RAPPORT DE VULNÉRABILITÉ DE TESTS DE PÉNÉTRATION

Groupe F
Damilot Julien
Fétu Thimoté
Hulot Raffaele
Taillet Alexandre

Table des Matières

1. Introduction

1.1. Contexte de la mission

Dans le cadre de notre formation au sein de l'équipe de pentesting **Rogue Sentinels**, nous avons été chargés de réaliser une série de challenges sous forme de CTF (Capture The Flag). L'objectif principal était d'identifier et d'exploiter des vulnérabilités sur une infrastructure dockerisée afin de capturer 5 flags distincts. Par souci de simplicité en cas de démonstration en temps réel, un script d'exploitation automatisée a été implémenter.

1.2. Périmètre et environnement de test

L'infrastructure de test a été déployée en local via un outil fourni par l'équipe, disponible sur GitHub.

Mise en place de l'environnement : implementation.sh

Cette commande a permis de déployer plusieurs conteneurs Docker constituant l'infrastructure cible du pentest.

1.3. Méthodologie de reconnaissance

1.3.1. Point de départ

Le seul élément d'information fourni était le nom de domaine principal : **`rogue-sentinels.io`**

1.3.2. Énumération des sous-domaines

Pour découvrir les différentes applications et services exposés, nous avons procédé à une énumération DNS à l'aide de l'outil **Gobuster**.

Paramètres utilisés :

- `--domain` : domaine cible
- `-w` : wordlist de sous-domaines communs
- `-t 50` : 50 threads pour accélérer le scan

Résultat : gobuster.png

Cette énumération a révélé les sous-domaines suivants :

- `dns.rogue-sentinels.io`
- `fileserver.rogue-sentinels.io`
- `restricted.rogue-sentinels.io`
- `staging.rogue-sentinels.io`
- `travel.rogue-sentinels.io`
- `webservices.rogue-sentinels.io`

1.3.3. Scan de ports avec Nmap

Après avoir tenté d'accéder aux sous-domaines via navigateur et commande `ping` sans succès, nous avons compris que ces services fonctionnaient sur des ports non-standard. Un scan complet des ports a donc été effectué avec nmap.sh.

Paramètres utilisés :

- `-sS` : SYN scan (scan furtif)
- `-p-` : scan de tous les ports (1-65535)
- `-T4` : timing agressif pour accélérer le scan

Résultat : nmap-domain.png

Ce scan nous a permis d'identifier les ports ouverts pour chaque sous-domaine et de cibler nos attaques sur les services actifs.

1.4. Structure de ce rapport

Le présent rapport détaille l'exploitation de chacune des 5 vulnérabilités découvertes. Pour chaque vulnérabilité, nous présenterons :

1. **Description de la vulnérabilité** : nature et fonctionnement de la faille
2. **Découverte et reconnaissance** : comment nous l'avons identifiée
3. **Exploitation** : étapes détaillées de l'attaque
4. **Impact** : conséquences potentielles pour l'entreprise
5. **Recommandations** : mesures correctives à mettre en place

2. Vulnérabilité #1 : Accès non autorisé au serveur de fichiers

Sous-domaine cible : `fileserver.rogue-sentinels.io`

Type de vulnérabilité :

- `CWE-200 Exposure of Sensitive Information to an Unauthorized Actor : .htaccess peut être visualisé sans restriction.`
- `CWE-307 Improper Restriction of Excessive Authentication Attempts : aucune limite du nombre de tentatives de connections.`
- `CWE-521 Weak Password Policy : les mots de passes ne sont pas assez robustes et sont facilement déchiffrable.`
- `CWE-256 Plaintext storage of passwords : les mots de passes sont enregistrés directement tel quel sans être chiffré.`

Criticité : CRITIQUE (CVSS 9.1)

2.1. Contexte de Découverte

Suite au scan Nmap effectué lors de la phase de reconnaissance, le sous-domaine `fileserver.rogue-sentinels.io` a été identifié avec un port ouvert exposant une application web de gestion de fichiers. Cette application nécessitait une authentification mais présentait plusieurs failles de sécurité permettant un accès complet au système de fichiers et à des données sensibles.

2.2. Reproduction des Observations

2.2.1. Énumération des répertoires avec Dirbuster

Après identification du port ouvert, une énumération des répertoires et fichiers accessibles a été réalisée avec **Dirbuster**.

Paramètres utilisés :

- `-u` : URL cible
- `-l` : wordlist `common_urls.txt`
- `Use Blank Expression cocher une fois dirbuster ouvert`

Résultat : dirbuster-result.png

Fichiers et répertoires découverts :

- `/` - Racine du site (page d'accueil)
- `/index.html` - Page d'accueil principale
- `/.htaccess` - Fichier de configuration Apache
- `/secure` - Répertoire protégé (probablement interface d'administration)

2.2.2. Découverte d'informations sensibles dans .htaccess

Contenu exposé :

```
# Contact : security-admin@web2000-corp.com
```

Information critique obtenue : Adresse email de l'administrateur → `security-admin@web2000-corp.com`

2.2.3. Envoie d'une requête via le formulaire pour accéder à l'URL de l'API

2.2.4. Bruteforce de l'authentification

Commande Shell pour trouver le mot de passe : bruteforce-password.sh

Paramètres :

- Email cible : `security-admin@web2000-corp.com (découvert dans .htaccess)`
- Wordlist : mots de passes probables choisis au hasard

Résultat : password-bruteforce.png

Credentials valides identifiés :

- Email : `security-admin@web2000-corp.com`
- Mot de passe : `password`

2.2.5. Accès à l'interface File System Browser

Après authentification réussie, accès à l'interface de gestion de fichiers sur la page d'accueil.

L'interface révèle un champ **"Folder Name"** permettant de naviguer dans l'arborescence du serveur.

2.2.6. Énumération des fichiers de backup via bruteforce

Second script de bruteforce pour identifier les archives présentes et les déchiffrées : bruteforce-search.sh.

Découvertes :

- Dossier « `backups` »
- Fichier `_BACKUP_STRATEGY.txt` détaillant la politique de chiffrement
- Multiples archives : `backup_2001_124.zip` à `backup_2001_350.zip`

2.2.7. Analyse de la stratégie de backup

Contenu du fichier `_BACKUP_STRATEGY.txt` :

Vulnérabilité identifiée : Politique de mot de passe faible, PIN 4 chiffres de 1000 à 9999 = 9000 combinaisons possibles.

2.2.8. Bruteforce à 4 chiffres

Avec un script de bruteforce l'on peut tenter toutes les combinaisons de 1000 à 9999 sur tous les fichiers zip pour trouver quels fichiers sont encore sur le système de PIN et le PIN qui les ouvrent.

Résultat : backup_2001_322.zip | PIN : 4220

2.2.9. Extraction du flag

Une fois l'archive déchiffrée, nous obtenons un fichier backup.sql. Après l'ouverture de celui-ci, le flag est immédiatement visible.

