT :

HTTP autorisé

2.1.24 *Authentification Forte pour API*

MITRE ATT&CK : T1078

DESCRIPTION :

L'API REST WordPress doit utiliser des méthodes d'authentification robustes : JWT, OAuth2, ou Application Passwords plutôt que l'authentification basique.

```
# Tester authentification API
curl -u "admin:password" https://example.com/wp-json/wp/v2/posts
# Vérifier plugins d'authentification
wp plugin list | grep -iE "(jwt|oauth|auth)"
```

REMÉDIATION :

1. Installer JWT Authentication :

```
wp plugin install jwt-authentication-for-wp-rest-api --activate
```

REMÉDIATION :

1. Configuration JWT :

```
// wp-config.php
define('JWT_AUTH_SECRET_KEY', 'your-secret-key');
define('JWT_AUTH_CORS_ENABLE', true);
```

REMÉDIATION :

1. Désactiver auth basique :

```
add_filter('rest_authentication_errors', function($result) {
    if (isset($_SERVER['PHP_AUTH_USER'])) {
        return new WP_Error('rest_authentication_error', 'Authentification basique désactivée.');
    }
    return $result;
});
```

VALEUR PAR DÉFAUT :

Authentification basique autorisée

2.1.25 *Politique Verrouillage Compte*

MITRE ATT&CK : T1110

DESCRIPTION :

Définir une politique claire de verrouillage des comptes après échecs d'authentification, avec procédures de déverrouillage sécurisées.

```
# Tester politique de verrouillage
# (Nécessite plusieurs tentatives d'échec)
for i in {1..5}; do
  curl -d "log=admin&pwd=wrong$i" https://example.com/wp-login.php
done
# Vérifier état du verrouillage
```

REMÉDIATION :

1. Configuration Wordfence :

```
wp plugin install wordfence --activate
wp option update wordfence_ls_maxFailures 5
wp option update wordfence_ls_duration 1800
```

REMÉDIATION :

1. Politique personnalisée :

```
function advanced_account_lockout($username) {
    $lockout_key = 'failed_login_' . md5($username);
    $failed_count = get_transient($lockout_key) ?: 0;

    if ($failed_count >= 3) {
        // Verrouillage progressif
        $lockout_duration = min(pow(2, $failed_count - 3) * 300, 86400); // Max 24h
        set_transient($lockout_key . '_locked', true, $lockout_duration);

        wp_die("Compte temporairement verrouillé. Réessayez dans " .
               human_time_diff(time() + $lockout_duration) . ".");
    }
}
add_action('wp_login_failed', 'advanced_account_lockout');
```

VALEUR PAR DÉFAUT :

Pas de verrouillage automatique

3.0 — PLUGINS & EXTENSIONS

3.1.1 *Inventaire Complet des Plugins*

MITRE ATT&CK : T1190

DESCRIPTION :

Maintenir un inventaire complet et à jour de tous les plugins installés est essentiel pour la gestion des vulnérabilités et la réduction de la surface d'attaque.

```
# Lister tous les plugins
wp plugin list --format=table --fields=name,status,version,update_version
# Scanner avec WPScan
wpscan --url https://example.com --enumerate vp --plugins-detection aggressive --no-banner
# Plugins inactifs
wp plugin list --status=inactive
```

REMÉDIATION :

1. Documenter tous les plugins :

```
wp plugin list --format=csv --fields=name,version,status,auto_update > plugins_inventory.csv
```

REMÉDIATION :

1. Supprimer plugins inactifs :

```
wp plugin delete $(wp plugin list --status=inactive --field=name)
```

REMÉDIATION :

1. Audit régulier :

```
# Script d'audit mensuel
#!/bin/bash
wp plugin list --field=name | while read plugin; do
  echo "Plugin: $plugin - $(wp plugin get $plugin --field=version)"
done
```

VALEUR PAR DÉFAUT :

Plugins accumulés sans inventaire

3.1.2 *Plugins Obsolètes et Abandonnés*

MITRE ATT&CK : T1190

DESCRIPTION :

Les plugins obsolètes ou abandonnés par leurs développeurs représentent un risque critique car ils ne reçoivent plus de correctifs de sécurité.

```
# Vérifier mises à jour disponibles
wp plugin list --update=available
# Scanner plugins obsolètes avec WPScan
wpscan --url https://example.com --enumerate vp --plugins-version-detection aggressive
# Vérifier dernière mise à jour
wp plugin list --format=json | jq '.[] | {name: .name, version: .version}'
```

REMÉDIATION :

1. Identifier plugins sans mise à jour récente :

```
# Plugins sans update depuis 2 ans
wp plugin list --format=json | jq '.[] | select(.last_updated < "2022-01-01")'
```

REMÉDIATION :

1. Remplacer ou supprimer :

```
wp plugin deactivate abandoned-plugin
wp plugin delete abandoned-plugin
# Chercher alternative
wp plugin search "alternative functionality"
```

REMÉDIATION :

1. Surveillance automatisée :

```
function check_abandoned_plugins() {
    $plugins = get_plugins();
    foreach ($plugins as $plugin_file => $plugin_data) {
        // Vérifier âge dernière mise à jour
        $last_updated = get_plugin_data($plugin_file)['Version'];
        // Alerter si > 2 ans
    }
}
wp_schedule_event(time(), 'monthly', 'check_abandoned_plugins');
```

VALEUR PAR DÉFAUT :

Plugins conservés indéfiniment

3.1.3 *Vulnérabilités Connues (Base WPScan)*

MITRE ATT&CK : T1190

DESCRIPTION :

Vérifier régulièrement les plugins contre la base de vulnérabilités WPScan pour identifier les risques de sécurité connus et non patchés.

```
# Scan vulnérabilités avec API key
wpscan --url https://example.com --enumerate vp --plugins-detection aggressive --api-token YOUR_API_TOKEN
# Vérification spécifique
wpscan --url https://example.com --enumerate vp --plugins-detection aggressive | grep -A 5 -B 5 "vulnerabilities"
```

REMÉDIATION :

1. Audit automatisé quotidien :

```
#!/bin/bash
wpscan --url https://example.com --enumerate vp --api-token $WPSCAN_TOKEN --format json > daily_scan.json
if grep -q "vulnerabilities" daily_scan.json; then
  mail -s "Vulnérabilités détectées" admin@example.com < daily_scan.json
fi
```

REMÉDIATION :

1. Mise à jour immédiate des plugins vulnérables :

```
wp plugin update --all
```

REMÉDIATION :

1. Surveillance continue :

```
// Hook pour vérifier avant activation
function check_plugin_vulnerabilities($plugin) {
    // Intégration API WPScan
    $response = wp_remote_get("https://wpscan.com/api/v3/plugins/{$plugin}");
    // Analyser réponse et bloquer si vulnérable
}
add_action('activate_plugin', 'check_plugin_vulnerabilities');
```

VALEUR PAR DÉFAUT :

Aucune vérification automatique

3.1.4 *Permissions et Capacités des Plugins*

MITRE ATT&CK : T1078

DESCRIPTION :

Auditer les permissions demandées par les plugins et s'assurer qu'elles respectent le principe du moindre privilège. Certains plugins demandent des droits excessifs.

```
# Examiner les capacités des plugins
wp eval "
\$plugins = get_plugins();
foreach (\$plugins as \$plugin => \$data) {
    echo \$plugin . ' - ' . \$data['Name'] . PHP_EOL;
}
"
# Vérifier hooks et actions
grep -r "add_action\|add_filter" wp-content/plugins/ | head -20
```

REMÉDIATION :

1. Audit des permissions :

```
function audit_plugin_capabilities() {
    $active_plugins = get_option('active_plugins');
    foreach ($active_plugins as $plugin) {
        $plugin_data = get_plugin_data(WP_PLUGIN_DIR . '/' . $plugin);
        // Analyser les hooks et capabilities
        echo "Plugin: " . $plugin_data['Name'] . " - Version: " . $plugin_data['Version'] . "\n";
    }
}
```

REMÉDIATION :

1. Restriction des plugins par rôle :

```
function restrict_plugin_access() {
    if (!current_user_can('manage_options')) {
        remove_menu_page('plugins.php');
    }
}
add_action('admin_menu', 'restrict_plugin_access');
```

VALEUR PAR DÉFAUT :

Permissions larges non auditées

3.1.5 *Plugins Nulled et Piratés*

MITRE ATT&CK : T1505

DESCRIPTION :

Les plugins piratés ou "nulled" contiennent souvent des backdoors, malwares ou code malveillant. Ils représentent un risque critique pour la sécurité.

```
# Scanner fichiers suspects
find wp-content/plugins/ -name "*.php" -exec grep -l "eval\|base64_decode\|gzinflate" {} \;
# Vérifier intégrité avec checksums officiels
wp plugin verify-checksums --all
# Rechercher backdoors connues
grep -r "c99\|r57\|webshell" wp-content/plugins/
```

REMÉDIATION :

1. Supprimer immédiatement les plugins suspects :

```
wp plugin deactivate suspicious-plugin
wp plugin delete suspicious-plugin
```

REMÉDIATION :

1. Scanner régulièrement :

```
# Script de détection malware
#!/bin/bash
find wp-content/ -name "*.php" | xargs grep -l "eval.*base64_decode\|system.*\$_" > suspicious_files.txt
if [ -s suspicious_files.txt ]; then
  mail -s "Fichiers suspects détectés" admin@example.com < suspicious_files.txt
fi
```

REMÉDIATION :

1. Politique d'acquisition stricte :

```
// Bloquer installation depuis sources non officielles
function restrict_plugin_sources($result, $action, $args) {
    if ($action == 'install-plugin') {
        // Vérifier source officielle WordPress.org
        if (!str_contains($args['source'], 'downloads.wordpress.org')) {
            return new WP_Error('unauthorized_source', 'Installation uniquement depuis WordPress.org autorisée.');
        }
    }
    return $result;
}
add_filter('upgrader_pre_download', 'restrict_plugin_sources', 10, 3);
```

VALEUR PAR DÉFAUT :

Aucun contrôle des sources

3.1.6 *Auto-Update des Plugins Critiques*

MITRE ATT&CK : T1190

DESCRIPTION :

Activer les mises à jour automatiques pour les plugins de sécurité critiques assure une protection rapide contre les vulnérabilités nouvellement découvertes.

```
# Vérifier statut auto-update
wp plugin auto-updates status
wp plugin list --format=table --fields=name,auto_update
# Plugins de sécurité installés
wp plugin list | grep -iE "(security|firewall|backup|malware)"
```

REMÉDIATION :

1. Activer auto-update pour plugins critiques :

```
wp plugin auto-updates enable wordfence
wp plugin auto-updates enable updraftplus
wp plugin auto-updates enable sucuri-scanner
```

REMÉDIATION :

1. Configuration globale :

```
// wp-config.php - Auto-update tous les plugins
define('WP_AUTO_UPDATE_CORE', true);
add_filter('auto_update_plugin', '__return_true');
```

REMÉDIATION :

1. Auto-update sélectif :

```
function selective_plugin_auto_update($update, $item) {
    $security_plugins = array(
        'wordfence/wordfence.php',
        'updraftplus/updraftplus.php',
        'sucuri-scanner/sucuri.php'
    );
    return in_array($item->plugin, $security_plugins);
}
add_filter('auto_update_plugin', 'selective_plugin_auto_update', 10, 2);
```

VALEUR PAR DÉFAUT :

Auto-update désactivé

3.1.7 *Réduction Surface d'Attaque*

MITRE ATT&CK : T1190

DESCRIPTION :

Minimiser le nombre de plugins actifs réduit la surface d'attaque. Chaque plugin supplémentaire augmente les risques de vulnérabilités et de conflits.

```
# Compter plugins actifs
wp plugin list --status=active --format=count
# Analyser utilisation réelle
wp plugin list --format=table --fields=name,status,version
# Identifier plugins redondants
```

REMÉDIATION :

