Université du Québec
à Chicoutimi

Module d'informatique et de mathématique

Département d'informatique et de mathématiques

Rapport

Stage projet I

Sujet : Scan des vulnérabilités dans le cloud (K8S et Docker)

8SEC201

Automne 2025

Kelly Noelle Mapoue Pekeko

Sous la supervision du : Professeur Jonathan ROY

TABLE DES MATIERES

INTRODUCTION

La multiplication des environnements informatiques modernes, tels que les conteneurs Docker, les clusters Kubernetes et les machines virtuelles, a considérablement accru la surface d'attaque des systèmes d'information. Dans ce contexte, la détection et la gestion des vulnérabilités constituent un enjeu central de la cybersécurité défensive.

Les outils de scan automatisés, comme **Trivy**, permettent d'identifier rapidement un grand nombre de vulnérabilités. Toutefois, les résultats générés sont souvent volumineux et difficiles à exploiter directement, car ils manquent de **contexte**, de **priorisation** et d'**informations détaillées** sur le risque réel associé à chaque vulnérabilité.

Ce projet vise à mettre en place une **chaîne complète de gestion des vulnérabilités**, allant de la détection à la priorisation, en s'appuyant sur des outils open source et des bases de données de référence telles que la **National Vulnerability Database (NVD)**. L'objectif est de démontrer qu'une approche automatisée et contextualisée permet d'améliorer significativement la prise de décision en matière de cybersécurité.

I- Contexte du projet et Objectifs principaux

1.1 Contexte

Dans le cadre de l'exploitation des différents cours appris pendant mes 4 sessions a L'UQAC en particulier le cours de *Cybersécurité défensive : vulnérabilités et incidents*, l'objectif initial du projet était de mettre en œuvre une **chaîne complète de détection, d'analyse et de priorisation des vulnérabilités** dans un environnement réaliste et proche des conditions opérationnelles.

Le projet s'inscrit dans une approche **DevSecOps**, combinant :

- des environnements virtualisés la VM linux offert pour la réalisation du projet ,
- des technologies de conteneurisation tel que Kubernetes ,Docker ,
- et des outils open source de cybersécurité tel que trivy .

1.2 Objectifs

Dans le but de se familiariser avec les environnements cloud il était question pour moi de développer une solution automatisée permettant :

- de scanner a l'aide de trivy (outils open source au choix) plusieurs environnements informatiques tel que la VM linux, déployer kubernetes et docker a fin de les scanner également,
- Générer un rapport CSV consolidé contenant l'ensemble des vulnérabilités détectées,
- D'enrichir les vulnérabilités détectées a l'aide de la base de donner de la NVD via la cle d'api qui permet d'interroger la base de donner ,pour la réalisation de cette étapes il fallait développer un script Python permettant :
    - d'interroger l'API du NVD à l'aide d'une clé API,
    - d'enrichir les CVE avec des informations supplémentaires.
- Et de les prioriser en fonction de leur criticité réelle ,afin de savoir laquelle des vulnerabilites sera traite en premier pour réaliser cet étapes j'ai développer un second script Python pour :

- catégoriser les actifs,
- reprioriser les vulnérabilités selon leur criticité.

- Et pour finir Produire des rapports clairs et exploitables (CSV, Excel, PDF).

II- Déploiement des environnements de test

2.1 Environnements analysés

Pour l'implémentation de ce projet les trois types d'environnements suivant ont été déployés et analysés :

2.2 Cluster Kubernetes (K8S)

Kubernetes (K8S) qui est une plateforme open source d'orchestration(planificateur) de conteneurs permettant de déployer, gérer et mettre à l'échelle des applications conteneurisées. Dans ce projet, Kubernetes a été utilisé pour simuler un environnement de production moderne, où plusieurs conteneurs et services coexistent , Avant l'installation de Kubernetes, les éléments suivants sont requis Un système Linux (Ubuntu recommandé) ,avoir Docker installé et fonctionnel ,avoir un accès administrateur (sudo) et pour finir une Connexion Internet

Comment l'installer ? on peut utiliser la documentation du site officiel de K8S, mais une présentation minimale des étapes à suivre est la suivante :