Contenu de l'archive déchiffrée : backup_2001_322.zip

FLAG #1 CAPTURÉ : `FLAG_Web2000CorpBackups`

2.3. Informations Techniques Collectées

Application :

- Serveur web : Apache 2.4.x
- Backend : Node.js / Express
- Authentification : JWT (mal implémentée)

Vulnérabilités identifiées :

1. Exposition du fichier `.htaccess` (information disclosure)
2. Absence de rate limiting sur l'endpoint d'authentification
3. Politique de mots de passe faible (pas de complexité requise)
4. Mots de passe stockés en clair dans les backups SQL
5. Chiffrement ZIP avec PIN 4 chiffres (espace de clés trop réduit)
6. Accès non restreint aux archives de sauvegarde

2.4. Causes Techniques du Problème

Pour les développeurs :

1. **Fichiers de configuration exposés** :
    - Le fichier `.htaccess` est accessible publiquement, divulguant des métadonnées sensibles (adresse email admin)

2. **Absence de rate limiting** :
    - L'endpoint `/api/v1/auth` ne limite pas le nombre de tentatives de connexion, permettant des attaques par bruteforce automatisées
3. **Politique de mots de passe insuffisante** :
    - Aucune exigence de longueur minimale
    - Pas de complexité imposée (majuscules, caractères spéciaux)
    - Mots de passe couramment compromis non blacklistés
4. **Stockage non sécurisé des credentials** :
    - Mots de passe en clair dans les fichiers SQL de backup
    - Aucun hashage (bcrypt, argon2) appliqué
5. **Chiffrement faible des archives** :
    - Utilisation de ZIP avec mot de passe court (4 chiffres)
    - Espace de clés de seulement 9 000 combinaisons
    - Vulnérable aux attaques par force brute en quelques secondes
6. **Contrôle d'accès insuffisant** :
    - Le File System Browser permet de naviguer dans des répertoires sensibles
    - Aucune vérification des permissions utilisateur

2.5. Impact pour l'Entreprise

Risques immédiats :

1. **Compromission totale des comptes** :
    - L'accès aux backups contenant les identifiants de tous les utilisateurs permet une prise de contrôle complète du système
2. **Fuite de données sensibles** :
    - Informations personnelles des utilisateurs exposées
    - Données métiers potentiellement confidentielles dans les backups
3. **Réputation** :
    - Une fuite de ce type pourrait gravement nuire à l'image de l'entreprise auprès des clients et partenaires
4. **Financier** :
    - Amendes RGPD : jusqu'à 4% du chiffre d'affaires annuel ou 20 millions d'euros (la valeur la plus haute)
    - Coûts de remédiation et d'audit de sécurité
    - Pertes liées à l'arrêt de service pendant la correction
5. **Légal** :
    - Non-conformité aux standards de sécurité (ISO 27001, PCI-DSS si traitement de paiements)
    - **Violation RGPD** : Stockage non sécurisé de données personnelles

2.6. Recommandations et Remédiations

Mesures immédiates (à appliquer sous 48h) :

1. **Révoquer tous les accès compromis**
    - Forcer la réinitialisation des mots de passe de tous les comptes
    - Invalider toutes les sessions actives
    - Auditer les logs d'accès pour identifier les connexions suspectes

2. **Bloquer l'accès aux fichiers sensibles grâce à apache**

3. **Implémenter un rate limiting strict**

Mesures à moyen terme (1-2 semaines) :

4. **Renforcer la politique de mots de passe (en BACK-END)**

5. **Hasher tous les mots de passe**
6. **Implémenter une authentification multi-facteurs (MFA)**
    - Utiliser TOTP (Google Authenticator, Authy)
    - Envoyer des codes par SMS/email
    - Rendre obligatoire pour les comptes administrateurs
7. **Sécuriser les backups**

8. **Restreindre l'accès au File System Browser**
    - Implémenter un système de permissions granulaires
    - Limiter les répertoires accessibles (chroot / sandboxing)
    - Logger tous les accès aux fichiers sensibles

Mesures à long terme (1-3 mois) :

9. **Audit de sécurité complet**
    - Pentesting par un cabinet externe
    - Scan de vulnérabilités automatisé (OWASP ZAP, Burp Suite)
    - Code review focalisé sur la sécurité
10. **Mise en place d'un WAF (Web Application Firewall)**
    - Protection contre le bruteforce
    - Blocage automatique des IP malveillantes
    - Détection d'anomalies
11. **Monitoring et alerting**
    - SIEM pour centraliser les logs
    - Alertes en temps réel sur les tentatives de bruteforce
    - Dashboard de sécurité avec métriques clés
12. **Formation des développeurs**
    - OWASP Top 10
    - Secure coding practices
    - Sensibilisation aux attaques courantes

2.7. Références

- OWASP - Authentication Cheat Sheet
- NIST - Digital Identity Guidelines
- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-521: Weak Password Requirements

3. Vulnérabilité #2 : Upload de fichier malveillant et exécution de code arbitraire

Sous-domaine cible : travel.rogue-sentinels.io
Type de vulnérabilité :

- `CWE-434 Unrestricted File Upload with dangerous type : n'importe quel fichier peut être envoyé tant qu'il a l'extension .png ou .jpg dans son nom.`
- `CWE-94 Improper Control of Generation of Code ('Code Injection') : le code php du fichier envoyé est exécuté.`
- `CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection') : la possibilité d'exécution de commande au niveau de l'OS de la machine cible.`

Criticité : CRITIQUE (CVSS 9.8)

3.1. Contexte de Découverte

Suite à l'énumération DNS initiale, le sous-domaine travel.rogue-sentinels.io a été identifié. Après un scan `nmap` pour déterminer le port exposé, une application web de gestion de voyages a été découverte. Cette application proposait une fonctionnalité d'upload d'images pour les réservations, qui s'est révélée vulnérable à l'injection de code malveillant.

3.2. Reproduction des Observations

3.2.1. Reconnaissance de l'application web

Accès à l'application via navigateur : `http://travel.rogue-sentinels.io:80/`

Observations :

- Site web nommé "The Happy Salmon"
- Page d'accueil : index.php
- Fonctionnalité principale : système de réservation avec upload d'images
- Formulaire d'upload visible avec restrictions apparentes

3.2.2. Analyse du formulaire d'upload

Inspection du code HTML du formulaire : upload.html

Restrictions identifiées :

- Attribut HTML `accept="image/*"` (côté client uniquement)
- Validation supposée côté serveur

Tests initiaux : wrong_shell.sh

Résultat : wrong_shell.html (Only files containing .jpg or .png are allowed.)

Conclusion : Le serveur effectue une validation basique du type de fichier.