1. Audit d'utilisation :

```
# Désactiver temporairement et tester
wp plugin deactivate non-essential-plugin
# Tester fonctionnalité site
# Si OK, supprimer définitivement
wp plugin delete non-essential-plugin
```

REMÉDIATION :

1. Consolidation des fonctionnalités :

```
// Remplacer multiple plugins par un seul multifonction
// Ex: Wordfence remplace plusieurs plugins sécurité
wp plugin install wordfence --activate
wp plugin deactivate old-security-plugin1 old-security-plugin2
```

REMÉDIATION :

1. Politique de plugins minimale :

```
// Limite nombre de plugins actifs
function enforce_plugin_limit() {
    $active_count = count(get_option('active_plugins', array()));
    if ($active_count > 20) { // Limite à 20 plugins
        wp_die('Limite de plugins atteinte. Désactivez des plugins avant d\'en activer de nouveaux.');
    }
}
add_action('activate_plugin', 'enforce_plugin_limit');
```

VALEUR PAR DÉFAUT :

Accumulation illimitée de plugins

3.1.8 *Validation Code Source Plugins*

MITRE ATT&CK : T1505

DESCRIPTION :

Valider l'intégrité et la sécurité du code source des plugins avant installation, particulièrement pour les plugins provenant de sources tierces.

```
# Vérifier checksums plugins officiels
wp plugin verify-checksums --all
# Scanner code malveillant
find wp-content/plugins/ -name "*.php" -exec php -l {} \; | grep -v "No syntax errors"
# Rechercher patterns suspects
grep -r "eval\|exec\|system\|shell_exec" wp-content/plugins/
```

REMÉDIATION :

1. Validation avant installation :

```
# Télécharger et analyser avant installation
wget https://downloads.wordpress.org/plugin/plugin-name.zip
unzip plugin-name.zip
grep -r "eval\|base64_decode" plugin-name/
# Si clean, installer
wp plugin install plugin-name.zip
```

REMÉDIATION :

1. Monitoring intégrité :

```
function monitor_plugin_integrity() {
    $plugins_dir = WP_PLUGIN_DIR;
    $known_hashes = get_option('plugin_hashes', array());

    foreach (glob("$plugins_dir/*/*.php") as $file) {
        $current_hash = md5_file($file);
        $relative_path = str_replace($plugins_dir . '/', '', $file);

        if (isset($known_hashes[$relative_path]) &&
            $known_hashes[$relative_path] !== $current_hash) {
            // Fichier modifié - alerter
            wp_mail('admin@example.com', 'Plugin modifié', "Le fichier $file a été modifié.");
        }

        $known_hashes[$relative_path] = $current_hash;
    }

    update_option('plugin_hashes', $known_hashes);
}
wp_schedule_event(time(), 'daily', 'monitor_plugin_integrity');
```

VALEUR PAR DÉFAUT :

Aucune validation du code

3.1.9 *Gestion Dépendances des Plugins*

MITRE ATT&CK : T1195

DESCRIPTION :

Documenter et gérer les dépendances entre plugins pour éviter les conflits de sécurité et assurer la cohérence des mises à jour.

```
# Analyser dépendances
wp plugin list --format=json | jq '.[] | {name: .name, requires: .requires_wp}'
# Tester désactivation en cascade
wp plugin deactivate parent-plugin
# Vérifier erreurs
tail -f wp-content/debug.log
```

REMÉDIATION :

1. Cartographie des dépendances :

```
function map_plugin_dependencies() {
    $plugins = get_plugins();
    $dependencies = array();

    foreach ($plugins as $plugin_file => $plugin_data) {
        // Analyser headers et requirements
        if (isset($plugin_data['RequiresPlugins'])) {
            $dependencies[$plugin_file] = explode(',', $plugin_data['RequiresPlugins']);
        }
    }

    update_option('plugin_dependencies', $dependencies);
    return $dependencies;
}
```

REMÉDIATION :

1. Validation ordre de désactivation :

```
function validate_plugin_deactivation($plugin) {
    $dependencies = get_option('plugin_dependencies', array());

    foreach ($dependencies as $dependent_plugin => $required_plugins) {
        if (in_array($plugin, $required_plugins) && is_plugin_active($dependent_plugin)) {
            wp_die("Impossible de désactiver $plugin : requis par $dependent_plugin");
        }
    }
}
add_action('deactivate_plugin', 'validate_plugin_deactivation');
```

VALEUR PAR DÉFAUT :

Gestion manuelle des dépendances

3.1.10 *Sandbox et Tests Plugins*

MITRE ATT&CK : T1505

DESCRIPTION :

Tester les nouveaux plugins dans un environnement de staging isolé avant déploiement en production pour identifier les risques de sécurité.

```
# Vérifier environnement de staging
wp option get siteurl
wp config get WP_ENVIRONMENT_TYPE
# Comparer plugins prod vs staging
wp plugin list --format=csv > production_plugins.csv
```

REMÉDIATION :

1. Configuration environnement de test :

```
// wp-config.php staging
define('WP_ENVIRONMENT_TYPE', 'staging');
define('WP_DEBUG', true);
define('WP_DEBUG_LOG', true);
```

REMÉDIATION :

1. Procédure de test :

```
# Script de test automatisé
#!/bin/bash
STAGING_URL="https://staging.example.com"
PLUGIN_NAME=$1

# Installer sur staging
wp plugin install $PLUGIN_NAME --activate --url=$STAGING_URL
# Scanner vulnérabilités
wpscan --url $STAGING_URL --enumerate vp
# Test fonctionnel
curl -s $STAGING_URL | grep -i "error\|warning"
# Si OK, déployer en prod
```

REMÉDIATION :

1. Isolation des tests :

```
function restrict_staging_plugins() {
    if (wp_get_environment_type() === 'staging') {
        // Désactiver plugins dangereux en staging
        $dangerous_plugins = array('backup-plugin', 'mail-sender');
        foreach ($dangerous_plugins as $plugin) {
            if (is_plugin_active($plugin)) {
                deactivate_plugins($plugin);
            }
        }
    }
}
add_action('plugins_loaded', 'restrict_staging_plugins');
```

VALEUR PAR DÉFAUT :

Installation directe en production

3.1.11 *Monitoring Activité Plugins*

MITRE ATT&CK : T1505

DESCRIPTION :

Surveiller l'activité des plugins (installation, activation, désactivation, mises à jour) pour détecter les changements non autorisés.

```
# Vérifier logs d'activité
wp plugin list | grep -iE "(activity|audit|log)"
# Examiner logs plugin changes
tail -f wp-content/uploads/wp-security-audit-log/
```

REMÉDIATION :

1. Logging complet des activités :

```
function log_plugin_activities($plugin, $network_activation) {
    $user = wp_get_current_user();
    $log_entry = array(
        'timestamp' => current_time('mysql'),
        'user' => $user->user_login,
        'action' => 'plugin_activation',
        'plugin' => $plugin,
        'ip' => $_SERVER['REMOTE_ADDR']
    );

    error_log('PLUGIN_ACTIVITY: ' . json_encode($log_entry));

    // Alerter pour plugins critiques
    if (strpos($plugin, 'security') !== false) {
        wp_mail('admin@example.com', 'Plugin sécurité activé',
                "Plugin $plugin activé par {$user->user_login}");
    }
}
add_action('activated_plugin', 'log_plugin_activities', 10, 2);
add_action('deactivated_plugin', 'log_plugin_activities', 10, 2);
```

REMÉDIATION :

1. Dashboard monitoring :

```
function plugin_activity_dashboard() {
    $recent_activities = get_option('recent_plugin_activities', array());
    echo '<div class="notice notice-info">';
    echo '<h3>Activité Plugins Récente</h3>';
    foreach ($recent_activities as $activity) {
        echo "<p>{$activity['timestamp']} - {$activity['action']} - {$activity['plugin']}</p>";
    }
    echo '</div>';
}
add_action('admin_notices', 'plugin_activity_dashboard');
```

VALEUR PAR DÉFAUT :

Activité non surveillée

3.1.12 *Politique Installation Plugins*

MITRE ATT&CK : T1505

DESCRIPTION :

Établir une politique stricte d'installation et de gestion des plugins avec processus d'approbation et validation sécurisée.

```
# Vérifier qui peut installer des plugins
wp user list --role=administrator --format=table
# Vérifier restrictions actuelles
grep "DISALLOW_FILE_MODS" wp-config.php
```

REMÉDIATION :

1. Restriction installation :

```
// wp-config.php - Désactiver installation plugins
define('DISALLOW_FILE_MODS', true);
```

REMÉDIATION :

1. Processus d'approbation :

```
function require_plugin_approval($result, $action, $args) {
    if ($action === 'install-plugin') {
        $approved_plugins = get_option('approved_plugins', array());
        $plugin_slug = $args['slug'];

        if (!in_array($plugin_slug, $approved_plugins)) {
            return new WP_Error('unapproved_plugin',
                'Plugin non approuvé. Contactez l\'administrateur.');
        }
    }
    return $result;
}
add_filter('upgrader_pre_download', 'require_plugin_approval', 10, 3);
```

REMÉDIATION :

1. Whitelist plugins autorisés :

```
function enforce_plugin_whitelist() {
    $allowed_plugins = array(
        'wordfence/wordfence.php',
        'updraftplus/updraftplus.php',
        'yoast-seo/wp-seo.php'
    );

    $active_plugins = get_option('active_plugins');
    foreach ($active_plugins as $plugin) {
        if (!in_array($plugin, $allowed_plugins)) {
            deactivate_plugins($plugin);
            wp_mail('admin@example.com', 'Plugin non autorisé désactivé',
                    "Plugin $plugin automatiquement désactivé.");
        }
    }
}
add_action('admin_init', 'enforce_plugin_whitelist');
```

VALEUR PAR DÉFAUT :

Installation libre

3.1.13 *Backup avant Modifications Plugins*

MITRE ATT&CK : T1485

DESCRIPTION :

Effectuer systématiquement des sauvegardes avant l'installation, mise à jour ou modification de plugins pour permettre une restauration rapide.

```
# Vérifier politique backup
wp plugin list | grep -iE "(backup|updraft)"
# Dernière sauvegarde
wp option get updraftplus_last_backup
```

REMÉDIATION :

1. Backup automatique avant modifications :

```
function backup_before_plugin_changes($upgrader, $hook_extra) {
    if (isset($hook_extra['type']) && $hook_extra['type'] === 'plugin') {
        // Déclencher sauvegarde
        do_action('updraftplus_backup_now', 'plugins_only');

        // Attendre completion
        sleep(5);

        // Vérifier succès backup
        $last_backup = get_option('updraftplus_last_backup');
        if (!$last_backup || (time() - $last_backup) > 300) {
            wp_die('Sauvegarde échouée. Modification annulée.');
        }
    }
}
add_action('upgrader_process_complete', 'backup_before_plugin_changes', 10, 2);
```

REMÉDIATION :

1. Point de restauration :

```
# Script de point de restauration
#!/bin/bash
BACKUP_DIR="/backups/$(date +%Y%m%d_%H%M%S)"
mkdir -p $BACKUP_DIR
tar -czf $BACKUP_DIR/plugins.tar.gz wp-content/plugins/
mysqldump -u$DB_USER -p$DB_PASS $DB_NAME > $BACKUP_DIR/database.sql
echo "Backup créé: $BACKUP_DIR"
```

VALEUR PAR DÉFAUT :

Modifications sans sauvegarde

3.1.14 *Contrôle Versions Plugins*

MITRE ATT&CK : T1195

DESCRIPTION :

Maintenir un contrôle strict des versions des plugins utilisées, éviter les versions beta/alpha en production, documenter les versions approuvées.

```
# Versions actuelles
wp plugin list --format=table --fields=name,version,status
# Vérifier versions beta/dev
wp plugin list --format=json | jq '.[] | select(.version | contains("beta") or contains("dev") or contains("alpha"))'
```

REMÉDIATION :