2.1.2 Installation en local

-Installer minikube qui permet de déployer un cluster K8S local :

```
sudo apt update

sudo apt install -y curl wget apt-transport-https

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

sudo install minikube-linux-amd64 /usr/local/bin/minikube

et pour demarrer le cluster : minikube start

21.3 kubeadm environnement plus realiste

plus complique a utiliser et a installer par ce qu'il faut installer toutes les ressources de demarrages suivantes

- Installation de kubeadm, kubelet, kubectl
- Initialisation du cluster avec kubeadm init

Pour avoir explorer les 2 methodes je conseillerais pour un debutant de s'entrainer avec minikube ,et par la suite utiliser Kubeadm pour éviter de s'embrouiller lors des burgg avec les dependances .

2.2. Environnement Docker

Docker est une plateforme de conteneurisation permettant d'exécuter des applications de manière isolée dans des conteneurs légers.
Dans ce projet, Docker a été utilisé pour analyser les images et dépendances applicatives, qui représentent une source fréquente de vulnérabilités.

2.2.1 Installation de Docker

sudo apt update

sudo apt install -y docker.io

sudo systemctl start docker

sudo systemctl enable docker

pour ajouter un utilisateur au groupe docker : sudo usermod -aG docker $USER

par la suite pour vérifier l'installation : docker --version

docker run hello-world

2.2.2 Analyse d'images Docker

Elle se fait avec la commande trivy image nginx:latest mais avant il est necessaire de se rassurer que les images sont bien télécharger et également que trivy est configure et pour finir lancer le scan des dépendances et bibliothèques embarquées

2.3 Machine virtuelle Linux

Une machine virtuelle Linux a été fournie dans le cadre du cours afin de représenter un système classique en production.
Elle permet d'analyser les vulnérabilités liées :

- au système d'exploitation,
- Aux paquets installés,
- Aux bibliothèques système.

III- des vulnérabilités avec Trivy

3.1 Rôle de trivy

Pour la phase de scan des environnement, j'ai fait la sélection d'un outil open source parmi tant d'autre qui est Trivy pour détecter automatiquement les vulnérabilités de sécurité dans les différents types d'environnements qu'on avait a sscanner. trivy**a été cchoisi pour**les raison suivantes :

Outil open source largement utilisé, capable d'analyser :

- Des **images Docker** (packages OS et dépendances applicatives),
- Des **clusters Kubernetes** (ressources, images, configurations),
- Des **systèmes d'exploitation** (root filesystem),
- Des **dépendances applicatives** (Java, Python, Node.js, etc.),
- Génération de rapports structurés (JSON, CSV)

3.2 Génération des rapports de scan

Trivy a été utilisé pour scanner :

- Les images Docker
- Les ressources Kubernetes
- La machine virtuelle Linux

Les résultats ont été consolidés dans un **rapport CSV unique**, contenant l'ensemble des vulnérabilités détectées, incluant :

- Les identifiants CVE
- Les packages affectés
- La sévérité initiale
- L'environnement concerné (K8S, Docker, VM)

IV- Enrichissement des données via la NVD

4.1 Présentation de la National Vulnerability Database (NVD)

La National Vulnerability Database (NVD) est la base de données officielle de vulnérabilités maintenue par le National Institute of Standards and Technology (NIST) aux États-Unis.