3.2.3. Création d'un shell PHP déguisé en image

Pour contourner la validation, création d'un fichier hybride combinant :

1. **Code PHP** malveillant
2. **Double extension** : .png.php

Explication technique :

- Le code PHP est ignoré par les parsers d'image mais exécuté par l'interpréteur PHP
- La double extension .png.php force l'exécution en tant que PHP si le serveur traite .php

3.2.4. Upload du shell déguisé

Tentative d'upload du fichier hybride : shell.png.php

Résultat : good_shell.html (File successfully uploaded)

Analyse :

- La validation côté client vérifie **uniquement** que l'extension contienne un format d'image et le coté serveur n'effectue aucune vérification
- Elle ne vérifie **pas** le contenu réel ni la double extension
- Le fichier est accepté et stocké avec son nom d'origine

3.2.5. Localisation du fichier uploadé

Recherche du répertoire de stockage des fichiers uploadés :

- Envoie d'une image quelconque
- ***F12 -> NETWORK (après avoir uploadé l'image)***
- Répertoire d'upload identifié : `/salmon-uploads/`
- URL complète du shell : http://travel.rogue-sentinels.io:80/salmon-uploads/shell.png.php

3.2.6. Vérification de l'exécution du code

Test initial avec une commande simple : whoami.sh

Résultat : whoami.png

Confirmation : Le code PHP s'exécute avec les privilèges de l'utilisateur du serveur web (www-data).

Identifier le répertoire courant : pwd.sh

Résultat : pwd.png

3.2.7. Exploration du système de fichiers

Énumération du répertoire web racine : list.sh

Résultat : list.png

3.2.8. Lecture systématique des fichiers PHP

Extraction du contenu de chaque fichier pour trouver le flag : grep.sh

Résultat : grep.png

3.2.9. Découverte du flag dans reservations.php

FLAG #2 CAPTURÉ (Ligne 6) : `FLAG_SalmonDatabaseCredentials`

Information bonus récupérée :

- Host de base de données : localhost
- Utilisateur DB : salmon_admin
- Nom de la base : salmon_db

Étape BONUS : Remote Shell Access

Possibilité de rentrer en Remote Shell sur le site pour exécuter directement les commandes afin de trouver le flag, pour cela nous écoutons sur un port de notre machine via `nc -lvp 1339`.

La commande curl utilise le paquet PHP du serveur web et ouvre une lecture vers la machine attaquante redirigeant vers le `/bin/sh` de la machine attaquée.

Cela permet d'être dans une session `/bin/sh` dédié, donc pouvoir exécuter plus facilement des commandes à la chaine, avoir une meilleure visualisions des outputs de la machine. Cela peut aider et faciliter grandement à l'exploration des fichiers et des commandes de la machine pour les attaquants.

3.3. Informations Techniques Collectées

Stack technique :

- Serveur web : Apache/Nginx
- Backend : PHP 7.x / 8.x
- Base de données : MySQL/MariaDB
- Système d'exploitation : Linux (Ubuntu/Debian)

Vulnérabilités identifiées :

1. **Validation insuffisante des fichiers uploadés** (CWE-434)
    - Aucun renommage systématique des fichiers uploadés (nom aléatoire)

- Aucune validation de l'extension réelle

2. **Exécution de code arbitraire** (CWE-94)
    - PHP exécutable dans le répertoire d'upload
    - Aucune restriction .htaccess ou configuration serveur
3. **Exposition de credentials sensibles** (CWE-798)
    - Mot de passe de base de données hardcodé dans le code source
    - Accessible via RCE
4. **Permissions trop permissives** (CWE-732)
    - Répertoire d'upload accessible en lecture/écriture
    - Fichiers PHP exécutables publiquement

3.4. Causes Techniques du Problème

Pour les développeurs :

1. **Validation de fichier inadéquate** :

   Problème : Le formulaire accepte tout type de fichiers du moment que ceux-ci contiennent au moins une extension d'image même si celui-ci a une double extension ou si le contenu des fichiers n'est pas en accord avec l'extension.

2. **Absence de validation d'extension** :

   Problème : Le fichier `shell.jpg.php` est accepté car aucune whitelist d'extensions n'est appliquée. C'est-à-dire que le système ne décèle pas l'utilisation d'une seconde extension. Alors qu'il devrait refuser le fichier.

3. **Configuration serveur permissive** :

    1. **ABSENCE DE PROTECTION** dans `/salmon-uploads/`
    2. PHP est exécutable par défaut

   Problème : Le serveur Apache/Nginx exécute les fichiers `.php` même dans les répertoires d'upload.

4. **Credentials hardcodés** :

   Problème : Les identifiants de base de données sont stockés directement dans le code source, accessible via RCE.

5. **Absence de sanitisation du nom de fichier** :

- Accepte `shell.jpg.php` sans modification

3.5. Impact pour l'Entreprise

Risques immédiats :

1. **Compromission totale du serveur** :
    - Exécution de commandes arbitraires avec les privilèges www-data
    - Potentiel d'escalade de privilèges vers root
    - Installation de backdoors permanentes
2. **Accès complet à la base de données** :
    - Credentials de DB exposés → accès total aux données clients
    - Informations de réservation, paiements, données personnelles compromises
    - Possibilité de modifier/supprimer des données
3. **Exfiltration de données sensibles** :
4. **Violation RGPD** :
    - Fuite de données personnelles (noms, emails, adresses, cartes bancaires)
    - Notification obligatoire à la CNIL sous 72h
    - **Amendes potentielles** : jusqu'à 4% du CA annuel ou 20M€
5. **Risque de ransomware** :
    - Un attaquant pourrait chiffrer tous les fichiers du serveur
    - Demander une rançon pour restituer l'accès
    - Paralysie complète de l'activité
6. **Réputation et business** :
    - Perte de confiance des clients
    - Impact sur les réservations futures
    - Coûts de remédiation et d'audit
    - Potentiels contentieux clients
7. **Conformité PCI-DSS** :
    - Si traitement de cartes bancaires : non-conformité totale
    - Risque de suspension du traitement des paiements

3.6. Recommandations et Remédiations

Mesures immédiates (à appliquer sous 24h) :

1. **Supprimer tous les fichiers uploadés suspects**
2. **Désactiver immédiatement l'exécution PHP dans le répertoire d'upload**
   - Créer `/var/www/html/salmon-uploads/.htaccess`
3. **Révoquer et changer IMMÉDIATEMENT les credentials DB**
4. **Déplacer les credentials hors du code source**
5. **Auditer les logs pour identifier les accès malveillants**

Mesures à moyen terme (1-2 semaines) :

6. **Implémenter une validation stricte des fichiers uploadés**
7. **Renommer les fichiers uploadés avec des noms aléatoires**
8. **Stocker les fichiers hors de la racine web**
9. **Utiliser une bibliothèque de gestion de fichiers sécurisée**

Mesures à long terme (1-3 mois) :