1. Politique versions stables uniquement :

```
function enforce_stable_versions($result, $action, $args) {
    if ($action === 'install-plugin') {
        $version = $args['version'] ?? 'latest';
        if (preg_match('/(alpha|beta|dev|rc)/i', $version)) {
            return new WP_Error('unstable_version',
                'Versions non-stables interdites en production.');
        }
    }
    return $result;
}
add_filter('upgrader_pre_download', 'enforce_stable_versions', 10, 3);
```

REMÉDIATION :

1. Verrouillage versions critiques :

```
function lock_critical_plugin_versions($transient) {
    $locked_versions = array(
        'wordfence/wordfence.php' => '7.8.2',
        'updraftplus/updraftplus.php' => '1.22.23'
    );

    if (isset($transient->response)) {
        foreach ($locked_versions as $plugin => $locked_version) {
            if (isset($transient->response[$plugin])) {
                unset($transient->response[$plugin]);
            }
        }
    }

    return $transient;
}
add_filter('pre_set_site_transient_update_plugins', 'lock_critical_plugin_versions');
```

VALEUR PAR DÉFAUT :

Mises à jour automatiques vers dernière version

3.1.15 *Séparation Plugins par Environnement*

MITRE ATT&CK : T1505

DESCRIPTION :

Maintenir des configurations de plugins différentes selon les environnements (dev/staging/prod) pour éviter l'exposition de fonctionnalités de debug en production.

```
# Identifier environnement
wp config get WP_ENVIRONMENT_TYPE
# Plugins de debug actifs
wp plugin list | grep -iE "(debug|dev|test|query|profil)"
```

REMÉDIATION :

1. Configuration par environnement :

```
function environment_specific_plugins() {
    $environment = wp_get_environment_type();

    switch ($environment) {
        case 'production':
            $forbidden_plugins = array(
                'query-monitor/query-monitor.php',
                'debug-bar/debug-bar.php',
                'developer/developer.php'
            );

            foreach ($forbidden_plugins as $plugin) {
                if (is_plugin_active($plugin)) {
                    deactivate_plugins($plugin);
                }
            }
            break;

        case 'staging':
            // Activer plugins de test
            if (!is_plugin_active('query-monitor/query-monitor.php')) {
                activate_plugin('query-monitor/query-monitor.php');
            }
            break;
    }
}
add_action('plugins_loaded', 'environment_specific_plugins');
```

REMÉDIATION :

1. Must-use plugins par environnement :

```
// mu-plugins/environment-control.php
if (wp_get_environment_type() === 'production') {
    // Désactiver fonctionnalités dangereuses
    define('WP_DEBUG', false);
    define('SCRIPT_DEBUG', false);
} else {
    // Environnement de développement
    define('WP_DEBUG', true);
    define('SCRIPT_DEBUG', true);
}
```

VALEUR PAR DÉFAUT :

Configuration identique tous environnements

3.1.16 *Audit Permissions Système Plugins*

MITRE ATT&CK : T1078

DESCRIPTION :

Auditer les permissions système requises par les plugins (lecture/écriture fichiers, accès base de données, appels réseau) pour identifier les sur-privilèges.

```
# Analyser capabilities des plugins
grep -r "current_user_can\|capability" wp-content/plugins/ | head -20
# Vérifier accès fichier système
find wp-content/plugins/ -name "*.php" -exec grep -l "file_get_contents\|fopen\|curl" {} \;
```

REMÉDIATION :

1. Audit des capabilities :

```
function audit_plugin_capabilities() {
    $plugins = get_plugins();
    $capability_report = array();

    foreach ($plugins as $plugin_file => $plugin_data) {
        if (is_plugin_active($plugin_file)) {
            // Analyser le code pour les capabilities utilisées
            $plugin_path = WP_PLUGIN_DIR . '/' . $plugin_file;
            $content = file_get_contents($plugin_path);

            preg_match_all('/current_user_can\s*\(\s*[\'"]([^\'"]+)[\'"]/', $content, $matches);
            $capability_report[$plugin_data['Name']] = array_unique($matches[1]);
        }
    }

    update_option('plugin_capability_audit', $capability_report);
    return $capability_report;
}
```

REMÉDIATION :

1. Restriction capabilities par plugin :

```
function restrict_plugin_capabilities($caps, $cap, $user_id, $args) {
    $restricted_plugins = array(
        'non-critical-plugin' => array('manage_options', 'edit_users')
    );

    $current_plugin = $this->get_current_plugin();
    if (isset($restricted_plugins[$current_plugin]) &&
        in_array($cap, $restricted_plugins[$current_plugin])) {
        return array('do_not_allow');
    }

    return $caps;
}
add_filter('user_has_cap', 'restrict_plugin_capabilities', 10, 4);
```

VALEUR PAR DÉFAUT :

Permissions non auditées

3.1.17 *Protection Fichiers Configuration Plugins*

MITRE ATT&CK : T1005

DESCRIPTION :

Protéger les fichiers de configuration des plugins contre l'accès web direct, particulièrement ceux contenant des clés API ou credentials.

```
# Rechercher fichiers config exposés
find wp-content/plugins/ -name "*.ini" -o -name "*.conf" -o -name "config.php"
curl -s https://example.com/wp-content/plugins/plugin-name/config.ini
# Vérifier clés API dans fichiers
grep -r "api.*key\|secret.*key" wp-content/plugins/ | grep -v ".git"
```

REMÉDIATION :

1. Protection via .htaccess :

```
# wp-content/plugins/.htaccess
<FilesMatch "\.(ini|conf|config)$">
Order allow,deny
Deny from all
</FilesMatch>

<FilesMatch "config\.php$">
Order allow,deny
Deny from all
</FilesMatch>
```

REMÉDIATION :

1. Déplacement fichiers sensibles :

```
// Déplacer config hors web root
function secure_plugin_config() {
    $config_dir = ABSPATH . '../plugin-configs/';
    if (!is_dir($config_dir)) {
        mkdir($config_dir, 0700, true);
    }

    // Rediriger les plugins vers ce répertoire
    define('SECURE_CONFIG_DIR', $config_dir);
}
add_action('plugins_loaded', 'secure_plugin_config', 1);
```

REMÉDIATION :

1. Chiffrement des configurations :

```
function encrypt_plugin_config($data) {
    $key = wp_salt('AUTH_KEY');
    return base64_encode(openssl_encrypt($data, 'AES-256-CBC', $key, 0, substr($key, 0, 16)));
}

function decrypt_plugin_config($encrypted_data) {
    $key = wp_salt('AUTH_KEY');
    return openssl_decrypt(base64_decode($encrypted_data), 'AES-256-CBC', $key, 0, substr($key, 0, 16));
}
```

VALEUR PAR DÉFAUT :

Fichiers config potentiellement accessibles

3.1.18 *Isolation Plugins Critiques*

MITRE ATT&CK : T1505

DESCRIPTION :

Isoler les plugins critiques (sécurité, backup) pour éviter les interférences et garantir leur fonctionnement même en cas de problème avec d'autres plugins.

```
# Identifier plugins critiques
wp plugin list | grep -iE "(security|backup|firewall|malware)"
# Vérifier conflits potentiels
wp plugin list --status=active --format=count
```

REMÉDIATION :

1. Must-Use Plugins pour critiques :

```
# Déplacer plugins critiques vers mu-plugins
mkdir -p wp-content/mu-plugins
cp -r wp-content/plugins/wordfence wp-content/mu-plugins/
```

REMÉDIATION :

1. Chargement prioritaire :

```
// mu-plugins/critical-plugins-loader.php
function load_critical_plugins_first() {
    $critical_plugins = array(
        'wordfence/wordfence.php',
        'updraftplus/updraftplus.php'
    );

    foreach ($critical_plugins as $plugin) {
        if (file_exists(WP_PLUGIN_DIR . '/' . $plugin)) {
            include_once WP_PLUGIN_DIR . '/' . $plugin;
        }
    }
}
add_action('muplugins_loaded', 'load_critical_plugins_first', 1);
```

REMÉDIATION :

1. Monitoring isolation :

```
function monitor_plugin_isolation() {
    $critical_plugins = array('wordfence', 'updraftplus');

    foreach ($critical_plugins as $plugin) {
        if (!is_plugin_active($plugin . '/' . $plugin . '.php')) {
            wp_mail('admin@example.com', 'Plugin critique inactif',
                    "Le plugin critique $plugin est inactif!");
        }
    }
}
wp_schedule_event(time(), 'hourly', 'monitor_plugin_isolation');
```

VALEUR PAR DÉFAUT :

Tous plugins au même niveau

3.1.19 *Gestion Licences et Mises à Jour Premium*

MITRE ATT&CK : T1195

DESCRIPTION :

Gérer correctement les licences des plugins premium pour assurer la réception des mises à jour de sécurité et éviter l'utilisation de versions piratées.

```
# Plugins premium installés
wp plugin list --format=json | jq '.[] | select(.update == "available" and .version != .update_version)'
# Vérifier licences expirées
grep -r "license.*expir\|invalid.*license" wp-content/plugins/
```

REMÉDIATION :

1. Audit des licences :

```
function audit_premium_licenses() {
    $premium_plugins = array(
        'gravityforms' => get_option('rg_gforms_key'),
        'wpforms-lite' => get_option('wpforms_license_key'),
        'elementor-pro' => get_option('elementor_pro_license_key')
    );

    foreach ($premium_plugins as $plugin => $license) {
        if (empty($license) || $this->is_license_expired($license)) {
            wp_mail('admin@example.com', 'Licence plugin expirée',
                    "La licence du plugin $plugin nécessite un renouvellement.");
        }
    }
}
wp_schedule_event(time(), 'weekly', 'audit_premium_licenses');
```

REMÉDIATION :

1. Alertes expiration :

```
function check_license_expiration($plugin_name, $license_data) {
    if (isset($license_data['expires']) &&
        strtotime($license_data['expires']) < strtotime('+30 days')) {

        // Alerte 30 jours avant expiration
        wp_mail('admin@example.com', 'Licence bientôt expirée',
                "La licence de $plugin_name expire le {$license_data['expires']}");
    }
}
```

VALEUR PAR DÉFAUT :

Licences non surveillées

3.1.20 *Tests Sécurité Automatisés*

MITRE ATT&CK : T1190

DESCRIPTION :

Implémenter des tests de sécurité automatisés pour détecter les vulnérabilités dans les plugins après chaque installation ou mise à jour.

```
# Scanner automatique avec WPScan
wpscan --url https://example.com --enumerate vp --format json > plugin_scan.json
# Analyser résultats
jq '.plugins[].vulnerabilities[]?' plugin_scan.json
```

REMÉDIATION :

1. Pipeline de tests automatisés :

```
#!/bin/bash
# tests/security-scan.sh
SITE_URL=$1
SCAN_RESULTS="/tmp/wpscan_$(date +%s).json"

# Scan plugins
wpscan --url $SITE_URL --enumerate vp --format json --output $SCAN_RESULTS

# Analyser vulnérabilités
VULN_COUNT=$(jq '[.plugins[].vulnerabilities[]?] | length' $SCAN_RESULTS)

if [ $VULN_COUNT -gt 0 ]; then
    echo "ALERTE: $VULN_COUNT vulnérabilités détectées"
    jq '.plugins[].vulnerabilities[]?' $SCAN_RESULTS | mail -s "Vulnérabilités plugins détectées" admin@example.com
    exit 1
fi
```

REMÉDIATION :

1. Intégration continue :

```
function automated_security_scan() {
    // Après mise à jour plugin
    $command = "wpscan --url " . site_url() . " --enumerate vp --format json";
    $output = shell_exec($command);
    $results = json_decode($output, true);

    if (isset($results['plugins'])) {
        foreach ($results['plugins'] as $plugin => $data) {
            if (!empty($data['vulnerabilities'])) {
                // Désactiver plugin vulnérable
                deactivate_plugins($plugin);
                wp_mail('admin@example.com', 'Plugin vulnérable désactivé',
                        "Plugin $plugin désactivé automatiquement - vulnérabilités détectées");
            }
        }
    }
}
add_action('upgrader_process_complete', 'automated_security_scan');
```