Elle constitue la référence mondiale en matière d'informations sur les vulnérabilités de sécurité, notamment :

- Identifiants CVE
- Scores CVSS (v2, v3.x, et plus récemment v4.0)
- Types de faiblesses CWE
- Informations sur l'exploitation
- Références techniques et correctifs

La NVD enrichit les données brutes issues des scanners de vulnérabilités en fournissant une analyse normalisée et approfondie.

4.2 Accès à la base de données NVD et rôle dans le projet

Afin d'obtenir des informations plus détaillées sur chaque vulnérabilité trouve lors du scan dans les différents environnements, j'ai fait la demande d'une **clé API officielle du NVD** sur leur site officiel pour avoir la possibilité d'interroger la BD de la NVD et enrichir mes rapports.

Les rapports Trivy fournissent une **liste de CVE détectées** la clé d'API permet donc d'accéder à des informations fiables et normalisées, notamment :

- Scores CVSS
- Types de vulnérabilités ou de faiblesses (CWE)
- Descriptions détaillées
- Dates de publication et de mise à jour

Par la suite ces donnes seront utilise pour la **catégorisation des actifs, et** la **repriorisation des vulnérabilités plus tard**

4.2 Script Python d'enrichissement

Pour enrichir mes différents rapports je ne l'ai pas fait en faisant des recherches ,du type CVE par CVE il était nécessaire d'automatiser tout ce processus, d'où l'importance d'une clé d'api ,j'ai donc développe un **premier script Python** afin de :

1. Lire le rapport CSV généré par Trivy
2. Extraire les identifiants CVE
3. Interroger l'API du NVD pour chaque CVE
4. Enrichir le rapport initial avec les données récupérées

Le résultat est un **rapport enrichi**, combinant les données issues de Trivy et de la NVD.

4.3 Intégration de l'API NVD dans le script Python

4.3.1 Bibliothèques Python utilisées

Le développement des scripts Python de ce projet repose sur plusieurs bibliothèques, chacune jouant un rôle précis dans les différentes phases : collecte des données, enrichissement via l'API NVD, traitement, visualisation et génération de rapports

Le script Python utilise les bibliothèques suivantes :

➢ **Bibliothèques pour l'interaction avec l'API NVD**

- **Requests** utilisée pour effectuer les requêtes HTTP vers l'API REST de la NVD.
- **Json** permet le traitement et l'extraction des données JSON retournées par l'API.
- **Time** employée pour gérer les délais entre les requêtes afin de respecter les limites imposées par l'API NVD, et éviter le blocage ou le rejet des requêtes.

➢ **Bibliothèques principales de traitement et d'analyse**

- **Pandas** : utilisée pour la manipulation et l'analyse des données tabulaires.
  permettant la lecture et l'écriture de fichiers CSV, la fusion des données issues de Trivy et de la NVD, et pour finir le tri, le filtrage et la priorisation des vulnérabilités.
- **Numpy** elle est employée pour les calculs numériques, notamment le calcul de scores agrégés, les pondérations liées à la criticité des actifs, les opérations mathématiques nécessaires à la repriorisation.

➢ **Bibliothèques de visualisation**

- **Matplotlib** c'est la Bibliothèque de base pour la création de graphiques statistiques.