10. **Implémenter un système de scanning antivirus**
11. **Content Security Policy (CSP)**
12. **Web Application Firewall (WAF)**
    - Déployer ModSecurity avec règles OWASP CRS
    - Bloquer automatiquement les tentatives d'upload de fichiers suspects
    - Alertes en temps réel sur les comportements malveillants
13. **Monitoring et alerting**
14. **Audit de sécurité régulier**
    - Pentest trimestriel
    - Scan de vulnérabilités automatisé hebdomadaire
    - Code review focalisé sur les uploads de fichiers
15. **Formation de l'équipe de développement**
    - OWASP Top 10 (A04:2021 - Insecure Design)
    - Secure file upload best practices
    - Threat modeling pour les fonctionnalités critiques

16. **Chiffrement des credentials**

3.7. Chaîne d'Exploitation Complète (Résumé)

1. **Reconnaissance**
   - Nmap scan → Identification du port
2. **Analyse de l'application**
   - Découverte du formulaire d'upload
3. **Test de validation**
   - Identification de la validation faible (extension uniquement)
4. **Création du payload**
   - `shell.png.php` avec code PHP
5. **Upload du shell**
   - Contournement de la validation
6. **Localisation du fichier**
   - `/salmon-uploads/shell.png.php`
7. **Exécution de code à distance (RCE)**
   - `?cmd=whoami` → `www-data`
8. **Énumération du système**
   - Listing des fichiers PHP
9. **Exfiltration de données**
   - Lecture de reservations.php
10. **Capture du flag**
    - `FLAG_SalmonDatabaseCredentials`

3.8. Références

- OWASP - File Upload Cheat Sheet
- CWE-434: Unrestricted Upload of File with Dangerous Type
- CWE-94: Improper Control of Generation of Code (Code Injection)
- OWASP Top 10 2021 - A04: Insecure Design
- PHP Security - File Uploads

4. Vulnérabilité #3 : Déni de service par Fuzzing et exposition d'informations sensibles

Sous-domaine cible : `restricted.rogue-sentinels.io:64375`
Type de vulnérabilité :

- `Denial of Service (DoS)`
- `Information Disclosure`
- `Improper Error Handling`

Criticité : ÉLEVÉE (CVSS 7.5)

4.1. Contexte de Découverte

Le sous-domaine `restricted.rogue-sentinels.io` hébergeait une application web sur le port `64375`. Cette application proposait un simple calculateur arithmétique permettant d'additionner trois valeurs. L'analyse de cette fonctionnalité apparemment anodine a révélé une vulnérabilité critique de déni de service déclenchée par une valeur spécifique, ainsi qu'une fuite d'informations dans les pages d'erreur.

4.2. Reproduction des Observations

4.2.1. Reconnaissance de l'application

Accès à l'application via navigateur : `http://restricted.rogue-sentinels.io:64375`

Observations :

- Page d'accueil avec titre "Ultimate adder"
- Formulaire HTML avec trois champs numériques
- Fonction : addition de trois nombres (x + y + z)
- Interface minimaliste sans fonctionnalités apparentes supplémentaires
- Route `/calculate` permettant de renvoyer le résultat

Inspection du code de la page d'accueil

Contraintes identifiées :

- Trois paramètres numériques : `x`, `y`, `z`
- Valeurs acceptées : 0 à 999,999
- Validation côté client avec attributs `min`/`max`

Tests initiaux :

- Requête avec valeurs normales
- Requête avec valeurs maximales

4.2.2. Fuzzing systématique - Recherche de valeurs critiques

Face à l'absence de vulnérabilités évidentes, une approche de **fuzzing** a été adoptée pour identifier des comportements anormaux liés à des valeurs spécifiques.

Hypothèse : Certaines valeurs ou combinaisons de valeurs pourraient déclencher des bugs, crashs ou comportements inattendus.

Script **de fuzzing Python**

Exécution **du script**

Résultat critique détecté : La valeur 42069 provoque un crash du service.

4.2.3. Reproduction manuelle du crash

Résultat observé dans le navigateur

État du service : L'application devient totalement non-responsive. Toutes les tentatives d'accès ultérieures retournent la même erreur.

Vérification **du crash**

4.2.4. Analyse via requête cURL - Traversée de répertoires

Après le crash, une exploration avec cURL a été effectuée pour identifier des informations supplémentaires.

Informations critiques exposées :

1. **Chemin du fichier d'erreur :** `/usr/share/nginx/html/custom_50x.html`
   - Révèle que le serveur utilise **Nginx**
   - Expose la structure des répertoires
2. **Message de débogage dans le fichier** custom_50x.html **ainsi que dans la** console**.**

Chaîne suspecte identifiée : `JHtGTEFHX0Z1enp5RE9TfQ==`

4.2.5. Décodage de la chaîne Base64

La chaîne se termine par ==, ce qui suggère un encodage **Base64**.

FLAG #3 CAPTURÉ : `FLAG_FuzzyDOS`

4.3. Informations Techniques Collectées

Stack technique :

- Serveur web : **Nginx** (révélé par le chemin `/usr/share/nginx/html/`)
- Backend : Inconnu (Node.js, Python, Go probable)

- Pages d'erreur personnalisées : `custom_50x.html`

Vulnérabilités identifiées :

1. **Denial of Service (DoS) - CWE-400**
   - Valeur spécifique `42069` provoque un crash du service
   - Aucune validation ou sanitisation des entrées "dangereuses"
   - Absence de mécanisme de récupération automatique
2. **Information Disclosure - CWE-209**
   - Messages d'erreur verbeux exposant des informations techniques
   - Chemin absolu du fichier d'erreur révélé
   - Messages de débogage présents en production
3. **Improper Error Handling - CWE-755**
   - Crash complet du service au lieu d'une gestion gracieuse
   - Aucune isolation des erreurs (tout le service devient indisponible)
4. **Hardcoded Credentials in Client-Side Code - CWE-798**
   - Flag encodé présent dans le code JavaScript côté client
   - Accessible à tout utilisateur inspectant le code source

4.4. Causes Techniques du Problème

Pour les développeurs :

1. **Absence de validation des valeurs métier :**

Problème : Un easter egg mal implémentée ou un bug lié à une valeur spécifique provoque un crash complet.

2. **Gestion d'erreur inadéquate :**

Problème : Aucun mécanisme de capture d'exception globale pour isoler les erreurs.

3. **Messages d'erreur verbeux en production :**

Problème : Les chemins de fichiers et informations de débogage sont exposés dans le code source et la console.

4. **Credentials/Flags en client-side :**

Problème : Les informations sensibles ne doivent JAMAIS être présentes côté client, même encodées.

5. **Absence de rate limiting :**

Problème : Un attaquant peut faire autant de tentatives sans jamais déclencher le moindre système de protection.

6. **Aucune surveillance ou alerting :**

Problème : Le crash du service ne déclenche aucune alerte automatisée et le service ne se relance pas automatiquement.