VALEUR PAR DÉFAUT :

Tests manuels uniquement

3.1.21 *Chiffrement Données Plugins*

MITRE ATT&CK : T1005

DESCRIPTION :

Assurer que les plugins traitant des données sensibles implémentent un chiffrement approprié pour le stockage et la transmission.

```
# Rechercher données sensibles non chiffrées
grep -r "password\|api.*key\|secret" wp-content/plugins/ | grep -v "encrypt\|hash\|bcrypt"
# Vérifier utilisation SSL pour API
grep -r "http://" wp-content/plugins/ | grep -v "localhost"
```

REMÉDIATION :

1. Audit chiffrement des données :

```
function audit_plugin_encryption() {
    $sensitive_data_patterns = array(
        'api_key' => '/api.*key.*[\'"]([^\'"]{20,})[\'"]/',
        'password' => '/password.*[\'"]([^\'"]{8,})[\'"]/',
        'secret' => '/secret.*[\'"]([^\'"]{16,})[\'"]/'
    );

    $plugins_dir = WP_PLUGIN_DIR;
    $unencrypted_data = array();

    foreach (glob("$plugins_dir/*/*.php") as $file) {
        $content = file_get_contents($file);
        foreach ($sensitive_data_patterns as $type => $pattern) {
            if (preg_match($pattern, $content, $matches)) {
                $unencrypted_data[] = array(
                    'file' => $file,
                    'type' => $type,
                    'data' => substr($matches[1], 0, 10) . '...'
                );
            }
        }
    }

    if (!empty($unencrypted_data)) {
        wp_mail('admin@example.com', 'Données sensibles non chiffrées',
                'Données sensibles détectées: ' . print_r($unencrypted_data, true));
    }
}
```

REMÉDIATION :

1. Helper de chiffrement pour plugins :

```
class SecurePluginStorage {
    private static function get_key() {
        return wp_salt('AUTH_KEY');
    }

    public static function encrypt($data) {
        $key = self::get_key();
        $iv = openssl_random_pseudo_bytes(16);
        $encrypted = openssl_encrypt($data, 'AES-256-CBC', $key, 0, $iv);
        return base64_encode($iv . $encrypted);
    }

    public static function decrypt($encrypted_data) {
        $key = self::get_key();
        $data = base64_decode($encrypted_data);
        $iv = substr($data, 0, 16);
        $encrypted = substr($data, 16);
        return openssl_decrypt($encrypted, 'AES-256-CBC', $key, 0, $iv);
    }
}
```

VALEUR PAR DÉFAUT :

Stockage données sensibles en clair

3.1.22 *Contrôle Accès Réseau Plugins*

MITRE ATT&CK : T1071

DESCRIPTION :

Contrôler et limiter les accès réseau des plugins pour prévenir l'exfiltration de données et les communications avec des serveurs malveillants.

```
# Analyser appels réseau des plugins
grep -r "wp_remote_\|curl\|file_get_contents.*http" wp-content/plugins/ | head -20
# Vérifier domaines contactés
netstat -an | grep :80 | grep ESTABLISHED
```

REMÉDIATION :

1. Whitelist des domaines autorisés :

```
function filter_plugin_http_requests($result, $args, $url) {
    $allowed_domains = array(
        'api.wordpress.org',
        'downloads.wordpress.org',
        'secure.gravatar.com',
        'fonts.googleapis.com'
    );

    $domain = parse_url($url, PHP_URL_HOST);

    if (!in_array($domain, $allowed_domains)) {
        // Logger tentative accès non autorisée
        error_log("BLOCKED HTTP request to: $url");
        return new WP_Error('http_request_blocked', 'Domaine non autorisé');
    }

    return $result;
}
add_filter('pre_http_request', 'filter_plugin_http_requests', 10, 3);
```

REMÉDIATION :

1. Monitoring communications :

```
function monitor_plugin_communications($response, $args, $url) {
    $suspicious_patterns = array(
        'base64', 'eval', 'exec', 'system', 'shell_exec'
    );

    if (isset($args['body'])) {
        foreach ($suspicious_patterns as $pattern) {
            if (strpos($args['body'], $pattern) !== false) {
                wp_mail('admin@example.com', 'Communication suspecte détectée',
                        "Requête suspecte vers $url avec pattern: $pattern");
                break;
            }
        }
    }

    return $response;
}
add_filter('http_response', 'monitor_plugin_communications', 10, 3);
```

VALEUR PAR DÉFAUT :

Accès réseau illimité

3.1.23 *Validation Entrées Plugins*

MITRE ATT&CK : T1190

DESCRIPTION :

Vérifier que les plugins valident et assainissent correctement toutes les entrées utilisateurs pour prévenir les injections et autres attaques.

```
# Rechercher validation insuffisante
grep -r "\$_POST\|\$_GET\|\$_REQUEST" wp-content/plugins/ | grep -v "sanitize\|escape\|validate"
# Analyser fonctions de sanitisation
grep -r "sanitize_\|wp_kses\|esc_" wp-content/plugins/ | wc -l
```

REMÉDIATION :

1. Audit validation des entrées :

```
function audit_plugin_input_validation() {
    $plugins_dir = WP_PLUGIN_DIR;
    $validation_issues = array();

    foreach (glob("$plugins_dir/*/*.php") as $file) {
        $content = file_get_contents($file);

        // Rechercher $_POST, $_GET non sécurisés
        if (preg_match_all('/\$_(POST|GET|REQUEST)\[[\'"]([^\'"]+)[\'"]\]/', $content, $matches, PREG_OFFSET_CAPTURE)) {
            foreach ($matches[0] as $index => $match) {
                $line_start = strrpos(substr($content, 0, $match[1]), "\n");
                $line_end = strpos($content, "\n", $match[1]);
                $line = substr($content, $line_start, $line_end - $line_start);

                // Vérifier si la ligne contient une fonction de sanitisation
                if (!preg_match('/(sanitize_|wp_kses|esc_|intval|absint|wp_verify_nonce)/', $line)) {
                    $validation_issues[] = array(
                        'file' => $file,
                        'variable' => $match[0],
                        'line' => $line
                    );
                }
            }
        }
    }

    if (!empty($validation_issues)) {
        wp_mail('admin@example.com', 'Problèmes validation détectés',
                'Entrées non validées: ' . print_r($validation_issues, true));
    }
}
```

REMÉDIATION :

1. Renforcement global validation :

```
function enforce_input_validation() {
    // Sanitiser automatiquement toutes les entrées
    $_POST = array_map('sanitize_text_field', $_POST);
    $_GET = array_map('sanitize_text_field', $_GET);
    $_REQUEST = array_map('sanitize_text_field', $_REQUEST);
}
add_action('init', 'enforce_input_validation', 1);
```

VALEUR PAR DÉFAUT :

Validation dépendante du développeur plugin

3.1.24 *Gestion Erreurs et Exceptions Plugins*

MITRE ATT&CK : T1005

DESCRIPTION :

Assurer que les plugins gèrent correctement les erreurs sans révéler d'informations sensibles et avec une logging appropriée.

```
# Rechercher gestion d'erreurs
grep -r "try\|catch\|throw\|error_log" wp-content/plugins/ | wc -l
# Vérifier exposition d'erreurs
grep -r "print_r\|var_dump\|echo.*error" wp-content/plugins/
```

REMÉDIATION :

1. Standards gestion d'erreurs :

```
function standardize_plugin_error_handling() {
    // Gestionnaire d'erreurs global pour plugins
    set_error_handler(function($severity, $message, $file, $line) {
        if (strpos($file, WP_PLUGIN_DIR) !== false) {
            // Log sécurisé pour erreurs plugins
            error_log("PLUGIN_ERROR: [$severity] $message in $file:$line");

            // En production, ne pas afficher les erreurs
            if (wp_get_environment_type() === 'production') {
                return true; // Supprime l'affichage
            }
        }
        return false; // Laisse le gestionnaire par défaut
    });
}
add_action('plugins_loaded', 'standardize_plugin_error_handling', 1);
```

REMÉDIATION :

1. Wrapper sécurisé pour plugins :

```
class SecurePluginLogger {
    public static function log_error($plugin_name, $message, $data = null) {
        $log_entry = array(
            'plugin' => $plugin_name,
            'timestamp' => current_time('mysql'),
            'message' => $message,
            'ip' => $_SERVER['REMOTE_ADDR'],
            'user_agent' => $_SERVER['HTTP_USER_AGENT']
        );

        if ($data) {
            // Nettoyer données sensibles avant log
            $log_entry['data'] = $this->sanitize_log_data($data);
        }

        error_log('SECURE_PLUGIN_LOG: ' . json_encode($log_entry));
    }

    private static function sanitize_log_data($data) {
        $sensitive_keys = array('password', 'api_key', 'secret', 'token');

        if (is_array($data)) {
            foreach ($data as $key => $value) {
                if (in_array(strtolower($key), $sensitive_keys)) {
                    $data[$key] = '***REDACTED***';
                }
            }
        }

        return $data;
    }
}
```

VALEUR PAR DÉFAUT :

Gestion d'erreurs variable selon plugins

3.1.25 *Documentation et Traçabilité Plugins*

MITRE ATT&CK : T1070

DESCRIPTION :

Maintenir une documentation complète des plugins installés, leurs fonctions, configurations et historique des modifications pour faciliter les audits et investigations.

```
# Documentation existante
find . -name "*plugin*doc*" -o -name "*README*" -o -name "*CHANGELOG*"
# Historique modifications
wp plugin list --format=csv --fields=name,version,status > current_plugins.csv
diff previous_plugins.csv current_plugins.csv
```

REMÉDIATION :

1. Documentation automatisée :

```
function generate_plugin_documentation() {
    $plugins = get_plugins();
    $active_plugins = get_option('active_plugins');
    $documentation = array();

    foreach ($plugins as $plugin_file => $plugin_data) {
        $doc_entry = array(
            'name' => $plugin_data['Name'],
            'version' => $plugin_data['Version'],
            'description' => $plugin_data['Description'],
            'author' => $plugin_data['Author'],
            'active' => in_array($plugin_file, $active_plugins),
            'last_updated' => get_plugin_data(WP_PLUGIN_DIR . '/' . $plugin_file)['Version'],
            'security_audit_date' => get_option("plugin_audit_$plugin_file", 'Never'),
            'configuration' => $this->get_plugin_configuration($plugin_file)
        );

        $documentation[] = $doc_entry;
    }

    // Générer rapport
    file_put_contents(ABSPATH . 'plugin-documentation.json',
                      json_encode($documentation, JSON_PRETTY_PRINT));

    return $documentation;
}
```

REMÉDIATION :

1. Historique des changements :

```
function track_plugin_changes($plugin, $network_activation) {
    $change_log = get_option('plugin_change_log', array());

    $change_entry = array(
        'timestamp' => current_time('mysql'),
        'plugin' => $plugin,
        'action' => current_filter(), // activated_plugin, deactivated_plugin
        'user' => wp_get_current_user()->user_login,
        'ip' => $_SERVER['REMOTE_ADDR'],
        'user_agent' => $_SERVER['HTTP_USER_AGENT']
    );

    $change_log[] = $change_entry;

    // Garder seulement les 1000 dernières entrées
    if (count($change_log) > 1000) {
        $change_log = array_slice($change_log, -1000);
    }

    update_option('plugin_change_log', $change_log);
}
add_action('activated_plugin', 'track_plugin_changes', 10, 2);
add_action('deactivated_plugin', 'track_plugin_changes', 10, 2);
```

VALEUR PAR DÉFAUT :

Documentation manuelle et incomplète

4.0 — THÈMES

4.1.1 *Thème Actif Sécurisé et À Jour*

MITRE ATT&CK : T1190

DESCRIPTION :