  Elle est utilisée pour représenter la distribution des scores CVSS,visualiser le nombre de vulnérabilités par sévérité, et pour appuyer l'analyse par des graphiques clairs et interprétables.
- **Seaborn** est une extension de matplotlib offrant des graphiques plus lisibles et esthétiques. Elle facilite la visualisation comparative, l'analyse des tendances, la présentation des résultats dans le rapport final.

➢ **Bibliothèques pour la génération de rapports**

- **fpdf / fpdf2** utilisée pour la génération automatique de rapports **PDF**.
  Elle permettant la création de tableaux synthétiques, l'intégration de graphiques, l'ajout de légendes et d'explications pour faciliter la compréhension des résultats.
- **Xlsxwriter** utilisée pour l'export des résultats au format **Excel (.xlsx)**, afin de permettre une exploitation ultérieure des données, une analyse manuelle complémentaire, une compatibilité avec des outils bureautiques.

➢ **Bibliothèques utilitaires et de support**

- **Argparse** utilisée pour gérer les arguments en ligne de commande, permettant de spécifier les dossiers d'entrée et de sortie, et de rendre le script réutilisable et flexible.
- **Logging** permet d'effectuer la journalisation des événements du script tel que les erreurs, les avertissements, et les informations d'exécution.
- **Datetime** utilisée pour la gestion des dates et horodatages ,nommage des fichiers de sortie, et le suivi temporel des analyses.
- **Typing** fournit des annotations de type afin d'améliorer la lisibilité du code, la maintenabilité, la robustesse du développement.

4.3.2 Processus d'enrichissement

Le processus d'enrichissement a pour objectif d'augmenter la valeur informationnelle des vulnérabilités détectées par Trivy, en exploitant les données officielles de la National Vulnerability Database (NVD) ainsi que des sources complémentaires (EPSS, KEV).

Pour chaque CVE extraite des rapports Trivy, le script applique les étapes suivantes :

➢ **Extraction des identifiants CVE**
Les rapports générés par Trivy (Kubernetes, Docker et machine virtuelle Linux) sont analysés afin d'extraire les identifiants CVE uniques associés aux vulnérabilités détectées.

➢ **Construction et envoi de la requête API**
Pour chaque CVE, une requête HTTP est construite vers l'API REST de la NVD en utilisant une **clé API personnelle**, conformément aux exigences du NVD.

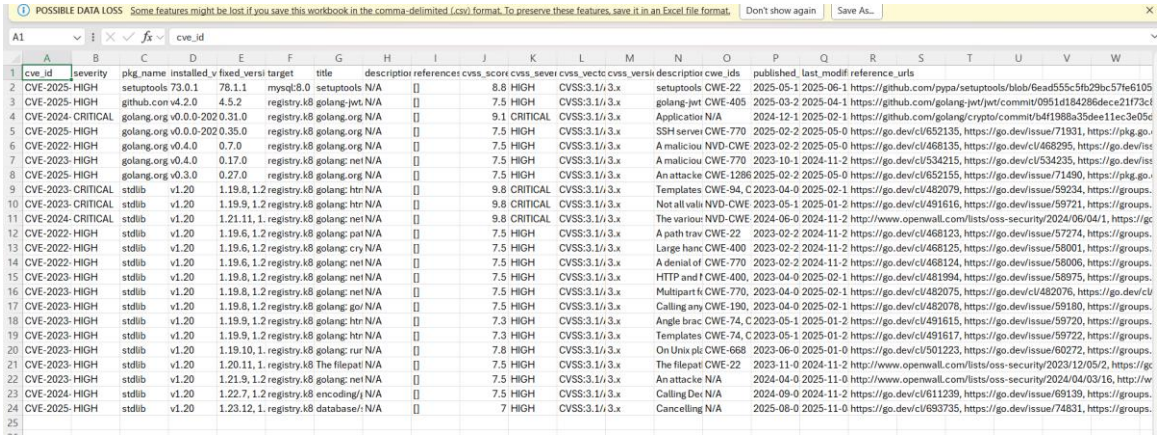
➢ **Réception et traitement de la réponse JSON**
Les réponses de l'API, fournies au format JSON, sont analysées afin d'extraire les champs pertinents.
Le module pandas est utilisé pour structurer ces données sous forme tabulaire.

➢ **Extraction des informations clés**
Le script récupère :

- les scores CVSS disponibles,
- la description officielle,
- les dates de publication et de modification,
- les identifiants CWE associés,
- ainsi que les métadonnées complémentaires (EPSS, KEV).

- **Normalisation et nettoyage des données**
  Les champs récupérés sont ensuite normalisés (noms de colonnes, types de données, valeurs manquantes) afin de garantir une cohérence globale du jeu de données.
- **Fusion avec les données Trivy**
  Les informations enrichies issues de la NVD sont fusionnées avec les données initiales de Trivy, constituant ainsi une base de données consolidée prête pour la phase de priorisation.

Ce processus est répété automatiquement pour l'ensemble des CVE détectées