4.5. Impact pour l'Entreprise

Risques immédiats :

1. **Disponibilité compromise (Availability) :**
   - Un seul attaquant peut rendre le service totalement indisponible
   - Impact sur tous les utilisateurs légitimes
   - Durée d'indisponibilité : jusqu'au redémarrage manuel
2. **Facilité d'exploitation :**
   - Aucune authentification requise
   - Exploitation triviale : une simple requête HTTP
   - Peut-être automatisée et répétée
3. **Attaque DDoS amplifiée :**
   - Un script simple peut saturer le site de requêtes avec des valeurs provoquant un crash pour rendre le site indisponible en permanence.
   - Ces crashs permanents peuvent saturer les logs et systèmes d'alerte
4. **Coûts opérationnels :**
   - Temps d'intervention pour redémarrage manuel
   - Perte de revenus pendant l'indisponibilité
   - Dégradation de l'expérience utilisateur
5. **Réputation :**
   - Service perçu comme instable et non-professionnel
   - Perte de confiance des utilisateurs
6. **Reconnaissance pour attaques futures :**
   - Exposition de la stack technique (Nginx)
   - Informations sur la structure des fichiers
   - Peut faciliter des attaques plus sophistiquées

Impact financier estimé :

- Indisponibilité : X€ / heure de downtime
- SLA breach : Pénalités contractuelles potentielles
- Coûts de support client : Gestion des tickets et plaintes

4.6. Recommandations et Remédiations

Mesures immédiates (à appliquer sous 24h) :

1. **Corriger le bug causant le crash : fixed_code**

2. **Supprimer tous les messages de débogage en production**
   - Retirer le commentaire donnant le chemin absolu du code html

- Retirer les informations importantes qui sont envoyer vers la console du coté utilisateur

3. **Implémenter un gestionnaire d'erreurs global**
4. **Configurer le redémarrage automatique**
5. **Pages d'erreur génériques sans informations sensibles**

Mesures à moyen terme (1-2 semaines) :

1. **Implémenter un rate limiting strict**
2. **Validation stricte côté serveur**
3. **Logging et monitoring**
4. **Tests de fuzzing automatisés**

Mesures à long terme (1-3 mois) :

1. **Implémenter un circuit breaker**
2. **Web Application Firewall (WAF)**
   - Déployer ModSecurity ou Cloudflare WAF
   - Règles pour détecter les patterns de fuzzing
   - Blocage automatique des IP malveillantes
3. **Monitoring et alerting avancés**
4. **Health checks et auto-healing**
5. **Conteneurisation avec orchestration**
6. **Chaos engineering**
   - Tests réguliers de résilience
   - Simulation de crashes pour vérifier la récupération
   - Validation des mécanismes de failover
7. **Code review et security testing**
   - Review systématique de toutes les PR
   - Tests de sécurité automatisés (SAST/DAST)
   - Pentest trimestriel

4.7. Références

- OWASP - Denial of Service
- CWE-400: Uncontrolled Resource Consumption
- CWE-209: Generation of Error Message Containing Sensitive Information
- CWE-755: Improper Handling of Exceptional Conditions
- OWASP Testing Guide - Fuzzing

5. Vulnérabilité #4 : Fuite d'informations par OSINT et exposition de secrets sur dépôt github public

Sous-domaine cible : `staging.rogue-sentinels.io:19480`
Type de vulnérabilité :

- `Information Disclosure`
- `OSINT (Open Source Intelligence)`
- `Exposed Secrets in Public Repository`

Criticité : CRITIQUE (CVSS 8.2)

5.1. Contexte de Découverte

Le sous-domaine `staging.rogue-sentinels.io` hébergeait un environnement de staging (pré-production) sur le port 19480. Contrairement aux vulnérabilités techniques précédentes, cette faille résulte d'une mauvaise gestion des informations sensibles et d'une exposition involontaire de secrets via des canaux publics. Une approche **OSINT** (Open Source Intelligence) a permis de tracer le développeur jusqu'à son dépôt GitHub public contenant des informations critiques.

5.2. Reproduction des Observations

5.2.1. Accès à l'environnement de staging

Navigation vers l'application : `http://staging.rogue-sentinels.io:19480`

Observations : homepage.png

- Page de destination (landing page) simple
- Interface minimaliste avec design épuré
- Aucune fonctionnalité interactive évidente
- Bannière ou mention "Staging Environment" ou "Development Preview"

Première analyse : L'environnement de staging expose souvent plus d'informations que la production (commentaires de debug, noms de développeurs, etc.)

5.2.2. Inspection du code source (F12)

Ouverture des DevTools du navigateur :

Clic droit > Inspecter l'élément (ou F12)

Analyse de la section <head>

Informations découvertes :

- Script JavaScript référencé : `script.js`

5.2.3. Analyse du fichier script.js

Accès direct au fichier JavaScript : `http://staging.rogue-sentinels.io:19480/script.js`

Contenu du fichier : script.sh

Analyse du contenu :

- Fonction simple affichant un message de bienvenue dans la console
- Aucune logique métier ou vulnérabilité évidente
- Nom : John Ellerbee

Vérification dans la console du navigateur (F12 > Console) :

“Bienvenue chez Liberty Wealth Planners »

5.2.4. OSINT - Recherche du développeur

Phase 1 : Recherche générale

Recherche Google :

« John Ellerbee »

Resultat :

8 Profils LinkedIn correspondant

Phase 2 : LinkedIn

Ouverture de LinkedIn : linkedin.png

Profil LinkedIn identifié :

- Nom : John Ellerbee
- Poste : Developer Web chez Liberty Wealth Planners
- Localisation : Schenectady, New York, États-Unis
- GitHub : github.com/DiamondHunter153

Information critique obtenue : Lien direct vers le profil GitHub public du développeur

5.2.5. Exploration du profil GitHub

Accès au profil GitHub : https://github.com/DiamondHunter153

Observations du profil :

- Nombre de repositories publics : 1
- Repository visible : landing page

Accès au repository landing-page :

https://github.com/DiamondHunter153/landing-page

5.2.6. Analyse du repository GitHub

Structure du repository : structure.sh / repo.png

Fichiers identifiés :

- `landing-page.html` - Code HTML de la page
- `script.js` - Code JavaScript (identique à celui du staging)
- `styles.css` - Feuille de style CSS
- `alpha.crt` - Fichier suspect avec extension `.crt`

Analyse initiale :

- Les 3 premiers fichiers correspondent au code du site de staging
- Le fichier `alpha.crt` est inhabituel pour un repository de landing page
- Extension `.crt` : habituellement utilisée pour les certificats SSL/TLS

5.2.7. Examen du fichier alpha.crt

Contenu du fichier : alpha.crt

Informations critiques exposées : ${FLAG_OSINTcertified}

FLAG #4 CAPTURÉ : `FLAG_OSINTcertified`

5.3. Informations Techniques Collectées

Informations exposées via OSINT :

1. **Identité du développeur :**
   - Nom complet : John Ellerbee
   - Profils sociaux : LinkedIn, GitHub
2. **Infrastructure :**
   - Environnement de staging : staging.rogue-sentinels.io:19480
3. **Secrets exposés :**
   - Flag : ${FLAG_OSINTcertified}

Vulnérabilités identifiées :

1. **Exposed Secrets in Public Repository - CWE-798**
   - Aucun scan de secrets avant le push
2. **Information Disclosure - CWE-200**
   - Informations de développeur dans le code HTML
3. **Lack of Security Awareness - CWE-1004**
   - Confusion entre repository personnel et code de production
   - Absence de formation sur les bonnes pratiques de sécurité

5.4. Causes Techniques du Problème

Pour les développeurs :

1. **Commit accidentel de secrets :** mauvaise_pratique.sh

Problème : L'utilisation de `git add .` inclut tous les fichiers sans distinction. Les secrets sont accidentellement versionnés.