Le thème actif doit être maintenu à jour et exempt de vulnérabilités connues. Les thèmes obsolètes constituent une porte d'entrée majeure pour les attaquants.

```
# Vérifier thème actif et version
wp theme list --status=active --format=table --fields=name,version,update_version
# Scanner vulnérabilités thèmes
wpscan --url https://example.com --enumerate vt --no-banner
# Vérifier dernière mise à jour
wp theme get $(wp theme list --status=active --field=name) --field=version
```

REMÉDIATION :

1. Mise à jour immédiate du thème :

```
wp theme update --all
wp theme update $(wp theme list --status=active --field=name)
```

REMÉDIATION :

1. Auto-update pour le thème actif :

```
wp theme auto-updates enable $(wp theme list --status=active --field=name)
```

REMÉDIATION :

1. Surveillance des mises à jour :

```
function monitor_theme_updates() {
    $current_theme = get_stylesheet();
    $theme_updates = get_site_transient('update_themes');

    if (isset($theme_updates->response[$current_theme])) {
        wp_mail('admin@example.com', 'Mise à jour thème disponible',
                "Une mise à jour est disponible pour le thème actif: $current_theme");
    }
}
wp_schedule_event(time(), 'daily', 'monitor_theme_updates');
```

VALEUR PAR DÉFAUT :

Mises à jour manuelles

4.1.2 *Suppression Thèmes Inactifs*

MITRE ATT&CK : T1190

DESCRIPTION :

Les thèmes inactifs représentent une surface d'attaque inutile. Ils peuvent contenir des vulnérabilités exploitables même sans être activés et doivent être supprimés.

```
# Lister thèmes inactifs
wp theme list --status=inactive --format=table
# Compter thèmes inactifs
wp theme list --status=inactive --format=count
# Vérifier vulnérabilités dans thèmes inactifs
wpscan --url https://example.com --enumerate vt --no-banner
```

REMÉDIATION :

1. Supprimer tous les thèmes inactifs :

```
wp theme delete $(wp theme list --status=inactive --field=name)
```

REMÉDIATION :

1. Garder seulement un thème de fallback :

```
# Garder un thème WordPress par défaut comme backup
wp theme install twentytwentythree
wp theme delete $(wp theme list --status=inactive --field=name | grep -v twentytwentythree)
```

REMÉDIATION :

1. Politique de suppression automatique :

```
function cleanup_inactive_themes() {
    $all_themes = wp_get_themes();
    $active_theme = get_stylesheet();
    $parent_theme = get_template();

    // Garder seulement le thème actif et son parent (si child theme)
    $keep_themes = array($active_theme, $parent_theme, 'twentytwentythree');

    foreach ($all_themes as $theme_slug => $theme) {
        if (!in_array($theme_slug, $keep_themes)) {
            delete_theme($theme_slug);
        }
    }
}
// Exécuter lors des mises à jour majeures WordPress
add_action('upgrader_process_complete', 'cleanup_inactive_themes');
```

VALEUR PAR DÉFAUT :

Accumulation de thèmes inactifs

4.1.3 *Child Theme Obligatoire*

MITRE ATT&CK : T1565

DESCRIPTION :

Utiliser un child theme préserve les personnalisations lors des mises à jour du thème parent et évite la perte de modifications de sécurité personnalisées.

```
# Vérifier si child theme utilisé
wp theme list --status=active --format=json | jq '.[].parent'
# Examiner structure du thème
ls -la wp-content/themes/$(wp theme list --status=active --field=name)/
# Vérifier fichier style.css
head -10 wp-content/themes/$(wp theme list --status=active --field=name)/style.css
```

REMÉDIATION :

1. Créer un child theme :

```
# Créer répertoire child theme
PARENT_THEME=$(wp theme list --status=active --field=name)
CHILD_THEME="${PARENT_THEME}-child"
mkdir wp-content/themes/$CHILD_THEME

# Créer style.css
cat > wp-content/themes/$CHILD_THEME/style.css << EOL
/*
Theme Name: $CHILD_THEME
Description: Child theme de $PARENT_THEME
Template: $PARENT_THEME
Version: 1.0
*/

@import url("../$PARENT_THEME/style.css");
EOL

# Créer functions.php
cat > wp-content/themes/$CHILD_THEME/functions.php << 'EOL'
<?php
function child_theme_enqueue_styles() {
    wp_enqueue_style('parent-style', get_template_directory_uri() . '/style.css');
    wp_enqueue_style('child-style', get_stylesheet_directory_uri() . '/style.css', array('parent-style'));
}
add_action('wp_enqueue_scripts', 'child_theme_enqueue_styles');
EOL

# Activer le child theme
wp theme activate $CHILD_THEME
```

VALEUR PAR DÉFAUT :

Thème parent directement modifié

4.1.4 *Détection Thèmes Nulled/Piratés*

MITRE ATT&CK : T1505

DESCRIPTION :

Les thèmes piratés contiennent souvent des backdoors et malwares. Détecter et éliminer ces thèmes est critique pour la sécurité.

```
# Scanner code malveillant dans thèmes
find wp-content/themes/ -name "*.php" -exec grep -l "eval\|base64_decode\|gzinflate\|str_rot13" {} \;
# Vérifier intégrité avec checksums
wp theme verify-checksums --all
# Rechercher backdoors connues
grep -r "c99\|r57\|WSO\|FilesMan" wp-content/themes/
```

REMÉDIATION :

1. Supprimer thèmes suspects :

```
# Identifier et supprimer thèmes compromis
find wp-content/themes/ -name "*.php" -exec grep -l "eval.*base64_decode" {} \; | head -5
# Supprimer thème suspect (remplacer par nom réel)
wp theme delete suspicious-theme
```

REMÉDIATION :

1. Scanner automatisé malware :

```
#!/bin/bash
# Script de détection malware thèmes
MALWARE_PATTERNS="eval.*base64_decode|gzinflate|str_rot13|system.*\$_|passthru"

find wp-content/themes/ -name "*.php" | while read file; do
    if grep -qE "$MALWARE_PATTERNS" "$file"; then
        echo "SUSPECT: $file"
        # Quarantaine
        mv "$file" "$file.quarantine"
    fi
done
```

REMÉDIATION :

1. Validation source thèmes :

```
function validate_theme_source($theme_slug) {
    // Vérifier si thème provient du dépôt officiel WordPress
    $response = wp_remote_get("https://api.wordpress.org/themes/info/1.1/?action=theme_information&request[slug]=$theme_slug");

    if (is_wp_error($response) || wp_remote_retrieve_response_code($response) !== 200) {
        // Thème non officiel - vérifier intégrité
        $theme_path = get_theme_root() . "/$theme_slug";
        $suspicious_files = array();

        foreach (glob("$theme_path/*.php") as $file) {
            $content = file_get_contents($file);
            if (preg_match('/(eval|base64_decode|gzinflate).*\(/', $content)) {
                $suspicious_files[] = $file;
            }
        }

        if (!empty($suspicious_files)) {
            wp_mail('admin@example.com', 'Thème suspect détecté',
                    "Fichiers suspects dans $theme_slug: " . implode(', ', $suspicious_files));
        }
    }
}
```

VALEUR PAR DÉFAUT :

Aucune vérification automatique

4.1.5 *Désactivation Éditeur de Thème*

MITRE ATT&CK : T1105

DESCRIPTION :

L'éditeur de thème permet la modification de code PHP directement depuis l'administration WordPress. Cette fonction doit être désactivée pour éviter l'injection de code malveillant.

```
# Vérifier si éditeur désactivé
grep "DISALLOW_FILE_EDIT" wp-config.php
# Tester accès éditeur de thème
curl -s https://example.com/wp-admin/theme-editor.php | grep -i "file editing"
```

REMÉDIATION :

1. Désactiver l'éditeur dans wp-config.php :

```
define('DISALLOW_FILE_EDIT', true);
```

REMÉDIATION :

1. Vérifier suppression du menu :

```
// Vérifier que le menu "Éditeur" n'est plus visible
add_action('admin_init', function() {
    if (current_user_can('edit_themes')) {
        remove_submenu_page('themes.php', 'theme-editor.php');
    }
});
```

REMÉDIATION :

1. Protection supplémentaire via .htaccess :

```
<Files "theme-editor.php">
Order allow,deny
Deny from all
</Files>
<Files "plugin-editor.php">
Order allow,deny
Deny from all
</Files>
```

VALEUR PAR DÉFAUT :

Éditeur de thème activé

4.1.6 *Audit Vulnérabilités Thèmes*

MITRE ATT&CK : T1190

DESCRIPTION :

Effectuer des audits réguliers des thèmes contre les bases de vulnérabilités connues pour identifier et corriger rapidement les failles de sécurité.

```
# Scanner avec WPScan
wpscan --url https://example.com --enumerate vt --api-token YOUR_TOKEN --no-banner
# Vérifier base CVE
curl -s "https://cve.circl.lu/api/search/wordpress theme" | jq '.[] | select(.summary | contains("theme"))'
```

REMÉDIATION :

1. Audit automatisé quotidien :

```
#!/bin/bash
# Script d'audit thèmes
wpscan --url https://example.com --enumerate vt --format json --api-token $API_TOKEN > theme_scan.json

# Analyser résultats
if jq -e '.themes[].vulnerabilities[]?' theme_scan.json > /dev/null; then
    echo "ALERTE: Vulnérabilités détectées dans les thèmes"
    jq '.themes[].vulnerabilities[]?' theme_scan.json | mail -s "Vulnérabilités thèmes" admin@example.com
fi
```

REMÉDIATION :

1. Monitoring continu :

```
function continuous_theme_monitoring() {
    $themes = wp_get_themes();

    foreach ($themes as $theme_slug => $theme) {
        // Vérifier version contre base de vulnérabilités
        $version = $theme->get('Version');
        $response = wp_remote_get("https://wpscan.com/api/v3/themes/$theme_slug");

        if (!is_wp_error($response)) {
            $data = json_decode(wp_remote_retrieve_body($response), true);
            if (isset($data['vulnerabilities']) && !empty($data['vulnerabilities'])) {
                // Alerter sur vulnérabilités
                wp_mail('admin@example.com', 'Vulnérabilité thème détectée',
                        "Vulnérabilité dans $theme_slug version $version");
            }
        }
    }
}
wp_schedule_event(time(), 'daily', 'continuous_theme_monitoring');
```

VALEUR PAR DÉFAUT :

Audits manuels occasionnels

4.1.7 *Validation Code Personnalisé Thèmes*

MITRE ATT&CK : T1190

DESCRIPTION :

Valider tout code personnalisé ajouté aux thèmes pour s'assurer qu'il respecte les bonnes pratiques de sécurité WordPress et ne contient pas de vulnérabilités.

```
# Analyser code personnalisé
find wp-content/themes/ -name "functions.php" -exec grep -l "add_action\|add_filter" {} \;
# Vérifier échappement des sorties
grep -r "echo\|print" wp-content/themes/ | grep -v "esc_\|wp_kses"
# Analyser validation des entrées
grep -r "\$_POST\|\$_GET" wp-content/themes/ | grep -v "sanitize"
```

REMÉDIATION :

1. Audit sécurisé du code :

```
function audit_theme_security() {
    $theme_dir = get_stylesheet_directory();
    $security_issues = array();

    // Patterns de sécurité à vérifier
    $patterns = array(
        'unescaped_output' => '/echo\s+\$[^;]*(?!.*esc_)/i',
        'unsanitized_input' => '/\$_(POST|GET|REQUEST)\[[^\]]+\](?!.*sanitize)/i',
        'dangerous_functions' => '/(eval|exec|system|shell_exec|passthru)\s*\(/i'
    );

    foreach (glob("$theme_dir/*.php") as $file) {
        $content = file_get_contents($file);

        foreach ($patterns as $type => $pattern) {
            if (preg_match_all($pattern, $content, $matches, PREG_OFFSET_CAPTURE)) {
                $security_issues[$file][$type] = $matches[0];
            }
        }
    }

    if (!empty($security_issues)) {
        wp_mail('admin@example.com', 'Problèmes sécurité thème détectés',
                print_r($security_issues, true));
    }

    return $security_issues;
}
```