	cve_id	severity	pkg_name	installed_v	fixed_versi	target	title	description	references	cvss_score	cvss_sever	cvss_vecto	cvss_versi	description	cwe_ids	published_	last_modifi	reference_urls
2	CVE-2025-	HIGH	setuptools	73.0.1	78.1.1	mysql:8.0	setuptools	N/A	[]		8.8	HIGH	CVSS:3.1/	3.x	setuptools	CWE-22	2025-05-1	2025-06-1	https://github.com/pypa/setuptools/blob/6ead555c5fb29bc57fe6105
3	CVE-2025-	HIGH	github.com	v4.2.0	4.5.2	registry.k8	golang-jwt	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	golang-jwt	CWE-405	2025-03-2	2025-04-1	https://github.com/golang-jwt/jwt/commit/0951d184286dece21f73cf
4	CVE-2024-	CRITICAL	golang.org	v0.0.0-202	0.31.0	registry.k8	golang.org	N/A	[]		9.1	CRITICAL	CVSS:3.1/	3.x	Applicatio	N/A	2024-12-1	2025-02-1	https://github.com/golang/crypto/commit/b4f1988a35dee11ec3e05c
5	CVE-2025-	HIGH	golang.org	v0.0.0-202	0.35.0	registry.k8	golang.org	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	SSH server	CWE-770	2025-02-2	2025-05-0	https://go.dev/cl/652135, https://go.dev/issue/71931, https://pkg.go.
6	CVE-2022-	HIGH	golang.org	v0.4.0	0.7.0	registry.k8	golang.org	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	A maliciou	NVD-CWE	2023-02-2	2025-05-0	https://go.dev/cl/468135, https://go.dev/cl/468295, https://go.dev/iss
7	CVE-2023-	HIGH	golang.org	v0.4.0	0.17.0	registry.k8	golang: net	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	A maliciou	CWE-770	2023-10-1	2024-11-2	https://go.dev/cl/534215, https://go.dev/cl/534235, https://go.dev/iss
8	CVE-2025-	HIGH	golang.org	v0.3.0	0.27.0	registry.k8	golang.org	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	An attacke	CWE-1286	2025-02-2	2025-05-0	https://go.dev/cl/652155, https://go.dev/issue/71490, https://pkg.go.
9	CVE-2023-	CRITICAL	stdlib	v1.20	1.19.8, 1.2	registry.k8	golang: htr	N/A	[]		9.8	CRITICAL	CVSS:3.1/	3.x	Templates	CWE-94, C	2023-04-0	2025-02-1	https://go.dev/cl/482079, https://go.dev/issue/59234, https://groups.
10	CVE-2023-	CRITICAL	stdlib	v1.20	1.19.9, 1.2	registry.k8	golang: htr	N/A	[]		9.8	CRITICAL	CVSS:3.1/	3.x	Not all vali	NVD-CWE	2023-05-1	2025-01-2	https://go.dev/cl/491616, https://go.dev/issue/59721, https://groups.
11	CVE-2024-	CRITICAL	stdlib	v1.20	1.21.11, 1.	registry.k8	golang: net	N/A	[]		9.8	CRITICAL	CVSS:3.1/	3.x	The variou	NVD-CWE	2024-06-0	2024-11-2	http://www.openwall.com/lists/oss-security/2024/06/04/1, https://gc
12	CVE-2022-	HIGH	stdlib	v1.20	1.19.6, 1.2	registry.k8	golang: pat	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	A path trav	CWE-22	2023-02-2	2024-11-2	https://go.dev/cl/468123, https://go.dev/issue/57274, https://groups.
13	CVE-2022-	HIGH	stdlib	v1.20	1.19.6, 1.2	registry.k8	golang: cry	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	Large hanc	CWE-400	2023-02-2	2024-11-2	https://go.dev/cl/468125, https://go.dev/issue/58001, https://groups.
14	CVE-2022-	HIGH	stdlib	v1.20	1.19.6, 1.2	registry.k8	golang: net	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	A denial of	CWE-770	2023-02-2	2024-11-2	https://go.dev/cl/468124, https://go.dev/issue/58006, https://groups.
15	CVE-2023-	HIGH	stdlib	v1.20	1.19.8, 1.2	registry.k8	golang: net	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	HTTP and f	CWE-400,	2023-04-0	2025-02-1	https://go.dev/cl/481994, https://go.dev/issue/58975, https://groups.