2. **Absence de .gitignore approprié**

3. **Confusion entre repositories personnels et professionnels :**
    - Le développeur a utilisé son compte GitHub personnel
    - Pas de séparation claire entre code personnel et professionnel
    - Le repository devrait être privé ou hébergé sur GitHub Enterprise
4. **Métadonnées excessives dans le code**

Problème : Ces informations facilitent l'OSINT et permettent de tracer le développeur jusqu'à ses autres ressources en ligne.

5. **Absence d'outils de détection de secrets :**
    - Aucun pre-commit hook pour scanner les secrets
    - Pas d'intégration CI/CD avec GitGuardian, TruffleHog, etc.

5.5. Impact pour l'Entreprise

Risques immédiats :

1. **Compromission totale du serveur de staging**

2. **Accès aux données sensibles :**
    - Code source complet de l'application

6. **Violation RGPD :**
    - Si l'environnement de staging contient des copies de production
    - Données personnelles potentiellement compromises
    - Notification obligatoire à la CNIL
7. **Réputation et confiance :**
    - Exposition publique de mauvaises pratiques de sécurité
    - Perte de confiance des clients et partenaires
    - Impact négatif sur le recrutement (développeurs qualifiés)
8. **Supply Chain Attack :**
    - Si le repository est cloné/forké par d'autres développeurs
    - Distribution involontaire de secrets compromis

Impact financier :

- Coût de révocation et rotation de toutes les clés
- Audit de sécurité complet de l'infrastructure
- Amendes RGPD potentielles
- Coûts de remédiation et d'incident response

Impact juridique :

- Responsabilité du développeur (négligence)
- Non-conformité aux standards de sécurité (ISO 27001, SOC 2)

5.6. Recommandations et Remédiations

Mesures immédiates (à appliquer sous 24h) :

1. **Supprimer le fichier du repository GitHub**
2. **Contacter GitHub Support :**
    - Demander l'invalidation du cache GitHub
    - Signaler l'exposition accidentelle de secrets
    - GitHub peut aider à purger les caches de leurs CDN

Mesures à moyen terme (1 semaine) :

3. **Implémenter un .gitignore complet :** .gitignore
4. **Installer des pre-commit hooks pour détecter les secrets :**
    - pre-commit_hooks.sh
5. **Implémenter pre-commit avec détection de secrets :**
    - pre-commit-config.yaml
6. **Scan du repository avec TruffleHog :**
    - truffleHog.sh
7. **Formation de l'équipe de développement**
    - Bonnes pratiques Git et gestion des secrets
    - OSINT et exposition d'informations
    - Utilisation de gestionnaires de secrets (Vault, AWS Secrets Manager)
8. **Supprimer les métadonnées des développeurs du code :**
    - To-delete.sh

Mesures à long terme (1-3 mois) :

9. **Implémenter une solution de gestion de secrets :**
   secret.py
10. **Intégration CI/CD avec scanning de secrets :**
    security-scan.yml
11. **GitHub Advanced Security / Secret Scanning**

    GitHub Repository > Settings > Security & analysis

    > Enable Secret scanning

    > Enable Push protection

12. **Politique de repositories GitHub**
    - Séparer les repositories personnels et professionnels
    - Utiliser GitHub Enterprise pour le code d'entreprise
    - Repositories privés par défaut
    - Branch protection rules (require reviews, CI checks)
13. **Certificats et clés via certificate authority interne**
    - certificat.sh

14. **Monitoring GitHub avec SIEM**

    - Alertes sur les nouveaux repositories publics
    - Détection de patterns sensibles (BEGIN PRIVATE KEY, etc.)
    - Audit des commits contenant des fichiers suspects

15. **Programme de bug bounty / responsible disclosure**
    - security.txt à la racine du site web
    - Encourager le signalement responsable
    - Récompenser les chercheurs en sécurité
16. **Audit OSINT régulier**
    - osint.sh

5.7. Chaîne d'Exploitation OSINT (Résumé)

1. Reconnaissance de l'application
   - Accès à staging.rogue-sentinels.io:19480
2. Inspection du code source (F12)
   - Découverte du développeur : John Ellerbee
3. OSINT - LinkedIn
   - Profil LinkedIn avec lien GitHub
4. OSINT - GitHub
   - Repository public : landing-page

5. Énumération du repository
    - Fichier suspect : alpha.crt
6. Analyse du fichier
    - métadonnées
7. Extraction du flag
    - FLAG_OSINTcertified

5.8. Outils OSINT Utilisés

Outils manuels :

- Google Search
- LinkedIn
- GitHub

5.9. Références

- OWASP - Sensitive Data Exposure
- CWE-798: Use of Hard-coded Credentials
- CWE-200: Exposure of Sensitive Information
- GitHub - Removing sensitive data from a repository
- OSINT Framework

6. Vulnérabilité #5 : Exploitation d'une CVE CRITIQUE sur Geoserver et récupération de CREDENTIALS

Sous-domaine cible : `webservices.rogue-sentinels.io:8080`

Type de vulnérabilité :

- `CWE-95 : Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')`
- `CVE-2024-36401 : Remote Code Execution (RCE) vulnerability in evaluating property name expressions`
- `CVE-2024-36404 : Remote Code Execution (RCE) vulnerability in evaluating XPath expressions`

Criticité : CRITIQUE (CVSS 9.8)

6.1. Contexte de Découverte

Le sous-domaine `webservices.rogue-sentinels.io` exposait un service web sur le port 8080. Contrairement aux flags précédents, cette vulnérabilité simule une **condition réelle d'attaque** : l'objectif n'était pas de trouver un flag textuel mais de compromettre le système et d'exfiltrer des **informations sensibles réelles** (credentials administrateur).

L'exploitation s'est basée sur une **CVE récente et critique** affectant GeoServer, un serveur open-source de géodonnées largement utilisé dans l'industrie.