REMÉDIATION :

1. Standards de développement sécurisé :

```
// Exemple de code sécurisé pour functions.php
function secure_theme_customization() {
    // Validation et sanitisation correctes
    if (isset($_POST['custom_field']) && wp_verify_nonce($_POST['nonce'], 'custom_action')) {
        $safe_value = sanitize_text_field($_POST['custom_field']);
        update_option('theme_custom_field', $safe_value);
    }

    // Échappement correct des sorties
    $option_value = get_option('theme_custom_field', '');
    echo esc_html($option_value);
}
```

VALEUR PAR DÉFAUT :

Code personnalisé non audité

4.1.8 *Protection Fichiers Thème Sensibles*

MITRE ATT&CK : T1005

DESCRIPTION :

Protéger l'accès aux fichiers sensibles des thèmes (functions.php, configuration, etc.) contre l'accès web direct non autorisé.

```
# Tester accès direct aux fichiers thème
curl -s https://example.com/wp-content/themes/active-theme/functions.php
curl -s https://example.com/wp-content/themes/active-theme/config.php
# Vérifier permissions fichiers
ls -la wp-content/themes/$(wp theme list --status=active --field=name)/
```

REMÉDIATION :

1. Protection via .htaccess dans le thème :

```
# wp-content/themes/theme-name/.htaccess
<FilesMatch "\.(php|inc|conf)$">
Order allow,deny
Deny from all
</FilesMatch>

# Autoriser seulement index.php et style.css publiquement
<FilesMatch "^(index\.php|style\.css|screenshot\.(png|jpg))$">
Order allow,deny
Allow from all
</FilesMatch>
```

REMÉDIATION :

1. En-têtes de sécurité PHP :

```
// Ajouter en début de functions.php
if (!defined('ABSPATH')) {
    exit; // Sortie si accès direct
}

// Protection supplémentaire
if (!function_exists('wp_head')) {
    http_response_code(403);
    die('Accès direct interdit');
}
```

REMÉDIATION :

1. Permissions système appropriées :

```
# Permissions sécurisées pour thèmes
find wp-content/themes/ -type f -name "*.php" -exec chmod 644 {} \;
find wp-content/themes/ -type d -exec chmod 755 {} \;
# functions.php plus restrictif
chmod 640 wp-content/themes/*/functions.php
```

VALEUR PAR DÉFAUT :

Fichiers potentiellement accessibles

4.1.9 *Gestion Versions et Rollback Thèmes*

MITRE ATT&CK : T1565

DESCRIPTION :

Maintenir un système de versioning des thèmes et la capacité de rollback rapide en cas de problème lors des mises à jour.

```
# Historique des versions
wp theme list --format=json | jq '.[] | {name: .name, version: .version}'
# Backups existants
ls -la backups/themes/ 2>/dev/null || echo "Pas de backups thèmes"
```

REMÉDIATION :

1. Backup automatique avant mises à jour :

```
#!/bin/bash
# Script de backup thème avant mise à jour
THEME_NAME=$(wp theme list --status=active --field=name)
BACKUP_DIR="backups/themes/$(date +%Y%m%d_%H%M%S)"

mkdir -p $BACKUP_DIR
cp -r wp-content/themes/$THEME_NAME $BACKUP_DIR/
echo "Backup thème créé: $BACKUP_DIR"

# Garder seulement les 10 derniers backups
ls -t backups/themes/ | tail -n +11 | xargs -r rm -rf
```

REMÉDIATION :

1. Système de rollback :

```
function theme_rollback_system() {
    if (isset($_POST['rollback_theme']) && wp_verify_nonce($_POST['nonce'], 'theme_rollback')) {
        $backup_path = sanitize_text_field($_POST['backup_path']);
        $current_theme = get_stylesheet();

        if (is_dir("backups/themes/$backup_path")) {
            // Backup actuel avant rollback
            $current_backup = "backups/themes/pre-rollback-" . date('Y-m-d-H-i-s');
            mkdir($current_backup, 0755, true);
            exec("cp -r wp-content/themes/$current_theme $current_backup/");

            // Restaurer backup
            exec("rm -rf wp-content/themes/$current_theme");
            exec("cp -r backups/themes/$backup_path/$current_theme wp-content/themes/");

            wp_redirect(admin_url('themes.php?rollback=success'));
            exit;
        }
    }
}
add_action('admin_init', 'theme_rollback_system');
```

VALEUR PAR DÉFAUT :

Pas de système de rollback

4.1.10 *Monitoring Modifications Thèmes*

MITRE ATT&CK : T1070

DESCRIPTION :

Surveiller toutes les modifications des fichiers de thèmes pour détecter les changements non autorisés ou l'injection de code malveillant.

```
# Vérifier intégrité fichiers thème
find wp-content/themes/ -name "*.php" -exec md5sum {} \; > theme_checksums.txt
# Comparer avec checksums précédents
diff theme_checksums_previous.txt theme_checksums.txt
```

REMÉDIATION :

1. Monitoring intégrité fichiers :

```
function monitor_theme_file_integrity() {
    $theme_dir = get_stylesheet_directory();
    $stored_hashes = get_option('theme_file_hashes', array());
    $current_hashes = array();
    $changes_detected = false;

    foreach (glob("$theme_dir/*.php") as $file) {
        $relative_path = str_replace($theme_dir . '/', '', $file);
        $current_hash = md5_file($file);
        $current_hashes[$relative_path] = $current_hash;

        if (isset($stored_hashes[$relative_path]) &&
            $stored_hashes[$relative_path] !== $current_hash) {
            // Fichier modifié
            $changes_detected = true;
            $this->log_file_change($file, 'modified');
        } elseif (!isset($stored_hashes[$relative_path])) {
            // Nouveau fichier
            $changes_detected = true;
            $this->log_file_change($file, 'added');
        }
    }

    // Vérifier fichiers supprimés
    foreach ($stored_hashes as $file => $hash) {
        if (!isset($current_hashes[$file])) {
            $changes_detected = true;
            $this->log_file_change("$theme_dir/$file", 'deleted');
        }
    }

    if ($changes_detected) {
        wp_mail('admin@example.com', 'Modifications thème détectées',
                'Des modifications ont été détectées dans les fichiers du thème.');
    }

    update_option('theme_file_hashes', $current_hashes);
}

function log_file_change($file, $action) {
    $log_entry = array(
        'timestamp' => current_time('mysql'),
        'file' => $file,
        'action' => $action,
        'ip' => $_SERVER['REMOTE_ADDR'] ?? 'unknown',
        'user_agent' => $_SERVER['HTTP_USER_AGENT'] ?? 'unknown'
    );

    error_log('THEME_FILE_CHANGE: ' . json_encode($log_entry));
}

wp_schedule_event(time(), 'hourly', 'monitor_theme_file_integrity');
```

VALEUR PAR DÉFAUT :

Modifications non surveillées

4.1.11 *Validation Headers et Métadonnées Thèmes*

MITRE ATT&CK : T1195

DESCRIPTION :

Valider les headers et métadonnées des thèmes pour s'assurer de leur authenticité et détecter les modifications malveillantes.

```
# Examiner headers des thèmes
head -20 wp-content/themes/*/style.css
# Vérifier métadonnées
grep -r "Theme Name\|Version\|Author" wp-content/themes/*/style.css
```

REMÉDIATION :

1. Validation des métadonnées :

```
function validate_theme_metadata() {
    $themes = wp_get_themes();
    $validation_results = array();

    foreach ($themes as $theme_slug => $theme) {
        $theme_data = array(
            'name' => $theme->get('Name'),
            'version' => $theme->get('Version'),
            'author' => $theme->get('Author'),
            'description' => $theme->get('Description'),
            'uri' => $theme->get('ThemeURI')
        );

        // Vérifier cohérence des métadonnées
        $issues = array();

        // Version suspecte
        if (preg_match('/[<>"\']/', $theme_data['version'])) {
            $issues[] = 'Version contient des caractères suspects';
        }

        // Auteur suspect
        if (preg_match('/(nulled|cracked|free|download)/i', $theme_data['author'])) {
            $issues[] = 'Auteur suspect (possiblement piraté)';
        }

        // URI suspecte
        if (!empty($theme_data['uri']) && !filter_var($theme_data['uri'], FILTER_VALIDATE_URL)) {
            $issues[] = 'URI invalide';
        }

        if (!empty($issues)) {
            $validation_results[$theme_slug] = $issues;
        }
    }

    if (!empty($validation_results)) {
        wp_mail('admin@example.com', 'Thèmes avec métadonnées suspectes',
                print_r($validation_results, true));
    }

    return $validation_results;
}
```

VALEUR PAR DÉFAUT :

Métadonnées non validées

4.1.12 *Contrôle Fonctionnalités Thème*

MITRE ATT&CK : T1505

DESCRIPTION :

Auditer et contrôler les fonctionnalités activées par le thème pour s'assurer qu'elles sont nécessaires et sécurisées.

```
# Analyser fonctionnalités du thème
grep -r "add_theme_support\|remove_theme_support" wp-content/themes/
# Vérifier hooks et actions
grep -r "add_action\|add_filter" wp-content/themes/*/functions.php
```

REMÉDIATION :

1. Audit des fonctionnalités :

```
function audit_theme_features() {
    $theme_supports = array();
    $dangerous_features = array('post-thumbnails', 'custom-header', 'custom-background');

    // Lister toutes les fonctionnalités supportées
    global $_wp_theme_features;
    foreach ($_wp_theme_features as $feature => $data) {
        if (current_theme_supports($feature)) {
            $theme_supports[] = $feature;
        }
    }

    // Analyser actions/filters du thème
    $theme_hooks = array();
    $functions_file = get_stylesheet_directory() . '/functions.php';

    if (file_exists($functions_file)) {
        $content = file_get_contents($functions_file);
        preg_match_all('/add_(action|filter)\s*\(\s*[\'"]([^\'"]+)[\'"]/', $content, $matches);

        for ($i = 0; $i < count($matches[0]); $i++) {
            $theme_hooks[] = array(
                'type' => $matches[1][$i],
                'hook' => $matches[2][$i]
            );
        }
    }

    $audit_report = array(
        'theme_supports' => $theme_supports,
        'theme_hooks' => $theme_hooks,
        'timestamp' => current_time('mysql')
    );

    update_option('theme_features_audit', $audit_report);
    return $audit_report;
}
```

REMÉDIATION :

1. Désactivation fonctionnalités non nécessaires :

```
function disable_unnecessary_theme_features() {
    // Désactiver fonctionnalités potentiellement dangereuses
    remove_theme_support('custom-header');
    remove_theme_support('custom-background');

    // Désactiver éditeur de fichiers si pas déjà fait
    if (!defined('DISALLOW_FILE_EDIT')) {
        define('DISALLOW_FILE_EDIT', true);
    }

    // Retirer générateur WordPress du header
    remove_action('wp_head', 'wp_generator');
}
add_action('after_setup_theme', 'disable_unnecessary_theme_features');
```

VALEUR PAR DÉFAUT :

Toutes les fonctionnalités thème activées

4.1.13 *Sécurisation Assets Thème*

MITRE ATT&CK : T1071

DESCRIPTION :

Sécuriser le chargement des assets du thème (CSS, JS, images) pour prévenir les attaques XSS et l'injection de contenu malveillant.

```
# Analyser chargement des assets
grep -r "wp_enqueue_\|wp_register_" wp-content/themes/*/functions.php
# Vérifier intégrité des assets
find wp-content/themes/ -name "*.js" -exec grep -l "eval\|document.write" {} \;
```

REMÉDIATION :