16	CVE-2023-	HIGH	stdlib	v1.20	1.19.8, 1.2	registry.k8	golang: net	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	Multipart fo	CWE-770,	2023-04-0	2025-02-1	https://go.dev/cl/482075, https://go.dev/cl/482076, https://go.dev/cl/
17	CVE-2023-	HIGH	stdlib	v1.20	1.19.8, 1.2	registry.k8	golang: go/	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	Calling any	CWE-190,	2023-04-0	2025-02-1	https://go.dev/cl/482078, https://go.dev/issue/59180, https://groups.
18	CVE-2023-	HIGH	stdlib	v1.20	1.19.9, 1.2	registry.k8	golang: htr	N/A	[]		7.3	HIGH	CVSS:3.1/	3.x	Angle brac	CWE-74, C	2023-05-1	2025-01-2	https://go.dev/cl/491615, https://go.dev/issue/59720, https://groups.
19	CVE-2023-	HIGH	stdlib	v1.20	1.19.9, 1.2	registry.k8	golang: htr	N/A	[]		7.3	HIGH	CVSS:3.1/	3.x	Templates	CWE-74, C	2023-05-1	2025-01-2	https://go.dev/cl/491617, https://go.dev/issue/59722, https://groups.
20	CVE-2023-	HIGH	stdlib	v1.20	1.19.10, 1.	registry.k8	golang: rur	N/A	[]		7.8	HIGH	CVSS:3.1/	3.x	On Unix pl	CWE-668	2023-06-0	2025-01-0	https://go.dev/cl/501223, https://go.dev/issue/60272, https://groups.
21	CVE-2023-	HIGH	stdlib	v1.20	1.20.11, 1.	registry.k8	The filepat	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	The filepat	CWE-22	2023-11-0	2024-11-2	http://www.openwall.com/lists/oss-security/2023/12/05/2, https://gc
22	CVE-2023-	HIGH	stdlib	v1.20	1.21.9, 1.2	registry.k8	golang: net	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	An attacke	N/A	2024-04-0	2025-11-0	http://www.openwall.com/lists/oss-security/2024/04/03/16, http://w
23	CVE-2024-	HIGH	stdlib	v1.20	1.22.7, 1.2	registry.k8	encoding/	N/A	[]		7.5	HIGH	CVSS:3.1/	3.x	Calling Dec	N/A	2024-09-0	2024-11-2	https://go.dev/cl/611239, https://go.dev/issue/69139, https://groups.
24	CVE-2025-	HIGH	stdlib	v1.20	1.23.12, 1.	registry.k8	database/	N/A	[]		7	HIGH	CVSS:3.1/	3.x	Cancelling	N/A	2025-08-0	2025-11-0	https://go.dev/cl/693735, https://go.dev/issue/74831, https://groups.

4.3.3 Données enrichies récupérées

Les données enrichies sont organisées en plusieurs catégories afin de faciliter leur exploitation.

Informations générales

- Identifiant **CVE**
- Description officielle issue de la NVD
- Date de publication
- Date de dernière modification

Scores de vulnérabilité

- Score **CVSS** (version disponible la plus récente)
- Vecteur CVSS
- Niveau de sévérité normalisé (CRITICAL, HAUT, MOYEN, FAIBLE)

Données prédictives et opérationnelles

- **EPSS (Exploit Prediction Scoring System)**
  Indique la probabilité qu'une vulnérabilité soit exploitée dans le monde réel.
- **KEV (Known Exploited Vulnerabilities)**
  Indique si la vulnérabilité figure dans la liste officielle des vulnérabilités activement exploitées.

Faiblesses associées

- Identifiant **CWE**
- Catégorie de faiblesse logicielle

Champs calculés (ceux-ci apparaissent après l'exécution du deuxième script a la priorisation)

- **priority_score**
  Score agrégé calculé à partir de :
  - CVSS,
  - EPSS,
  - présence dans la liste KEV.
- **priority_level**
  Classement final de la vulnérabilité :
  - **P1** : Critique / action immédiate
  - **P2** : Priorité moyenne
  - **P3** : Priorité basse

Références

- Liens vers les bulletins de sécurité
- Correctifs officiels
- Analyses techniques

4.3.4 Gestion des erreurs et limitations