6.2. Reproduction des Observations

6.2.1. Reconnaissance de l'application

Accès à l'application via navigateur : `http://webservices.rogue-sentinels.io:8080`

Résultat : erreur 404 mais avec un lien de redirection

Accès à `/geoserver` **avec le lien de redirection :** `http://webservices.rogue-sentinels.io:8080/geoserver`

Résultat : redirection vers une interface GeoServer

Observations :

- Interface web GeoServer standard
- Version de GeoServer identifiée : 2.22.0

6.2.2. Recherche de vulnérabilités connues (CVE) pour GeoServer 2.22.0

Recherche google : « geoserver 2.22.0 cve »

Resultat :

- https://geoserver.org/vulnerability/2024/09/12/cve-2024-36401.html

Liens github des CVE donné par le site :

- https://github.com/geoserver/geoserver/security/advisories/GHSA-6jj6-gm7p-fcvv
- https://github.com/geotools/geotools/security/advisories/GHSA-w3pj-wh35-fq8w

Résumé :

Les versions de GeoServer antérieures à 2.22.6, 2.23.6, 2.24.4 et 2.25.2 sont vulnérables a des exécutions de code arbitraire à distance dû à certaines méthodes de GeoTools, utilisées par GeoServer, qui évaluent certains inputs comme des expressions XPath.

6.2.3. Recherche d'un exploit public

Recherche google : CVE_2024_36401 github

Ou

Recherche directement sur github : CVE_2024_36401

Résultat : un dépôt github contenant un script permettant de créer un remote shell sur les versions de GeoServer vulnérable à cette CVE

Clonage du dépôt github

Analyse du script `exploit.py` :

6.2.4. Mise en place du listener et exécution de l'exploit

Terminal 1 - Démarrage du listener Netcat :

Utilisation de la commande suivante : `nc -lvnp 1337`

Résultat attendu : `listening on [any] 1337 ...`

Terminal 2 - Exécution de l'exploit :

1. Création et activation d'un environnement virtuel :

```
python3 -m venv venv
source venv/bin/activate
```

2. Installation des dépendances :

```
python3 -m pip install -r requirements.txt
```

3. Exécution de l'exploit

   exploit.sh

Résultat : erreur

Modification du script nécessaire : retirer le `not` de la ligne 83 du script

4. Tentative d'exploit avec script modifié

Résultat : succès

Terminal 1 - Connexion reçue :

Résultat : resultat.png

REVERSE SHELL ÉTABLI (AVEC PRIVILÈGE ROOT) !

6.2.5. Énumération du système compromis

Exploration du répertoire courant : ls

FICHIER SENSIBLE DÉCOUVERT : `ADMIN_CREDENTIALS.txt`

6.2.6. Extraction des credentials administrateur

Lecture du fichier sensible : cat

Contenu du fichier : ADMIN_CREDENTIALS

Informations extraites :

- **Username :** `admin`
- **Password :** 00b39b8be62c67284bbb157e7143e4ef796f061ca75f6e1d06a5123ab1e5144c

Analyse du hash : analyse

6.2.7. Cracking du hash SHA-256

Méthode 1 : Hashcat (plus rapide)

Commande utilisée : hashcat

Paramètres utilisés :

- -m 1400 : Mode SHA-256
- -a 0 : Attack mode dictionary

Resultat : resultat

Méthode 2 : John The Ripper

Commandes utilisées : johnTheRipper

Resultat : resultat

CREDENTIALS COMPROMIS :

- **Username :** `admin`
- **Password :** `batman1986`

6.3. Méthode Alternative : Exploitation avec Metasploit

Pour démontrer une approche plus automatisée, voici la méthode d'exploitation utilisant **Metasploit Framework**.

1. **Lancement de Metasploit**
2. **Recherche du module GeoServer**
3. **Configuration du module**
    - Activation du module
    - Choix du RHOST
    - Choix du RPORT
4. **Configuration du payload**
5. **Vérification de la configuration**
6. **Exécution de l'exploit**
7. **Vérifier que le reverse shell fonctionne**
8. **Voir le contenu du dossier courant**
9. **Voir le contenu du fichier ADMIN_CREDENTIALS.txt**

6.4. Informations Techniques Collectées

Stack technique compromise :

- **Application :** GeoServer 2.22.0
- **Serveur :** Apache
- **Système d'exploitation :** Linux (Ubuntu/Debian)
- **Utilisateur compromis :** `root` (admin account)

CVE exploitée :

Attribut	Valeur
ID	CVE-2024-36401
Type	Remote Code Execution (RCE)
CVSS Score	9.8 (Critical)
Vecteur d'attaque	Network
Authentification requise	None
Versions affectées	<2.22.6, < 2.23.6, < 2.24.4, <2.25.2

Vulnérabilités identifiées :

1. **CVE-2024-36401 - Remote Code Execution (CWE-94)**
   - Évaluation non sécurisée d'expressions XPath
   - Permet l'exécution de code Java arbitraire
   - Exploitable sans authentification
2. **Weak Password Storage (CWE-328)**
   - Mot de passe hashé en SHA-256
   - Aucun salt utilisé
   - Facilement crackable avec une wordlist
3. **Sensitive Data Exposure (CWE-200)**
   - Fichier de credentials en clair sur le serveur
   - Accessible à l'utilisateur du service web
   - Permissions trop permissives (644)
4. **Information Disclosure (CWE-209)**
   - Page 404 révélant l'existence de /geoserver
   - Version de GeoServer exposée publiquement
5. **Outdated Software (CWE-1104)**
   - Version 2.22.0 non patchée
   - Plus d'un an de retard sur les mises à jour de sécurité

6.4. Causes Techniques du Problème

Pour les développeurs :

1. **Logiciel non mis à jour :**

   - GeoServer 2.22.0 (Release: 2023)
   - GeoServer 2.22.6+, 2.23.6+, 2.24.4+ ou 2.25.2+ (avec correctifs CVE-2024-36401)

   Problème : L'équipe n'a pas appliqué les mises à jour de sécurité malgré la publication de la CVE.

2. **Vulnérabilité dans le code GeoServer :**

   - GeoServer évaluait les noms de propriétés comme des expressions XPath
   - L'entrée utilisateur est directement évaluée comme code

   Problème : L'absence de validation des entrées utilisateur permet l'injection de code.

3. **Hashage de mot de passe inadéquat :**

   - MAUVAISE PRATIQUE – SHA-256 sans salt
   - BONNE PRATIQUE - bcrypt avec salt

   Problème : SHA-256 n'est pas considéré comme sûr pour le stockage des mots de passe : il est trop rapide, ce qui permet des attaques par force brute efficaces.

4. **Stockage de credentials en clair :**

   - Fichier texte avec credentials lisible
   - Utiliser un gestionnaire de secrets (HashiCorp Vault, AWS Secrets Manager, etc.)