1. Chargement sécurisé des assets :

```
function secure_theme_assets() {
    // Utiliser HTTPS pour tous les assets
    if (is_ssl()) {
        // Forcer HTTPS pour les assets externes
        add_filter('script_loader_src', 'force_ssl_assets');
        add_filter('style_loader_src', 'force_ssl_assets');
    }

    // Ajouter intégrité pour assets externes
    add_filter('script_loader_tag', 'add_integrity_to_scripts', 10, 3);
    add_filter('style_loader_tag', 'add_integrity_to_styles', 10, 4);
}

function force_ssl_assets($src) {
    return str_replace('http://', 'https://', $src);
}

function add_integrity_to_scripts($tag, $handle, $src) {
    // Ajouter SRI pour scripts externes critiques
    $external_scripts = array('jquery', 'bootstrap');

    if (in_array($handle, $external_scripts) && strpos($src, home_url()) === false) {
        // Calculer hash SRI si nécessaire
        $integrity = get_option("script_integrity_$handle");
        if ($integrity) {
            $tag = str_replace('<script ', "<script integrity='$integrity' crossorigin='anonymous' ", $tag);
        }
    }

    return $tag;
}

add_action('wp_enqueue_scripts', 'secure_theme_assets');
```

REMÉDIATION :

1. Validation assets du thème :

```
function validate_theme_assets() {
    $theme_dir = get_stylesheet_directory();
    $suspicious_patterns = array(
        'eval(',
        'document.write(',
        'innerHTML',
        'outerHTML'
    );

    foreach (glob("$theme_dir/*.js") as $js_file) {
        $content = file_get_contents($js_file);

        foreach ($suspicious_patterns as $pattern) {
            if (strpos($content, $pattern) !== false) {
                wp_mail('admin@example.com', 'Asset JavaScript suspect',
                        "Pattern suspect '$pattern' trouvé dans $js_file");
            }
        }
    }
}
```

VALEUR PAR DÉFAUT :

Chargement assets sans validation

4.1.14 *Isolation Thème Sandbox*

MITRE ATT&CK : T1505

DESCRIPTION :

Implémenter des mécanismes d'isolation pour limiter l'impact d'une compromission du thème sur le reste du système.

```
# Vérifier isolation du thème
ls -la wp-content/themes/*/
# Permissions et propriétaire
stat wp-content/themes/*/functions.php
```

REMÉDIATION :

1. Isolation par permissions :

```
# Créer utilisateur dédié pour thèmes
sudo useradd -r -s /bin/false wp-theme-user
sudo chown -R wp-theme-user:wp-theme-user wp-content/themes/
# Permissions restrictives
find wp-content/themes/ -type f -exec chmod 644 {} \;
find wp-content/themes/ -type d -exec chmod 755 {} \;
```

REMÉDIATION :

1. Limitation des capacités thème :

```
function restrict_theme_capabilities() {
    // Désactiver fonctions dangereuses pour les thèmes
    $dangerous_functions = array('exec', 'system', 'shell_exec', 'passthru', 'eval');

    foreach ($dangerous_functions as $function) {
        if (function_exists($function)) {
            // Log tentative d'utilisation
            add_filter("pre_option_active_plugins", function($plugins) use ($function) {
                $backtrace = debug_backtrace();
                foreach ($backtrace as $trace) {
                    if (isset($trace['file']) && strpos($trace['file'], '/themes/') !== false) {
                        error_log("SECURITY: Tentative d'utilisation de $function dans thème: " . $trace['file']);
                        wp_die("Fonction $function interdite dans les thèmes");
                    }
                }
                return $plugins;
            });
        }
    }
}
add_action('after_setup_theme', 'restrict_theme_capabilities');
```

VALEUR PAR DÉFAUT :

Pas d'isolation spécifique

4.1.15 *Documentation et Maintenance Thèmes*

MITRE ATT&CK : T1070

DESCRIPTION :

Maintenir une documentation complète des thèmes, leurs personnalisations et un planning de maintenance régulier.

```
# Documentation existante
find . -name "*theme*doc*" -o -name "*README*" | grep -i theme
# Historique des modifications
ls -la wp-content/themes/*/changelog.txt 2>/dev/null
```

REMÉDIATION :

1. Documentation automatisée :

```
function generate_theme_documentation() {
    $themes = wp_get_themes();
    $documentation = array();

    foreach ($themes as $theme_slug => $theme) {
        $theme_info = array(
            'name' => $theme->get('Name'),
            'version' => $theme->get('Version'),
            'description' => $theme->get('Description'),
            'author' => $theme->get('Author'),
            'template' => $theme->get('Template'),
            'active' => ($theme_slug === get_stylesheet()),
            'parent_theme' => $theme->parent() ? $theme->parent()->get('Name') : null,
            'customizations' => $this->get_theme_customizations($theme_slug),
            'last_modified' => $this->get_theme_last_modified($theme_slug),
            'security_scan_date' => get_option("theme_scan_date_$theme_slug", 'Never')
        );

        $documentation[$theme_slug] = $theme_info;
    }

    // Générer rapport
    file_put_contents(ABSPATH . 'theme-documentation.json',
                      json_encode($documentation, JSON_PRETTY_PRINT));

    return $documentation;
}

function get_theme_customizations($theme_slug) {
    $customizations = array();
    $theme_dir = get_theme_root() . "/$theme_slug";

    // Vérifier fichiers personnalisés
    $custom_files = array('functions.php', 'style.css', 'custom.css');

    foreach ($custom_files as $file) {
        if (file_exists("$theme_dir/$file")) {
            $customizations[$file] = array(
                'size' => filesize("$theme_dir/$file"),
                'modified' => date('Y-m-d H:i:s', filemtime("$theme_dir/$file"))
            );
        }
    }

    return $customizations;
}
```

REMÉDIATION :

1. Planning de maintenance :

```
function schedule_theme_maintenance() {
    // Planifier vérifications mensuelles
    if (!wp_next_scheduled('monthly_theme_maintenance')) {
        wp_schedule_event(time(), 'monthly', 'monthly_theme_maintenance');
    }
}

function monthly_theme_maintenance() {
    // Audit complet des thèmes
    $audit_results = array(
        'vulnerability_scan' => $this->scan_theme_vulnerabilities(),
        'file_integrity' => $this->check_theme_integrity(),
        'performance_impact' => $this->measure_theme_performance(),
        'recommendations' => $this->generate_maintenance_recommendations()
    );

    // Envoyer rapport de maintenance
    wp_mail('admin@example.com', 'Rapport maintenance thèmes mensuel',
            wp_json_encode($audit_results, JSON_PRETTY_PRINT));
}

add_action('init', 'schedule_theme_maintenance');
add_action('monthly_theme_maintenance', 'monthly_theme_maintenance');
```

VALEUR PAR DÉFAUT :

Documentation manuelle et incomplète

5.0 — PERMISSIONS FICHIERS & RÉPERTOIRES

5.1.1 *wp-config.php Permissions Restrictives*

MITRE ATT&CK : T1005

DESCRIPTION :

Le fichier wp-config.php contient des informations critiques (credentials DB, clés de sécurité). Il doit avoir des permissions très restrictives et être protégé contre l'accès web.

```
# Vérifier permissions wp-config.php
ls -la wp-config.php
stat -c "%a %n" wp-config.php
# Test accès web
curl -s https://example.com/wp-config.php | head -5
curl -s https://example.com/wp-config.php.bak | head -5
```

REMÉDIATION :

1. Permissions système appropriées :

```
# Permissions restrictives pour wp-config.php
chmod 600 wp-config.php
chown www-data:www-data wp-config.php
# Vérifier résultat
ls -la wp-config.php
```

REMÉDIATION :

1. Protection web via .htaccess :

```
# .htaccess racine
<Files wp-config.php>
Order allow,deny
Deny from all
</Files>

<Files wp-config.php.bak>
Order allow,deny
Deny from all
</Files>

<Files "wp-config*">
Order allow,deny
Deny from all
</Files>
```

REMÉDIATION :

1. Déplacement hors web root :

```
# Déplacer wp-config.php un niveau au-dessus
mv wp-config.php ../wp-config.php
# WordPress le trouvera automatiquement
```

VALEUR PAR DÉFAUT :

644 (lisible par tous)

5.1.2 *.htaccess Permissions et Protection*

MITRE ATT&CK : T1005

DESCRIPTION :

Le fichier .htaccess contrôle la configuration du serveur web. Il doit être protégé en écriture et contre la modification non autorisée pour maintenir les règles de sécurité.

```
# Vérifier permissions .htaccess
ls -la .htaccess
stat -c "%a %n" .htaccess
# Vérifier contenu
cat .htaccess | head -10
```

REMÉDIATION :

1. Permissions appropriées :

```
# Lecture seule pour .htaccess
chmod 644 .htaccess
chown www-data:www-data .htaccess
# Immutable (optionnel, nécessite droits root)
sudo chattr +i .htaccess
```

REMÉDIATION :

1. Protection dans .htaccess lui-même :

```
# Auto-protection du .htaccess
<Files .htaccess>
Order allow,deny
Deny from all
</Files>

# Protection fichiers de configuration
<FilesMatch "\.(htaccess|htpasswd|ini|log|sh|inc|bak)$">
Order allow,deny
Deny from all
</FilesMatch>
```

REMÉDIATION :

1. Monitoring modifications :

```
function monitor_htaccess_changes() {
    $htaccess_file = ABSPATH . '.htaccess';
    $stored_hash = get_option('htaccess_hash');

    if (file_exists($htaccess_file)) {
        $current_hash = md5_file($htaccess_file);

        if ($stored_hash && $stored_hash !== $current_hash) {
            // .htaccess modifié
            wp_mail('admin@example.com', '.htaccess modifié',
                    'Le fichier .htaccess a été modifié. Vérifiez les changements.');
        }

        update_option('htaccess_hash', $current_hash);
    }
}
wp_schedule_event(time(), 'hourly', 'monitor_htaccess_changes');
```

VALEUR PAR DÉFAUT :

644 (modifiable par le serveur web)

5.1.3 *Répertoire wp-content/uploads Sécurisé*

MITRE ATT&CK : T1105

DESCRIPTION :

Le répertoire uploads est souvent ciblé pour l'upload de webshells. L'exécution PHP doit y être désactivée et les types de fichiers strictement contrôlés.

```
# Tester exécution PHP dans uploads
echo "<?php echo 'PHP executable'; ?>" > wp-content/uploads/test.php
curl -s https://example.com/wp-content/uploads/test.php
rm wp-content/uploads/test.php
# Vérifier .htaccess uploads
ls -la wp-content/uploads/.htaccess
```

REMÉDIATION :

1. Désactiver exécution PHP via .htaccess dans wp-content/uploads/
2. Permissions appropriées : chmod 755 uploads/, fichiers 644
3. Validation types MIME stricte via hooks WordPress

VALEUR PAR DÉFAUT :

Exécution PHP possible

5.1.4 *Désactivation Directory Listing*

MITRE ATT&CK : T1083

DESCRIPTION :

Le directory listing révèle la structure des fichiers aux attaquants. Il doit être désactivé sur tous les répertoires WordPress pour empêcher la reconnaissance.

```
# Tester directory listing sur différents répertoires
curl -s https://example.com/wp-content/ | grep -i "index of"
curl -s https://example.com/wp-includes/ | grep -i "index of"
curl -s https://example.com/wp-content/themes/ | grep -i "index of"
```

REMÉDIATION :

1. Désactivation globale via .htaccess : Options -Indexes
2. Fichiers index.php vides dans répertoires sensibles
3. Configuration serveur Nginx : autoindex off

VALEUR PAR DÉFAUT :

Directory listing souvent activé

5.1.5 *Protection wp-includes*

MITRE ATT&CK : T1005

DESCRIPTION :

Le répertoire wp-includes contient le cœur de WordPress et ne devrait pas être accessible directement via le web, sauf pour les assets nécessaires.

```
# Tester accès direct aux fichiers wp-includes
curl -s https://example.com/wp-includes/version.php
curl -s https://example.com/wp-includes/wp-db.php
# Vérifier assets légitimes
curl -s -I https://example.com/wp-includes/css/admin-bar.min.css
```

REMÉDIATION :

1. Protection sélective wp-includes via .htaccess
2. Bloquer .php, autoriser CSS/JS/images
3. Hooks PHP pour bloquer accès direct