Plusieurs mécanismes ont été mis en place pour assurer la robustesse du script :

- Gestion des CVE absentes de la NVD,
- Gestion des réponses vides ou incomplètes,
- délais (sleep) pour respecter les limites de l'API,
- Journalisation des erreurs.

Ces mécanismes évitent l'arrêt du script en cas de problème ponctuel.

4.3.5 Normalisation et nettoyage des données

Les données issues de la NVD présentent une forte hétérogénéité liée :

- À l'ancienneté des CVE,
- Aux différentes versions de CVSS,
- À la qualité variable des métadonnées.

Le script effectue donc les opérations suivantes :

- normalisation des noms de colonnes,
- conversion des types (scores numériques, dates),
- harmonisation des niveaux de sévérité,
- remplacement des valeurs manquantes par des valeurs par défaut,
- uniformisation du format des dates.

Cette permet de garantit la cohérence des calculs de priorisation et la lisibilité des rapports finaux.

4.3.6 Limites de l'enrichissement via la NVD

Malgré la richesse des données fournies, l'enrichissement via la NVD présente certaines limites :

- certaines CVE ne disposent pas de score CVSS exploitable,
- les scores EPSS et KEV dépendent de la qualité des bases externes,
- absence de gestion avancée de la pagination de l'API,
- limitations liées aux quotas de requêtes,
- le score de priorité reste une approximation et ne remplace pas une analyse humaine.

Ces limites justifient la mise en place de la **phase suivante du projet**, dédiée à la **catégorisation des actifs et à la repriorisassions contextuelle** des vulnérabilités.

Top 20 des Vulnérabilités Critiques et Hautes

CVE	Sévérité	CVSS	Package	CWE
CVE-2024-45337	CRITICAL	9.1	golang.org/x/crypto	N/A
CVE-2023-24538	CRITICAL	9.8	stdlib	CWE-94, CWE-94
CVE-2023-24540	CRITICAL	9.8	stdlib	NVD-CWE-noinfo, CWE-77
CVE-2024-24790	CRITICAL	9.8	stdlib	NVD-CWE-noinfo
CVE-2025-47273	HIGH	8.8	setuptools	CWE-22
CVE-2023-45288	HIGH	7.5	stdlib	N/A
CVE-2023-45283	HIGH	7.5	stdlib	CWE-22
CVE-2023-29403	HIGH	7.8	stdlib	CWE-668
CVE-2023-29400	HIGH	7.3	stdlib	CWE-74, CWE-94
CVE-2023-24539	HIGH	7.3	stdlib	CWE-74, CWE-94
CVE-2023-24537	HIGH	7.5	stdlib	CWE-190, CWE-190
CVE-2023-24536	HIGH	7.5	stdlib	CWE-770, CWE-770
CVE-2023-24534	HIGH	7.5	stdlib	CWE-400, CWE-400
CVE-2022-41724	HIGH	7.5	stdlib	CWE-400
CVE-2024-34156	HIGH	7.5	stdlib	N/A
CVE-2022-41722	HIGH	7.5	stdlib	CWE-22
CVE-2025-22868	HIGH	7.5	golang.org/x/oauth2	CWE-1286
CVE-2023-39325	HIGH	7.5	golang.org/x/net	CWE-770
CVE-2022-41723	HIGH	7.5	golang.org/x/net	NVD-CWE-Other
CVE-2025-22869	HIGH	7.5	golang.org/x/crypto	CWE-770

V- Catégorisation et priorisation des vulnérabilités

La sévérité d'une vulnérabilité, exprimée par des métriques techniques telles que le score CVSS, ne suffit pas à elle seule pour déterminer l'urgence de sa correction.
En effet, le risque réel dépend fortement du contexte dans lequel la vulnérabilité est présente, notamment du type d'actif affecté.

Dans un environnement réel :

- une vulnérabilité sur un service exposé peut avoir un impact critique,
- la même vulnérabilité sur un système interne isolé peut représenter un risque moindre.

La catégorisation des actifs permet donc :

- de contextualiser l'analyse des vulnérabilités,
- d'orienter les efforts de correction vers les composants les plus sensibles,