5. **Page d'erreur** informative
   - Rester sur une page d'erreur générique sans informations supplémentaires

6.5. Impact pour l'Entreprise

Risques immédiats :

1. **Compromission totale du serveur :**
   - Exécution de code arbitraire avec privilèges `root`
   - Installation de backdoors permanentes
   - Pivot vers d'autres serveurs du réseau
2. **Vol de données géospatiales :**
   - GeoServer gère potentiellement des données sensibles
   - Cartes, coordonnées, informations de localisation
   - Données clients/partenaires compromises
3. **Credentials administrateur compromis :**
   - Accès à l'interface d'administration GeoServer
   - Modification des configurations
   - Ajout de nouveaux utilisateurs malveillants
   - **Réutilisation sur d'autres services** (si même mot de passe)
4. **Attaque ransomware :**
   - Un attaquant pourrait chiffrer toutes les données
5. **Crypto-mining :**
   - Utilisation des ressources serveur pour miner de la cryptomonnaie
   - Dégradation des performances
   - Augmentation des coûts d'infrastructure
6. **Conformité et légal :**
   - Violation potentielle RGPD si données personnelles
   - Non-conformité aux standards de sécurité
   - Amendes et poursuites judiciaires
7. **Réputation :**
   - Perte de confiance des utilisateurs
   - Impact sur les contrats existants
   - Difficulté à acquérir de nouveaux clients

Impact financier estimé :

- Incident response : 50,000€ - 150,000€
- Forensics et audit : 30,000€ - 80,000€
- Notification et communication : 10,000€ - 30,000€
- Amendes réglementaires : Variable (jusqu'à 4% CA)
- Perte de business : Incalculable

6.6. Recommandations et Remédiations

Mesures immédiates (à appliquer sous 24h) :

1. **Mettre à jour GeoServer IMMÉDIATEMENT :** update
2. **Migrer vers un système de hashage de mot de passe sécurisé**
    - Connectez-vous à votre interface GeoServer en tant qu'admin
    - Naviguer dans « settings » de la section « security »
    - Changer « Weak PBE » en « Strong PBE »
    - Cliquer sur « Save »
    - Utilizer bcrypt avec salt

3. **Changer IMMÉDIATEMENT tous les mots de passe**
    - Connectez-vous à votre interface GeoServer en tant qu'admin
    - Naviguer dans « Users, Groups, Role » de la section « security »
    - Allez dans l'onglet Users/Goups
    - Cliquer sur le nom d'utilisateur du compte dont vous voulez changer le mot de passe
    - Entrer le nouveau mot de passe dans la section « Password » et « Confirm passsword »
    - Appuyiez sur « Save » tout en bas de la page

4. **Assigner un compte utilisateur linux dédié pour le process GeoServer**
5. **Supprimer le fichier de credentials**
6. **Auditer les accès récents**

7. **Vérifier l'absence ou présence de backdoors**
8. **Isoler le serveur si compromission confirmée**

    - **Une fois le server sécurisé, rendre celui-ci accessible**

Mesures à moyen terme (1-2 semaines) :

8. **Implémenter un processus de gestion des patches**

9. **Implémenter un gestionnaire de secrets**

10. **Configurer un WAF (Web Application Firewall)**

11. **Restreindre l'accès au service GeoServer**

12. **Monitoring et alerting**

Mesures à long terme (1-3 mois) :

13. **Scan de vulnérabilités automatisé**
14. **Segmentation réseau**
15. **Renforcer la politique de mots de passes :**
    - Connectez-vous à votre interface GeoServer en tant qu'admin
    - Naviguez dans « Passwords » de la section « Security »
    - Cliquez sur « default » et « master » dans la section « Password Policies » pour pouvoir changer leurs attributs

    Une fois que vous êtes sur ces pages :

    - Cochez les cases « Must contain a digit », « Must contain an uppercase letter » et « Must contain a lowercase letter » pour forcer les mots de passes à avoir au moins un chiffre, une lettre majuscule et une lettre minuscule
    - Changez le champ « Minimum length » pour forcer une longueur de mot de passe assez longue pour être robuste, nous conseillions 14 minimum
    - Cliquez sur « Save »

    Du coté server

    - Vérifier que les mots des passes contiennent au moins un caractère spécial

    Maintenant que les nouveaux mots de passes sont robustes :

    - Changer les mots de passes de manière régulière, tous les 90 jours par exemple
    - Conserver un historique des derniers mots de passes utiliser et ne laisser pas les nouveaux mots de passes être les mêmes
    - Ajouter l'option de sécuriser les comptes avec une authentication multifacteur et rendez-la obligatoire pour les comptes administrateur
16. **Formation continue de l'équipe**
    - Veille CVE sur les technologies utilisées
    - Exercices de réponse aux incidents (tabletop exercises)
    - Formation OWASP et secure coding
17. **Bug bounty program**
    - Encourager le signalement responsable
    - Récompenser les chercheurs en sécurité
18. **Audit de sécurité régulier**
    - Pentest externe trimestriel
    - Exercices Red team annuel
    - Code review focalisée sécurité

6.7. Chaîne d'Exploitation Complète (Kill Chain)

CYBER KILL CHAIN

1. RECONNAISSANCE
    - Nmap scan → Port 8080 ouvert
    - Page 404 → Découverte /geoserver
    - Version GeoServer 2.22.0 identifiée
2. WEAPONIZATION
    - Recherche CVE-2024-36401
    - Téléchargement exploit GitHub
    - Modification du script (bypass version check)
3. DELIVERY
    - Envoi payload XML via /geoserver/wfs
    - Injection XPath malveillante
4. EXPLOITATION
    - Exécution de code Java arbitraire
    - Système : reverse shell bash
5. INSTALLATION
    - Reverse shell connecté
    - Accès utilisateur tomcat obtenu
6. COMMAND & CONTROL
    - Session shell interactive établie
    - Énumération du système (ls, cat, whoami)
7. ACTIONS ON OBJECTIVES
    - Découverte ADMIN_CREDENTIALS.txt
    - Extraction du hash SHA-256
    - Cracking → batman1984
    - CREDENTIALS COMPROMIS

6.8. Références

- CVE-2024-36401 - NVD
- GitHub - CVE-2024-36401 Exploit
- CWE-94: Improper Control of Generation of Code
- CWE-328: Use of Weak Hash
- OWASP - Password Storage Cheat Sheet
- Metasploit Framework Documentation
- MITRE ATT&CK - Exploitation of Remote Services

TOUS LES FLAGS ONT ÉTÉ CAPTURÉS !

Voici le récapitulatif des 5 vulnérabilités exploitées :

#	Sous-domaine	Vulnérabilité	Flag/Credentials
1	`fileserver.rogue-sentinels.io`	Bruteforce + Weak Encryption	`FLAG_Web2000CorpBackups`
2	`travel.rogue-sentinels.io`	Unrestricted File Upload + RCE	`FLAG_SalmonDatabaseCredentials`
3	`restricted.rogue-sentinels.io`	DoS + Information Disclosure	`FLAG_FuzzyDOS`
4	`staging.rogue-sentinels.io`	OSINT + Exposed Secrets	`FLAG_OSINTcertified`
5	`webservices.rogue-sentinels.io`	CVE-2024-36401 RCE + Weak Hash	`Username : admin` `Password : batman1984`