VALEUR PAR DÉFAUT :

Accès direct possible

6.0 — BASE DE DONNÉES

6.1.1 *Préfixe Table Personnalisé*

MITRE ATT&CK : T1190

DESCRIPTION :

Le préfixe par défaut wp_ facilite les attaques d'injection SQL automatisées. Un préfixe personnalisé ajoute une couche d'obscurité contre les attaques génériques.

```
# Via wp-config.php
grep "table_prefix" wp-config.php
# Via wp-cli
wp config get table_prefix
# Test base de données
wp db query "SHOW TABLES LIKE 'wp_%'"
```

REMÉDIATION :

1. Modifier wp-config.php : table_prefix = 'xyz123_';
2. Renommer tables existantes si nécessaire
3. Mettre à jour options et usermeta

VALEUR PAR DÉFAUT :

wp_

6.1.2 *Utilisateur Base de Données Dédié*

MITRE ATT&CK : T1078

DESCRIPTION :

WordPress doit utiliser un utilisateur de base de données dédié avec des privilèges minimaux, pas le compte root ou admin de la base.

```
# Vérifier utilisateur DB dans wp-config
grep "DB_USER" wp-config.php
# Tester privilèges
wp db query "SHOW GRANTS FOR CURRENT_USER"
# Vérifier si root
wp db query "SELECT USER(), CURRENT_USER()"
```

REMÉDIATION :

1. Créer utilisateur dédié WordPress
2. Accorder privilèges minimaux uniquement
3. Révoquer privilèges administratifs si existants

VALEUR PAR DÉFAUT :

Souvent utilisateur avec trop de privilèges

6.1.3 *Privilèges Minimaux Base de Données*

MITRE ATT&CK : T1078

DESCRIPTION :

L'utilisateur WordPress doit avoir uniquement les privilèges nécessaires : SELECT, INSERT, UPDATE, DELETE sur la base WordPress. Pas de privilèges globaux.

```
# Vérifier privilèges actuels
wp db query "SHOW GRANTS"
# Vérifier privilèges dangereux
wp db query "SELECT * FROM information_schema.user_privileges WHERE grantee LIKE '%wpuser%'"
```

REMÉDIATION :

1. Révoquer privilèges excessifs
2. Accorder seulement SELECT, INSERT, UPDATE, DELETE
3. Limiter à la base WordPress uniquement

VALEUR PAR DÉFAUT :

Privilèges souvent excessifs

6.1.4 *Connexions Chiffrées SSL/TLS*

MITRE ATT&CK : T1040

DESCRIPTION :

Les connexions à la base de données doivent être chiffrées via SSL/TLS pour protéger les données en transit, particulièrement sur des bases distantes.

```
# Vérifier SSL DB actuel
wp db query "SHOW STATUS LIKE 'Ssl_cipher'"
wp db query "SHOW VARIABLES LIKE 'have_ssl'"
# Configuration wp-config
grep -i ssl wp-config.php
```

REMÉDIATION :

1. Configurer MySQL/MariaDB avec SSL
2. Ajouter MYSQL_CLIENT_FLAGS dans wp-config.php
3. Forcer connexions SSL uniquement

VALEUR PAR DÉFAUT :

Connexions non chiffrées

6.1.5 *Sauvegarde Automatisée Base de Données*

MITRE ATT&CK : T1485

DESCRIPTION :

Mettre en place des sauvegardes automatisées, régulières et testées de la base de données WordPress avec rétention appropriée et stockage sécurisé.

```
# Vérifier plugins backup
wp plugin list | grep -iE "(backup|updraft)"
# Dernière sauvegarde
ls -la /backups/wordpress/ 2>/dev/null
# Cron backup
crontab -l | grep -i backup
```

REMÉDIATION :

1. Configuration plugin backup fiable
2. Sauvegarde quotidienne minimum
3. Test restauration régulier

VALEUR PAR DÉFAUT :

Sauvegardes manuelles ou absentes

Annexe : Checklist (95 controles)

#	Recommandation	Niveau	Oui	Non	N/A
Section 1 — INSTALLATION & CONFIGURATION DE BASE					
1.1.1	Version WordPress Actuelle et Supportée	●	☐	☐	☐
1.1.2	Version PHP Supportée et Sécurisée	●	☐	☐	☐
1.1.3	Configuration wp-config.php Sécurisée	●	☐	☐	☐
1.1.4	Clés de Sécurité SALT Configurées	●	☐	☐	☐
1.1.5	Préfixe Table Base de Données Personnalisé	●	☐	☐	☐
1.1.6	Mode Debug Désactivé en Production	●	☐	☐	☐
1.1.7	Édition de Fichiers Désactivée	●	☐	☐	☐
1.1.8	Installation de Plugins/Thèmes Contrôlée	●	☐	☐	☐
1.1.9	Révisions de Posts Limitées	●	☐	☐	☐
1.1.10	Corbeille Automatique Configurée	●	☐	☐	☐
1.1.11	Version MySQL/MariaDB Supportée	●	☐	☐	☐
1.1.12	Constantes de Sécurité Avancées	●	☐	☐	☐
1.1.13	Répertoire wp-content Personnalisé	●	☐	☐	☐
1.1.14	URL d'Administration Personnalisée	●	☐	☐	☐
1.1.15	Suppression Fichiers par Défaut	●	☐	☐	☐
1.1.16	Configuration Memory Limit Appropriée	●	☐	☐	☐
1.1.17	Configuration Timezone Sécurisée	●	☐	☐	☐
1.1.18	Limitation Taille Upload Sécurisée	●	☐	☐	☐
1.1.19	Configuration Cron Sécurisée	●	☐	☐	☐
1.1.20	Configuration Error Reporting Sécurisée	●	☐	☐	☐
Section 2 — AUTHENTIFICATION & GESTION DES UTILISATEURS					
2.1.1	Politique de Mots de Passe Forte	●	☐	☐	☐
2.1.2	Authentification Multi-Facteurs (2FA/MFA)	●	☐	☐	☐
2.1.3	Limitation Tentatives de Connexion	●	☐	☐	☐
2.1.4	Protection reCAPTCHA	●	☐	☐	☐
2.1.5	Gestion Roles Utilisateurs Restrictive	●	☐	☐	☐
2.1.6	Prévention Énumération Utilisateurs	●	☐	☐	☐
2.1.7	Désactivation XML-RPC	●	☐	☐	☐
2.1.8	Sécurisation wp-login.php	●	☐	☐	☐
2.1.9	Audit Sessions Utilisateurs	●	☐	☐	☐
2.1.10	Protection Comptes Inactifs	●	☐	☐	☐
2.1.11	Sécurisation Mot de Passe Administrateur	●	☐	☐	☐
2.1.12	Configuration Cookies Sécurisés	●	☐	☐	☐
2.1.13	Restriction Accès par IP	●	☐	☐	☐
2.1.14	Audit Trail Authentifications	●	☐	☐	☐
2.1.15	Protection Contre Attaques Timing	●	☐	☐	☐
2.1.16	Gestion Expiration Mots de Passe	●	☐	☐	☐
2.1.17	Protection Reset Mot de Passe	●	☐	☐	☐
2.1.18	Authentification Applications Tierces	●	☐	☐	☐
2.1.19	Séparation Comptes Fonctionnels	●	☐	☐	☐
2.1.20	Monitoring Escalade Privilèges	●	☐	☐	☐
2.1.21	Protection Contre User Enumeration API	●	☐	☐	☐
2.1.22	Gestion Sessions Concurrentes	●	☐	☐	☐
2.1.23	Protection Mot de Passe en Transit	●	☐	☐	☐
2.1.24	Authentification Forte pour API	●	☐	☐	☐
2.1.25	Politique Verrouillage Compte	●	☐	☐	☐
Section 3 — PLUGINS & EXTENSIONS					
3.1.1	Inventaire Complet des Plugins	●	☐	☐	☐
3.1.2	Plugins Obsolètes et Abandonnés	●	☐	☐	☐
3.1.3	Vulnérabilités Connues (Base WPScan)	●	☐	☐	☐
3.1.4	Permissions et Capacités des Plugins	●	☐	☐	☐
3.1.5	Plugins Nulled et Piratés	●	☐	☐	☐
3.1.6	Auto-Update des Plugins Critiques	●	☐	☐	☐
3.1.7	Réduction Surface d'Attaque	●	☐	☐	☐

#	Recommandation	Niveau	Oui	Non	N/A
3.1.8	Validation Code Source Plugins	●	☐	☐	☐
3.1.9	Gestion Dépendances des Plugins	●	☐	☐	☐
3.1.10	Sandbox et Tests Plugins	●	☐	☐	☐
3.1.11	Monitoring Activité Plugins	●	☐	☐	☐
3.1.12	Politique Installation Plugins	●	☐	☐	☐
3.1.13	Backup avant Modifications Plugins	●	☐	☐	☐
3.1.14	Contrôle Versions Plugins	●	☐	☐	☐
3.1.15	Séparation Plugins par Environnement	●	☐	☐	☐
3.1.16	Audit Permissions Système Plugins	●	☐	☐	☐
3.1.17	Protection Fichiers Configuration Plugins	●	☐	☐	☐
3.1.18	Isolation Plugins Critiques	●	☐	☐	☐
3.1.19	Gestion Licences et Mises à Jour Premium	●	☐	☐	☐
3.1.20	Tests Sécurité Automatisés	●	☐	☐	☐
3.1.21	Chiffrement Données Plugins	●	☐	☐	☐
3.1.22	Contrôle Accès Réseau Plugins	●	☐	☐	☐
3.1.23	Validation Entrées Plugins	●	☐	☐	☐
3.1.24	Gestion Erreurs et Exceptions Plugins	●	☐	☐	☐
3.1.25	Documentation et Traçabilité Plugins	●	☐	☐	☐
Section 4 — THÈMES					
4.1.1	Thème Actif Sécurisé et À Jour	●	☐	☐	☐
4.1.2	Suppression Thèmes Inactifs	●	☐	☐	☐
4.1.3	Child Theme Obligatoire	●	☐	☐	☐
4.1.4	Détection Thèmes Nulled/Piratés	●	☐	☐	☐
4.1.5	Désactivation Éditeur de Thème	●	☐	☐	☐
4.1.6	Audit Vulnérabilités Thèmes	●	☐	☐	☐
4.1.7	Validation Code Personnalisé Thèmes	●	☐	☐	☐
4.1.8	Protection Fichiers Thème Sensibles	●	☐	☐	☐
4.1.9	Gestion Versions et Rollback Thèmes	●	☐	☐	☐
4.1.10	Monitoring Modifications Thèmes	●	☐	☐	☐
4.1.11	Validation Headers et Métadonnées Thèmes	●	☐	☐	☐
4.1.12	Contrôle Fonctionnalités Thème	●	☐	☐	☐
4.1.13	Sécurisation Assets Thème	●	☐	☐	☐
4.1.14	Isolation Thème Sandbox	●	☐	☐	☐
4.1.15	Documentation et Maintenance Thèmes	●	☐	☐	☐
Section 5 — PERMISSIONS FICHIERS & RÉPERTOIRES					
5.1.1	wp-config.php Permissions Restrictives	●	☐	☐	☐
5.1.2	.htaccess Permissions et Protection	●	☐	☐	☐
5.1.3	Répertoire wp-content/uploads Sécurisé	●	☐	☐	☐
5.1.4	Désactivation Directory Listing	●	☐	☐	☐
5.1.5	Protection wp-includes	●	☐	☐	☐
Section 6 — BASE DE DONNÉES					
6.1.1	Préfixe Table Personnalisé	●	☐	☐	☐
6.1.2	Utilisateur Base de Données Dédié	●	☐	☐	☐
6.1.3	Privilèges Minimaux Base de Données	●	☐	☐	☐
6.1.4	Connexions Chiffrées SSL/TLS	●	☐	☐	☐
6.1.5	Sauvegarde Automatisée Base de Données	●	☐	☐	☐