- de s'inscrire dans une démarche de gestion du risque plutôt que de simple énumération de failles.

5.1 Catégorisation des actifs

Pour continuer un script intermédiaire permet d'extraire automatiquement les *targets* à partir des différents rapports Trivy (Kubernetes, Docker et machine virtuelle Linux).
Cette étape permet d'associer chaque vulnérabilité à son environnement et à son composant d'origine, fournissant ainsi un premier niveau de contextualisation technique.
Les calculs de priorité s'appuient ensuite sur cette information, combinée aux scores CVSS, EPSS et à la présence dans le catalogue KEV.

Un second script Python principale a ensuite été développé pour effectuer la catégorisation des actifs, en s'appuyant sur :

- Le type d'environnement (K8S, Docker, VM)
- La criticité supposée des actifs
- Les données enrichies issues de la NVD

  Cette étape permet de prendre en compte le **contexte opérationnel**, indispensable à une évaluation réaliste du risque.

5.2 Repriorisation des vulnérabilités

À partir des données enrichies et de la catégorisation des actifs, les vulnérabilités ont été classées selon leur **criticité réelle**, en intégrant :

- La gravité (CVSS)
- La probabilité d'exploitation (EPSS)
- L'exploitation active (KEV)
- L'impact lié à l'actif concerné

  Cette repriorisation permet de distinguer :
- Les vulnérabilités critiques nécessitant une correction immédiate
- Celles pouvant être traitées dans un cycle de maintenance normal

. Repriorisation des vulnérabilités

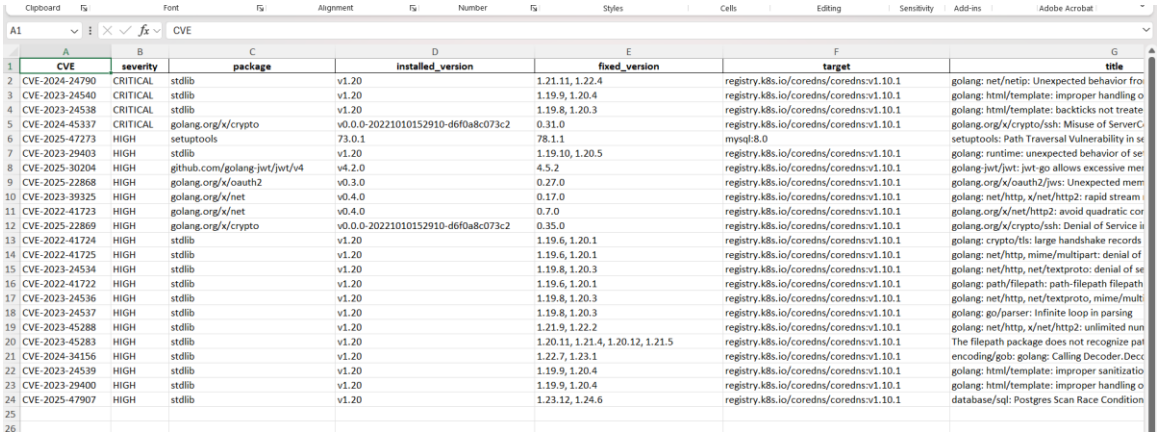
5.2.1 Méthodologie

Le script implémente une **méthodologie de priorisation orientée risque**, visant à classer les vulnérabilités non seulement selon leur gravité technique, mais également selon leur **probabilité d'exploitation** et **l'impact métier de l'actif concerné**.

Cette approche repose sur la combinaison de quatre indicateurs complémentaires :

- **CVSS (Common Vulnerability Scoring System)** : mesure la gravité technique intrinsèque de la vulnérabilité.
- **EPSS (Exploit Prediction Scoring System)** : estime la probabilité d'exploitation dans un contexte réel.
- **KEV (Known Exploited Vulnerabilities)** : indique si la vulnérabilité est activement exploitée.
- **Facteur FIPS (FIPS 199)** : représente la **criticité métier de l'actif**, déterminée à partir de la cible (target) analysée
  (ex. Kubernetes, Docker, VM Linux en production ou en test).

L'objectif est d'appliquer une logique de **Risk-Based Vulnerability Management (RBVM)**, plus représentative des risques opérationnels réels



CVE	severity	package	installed_version	fixed_version	target	title
CVE-2024-24790	CRITICAL	stdlib	v1.20	1.21.11, 1.22.4	registry.k8s.io/coredns/coredns:v1.10.1	golang: net/netip: Unexpected behavior fro
CVE-2023-24540	CRITICAL	stdlib	v1.20	1.19.9, 1.20.4	registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: improper handling o
CVE-2023-24538	CRITICAL	stdlib	v1.20	1.19.8, 1.20.3	registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: backticks not treate
CVE-2024-45337	CRITICAL	golang.org/x/crypto	v0.0.0-20221010152910-d6f0a8c073c2	0.31.0	registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/crypto/ssh: Misuse of ServerC
CVE-2025-47273	HIGH	setuptools	73.0.1	78.1.1	mysql:8.0	setuptools: Path Traversal Vulnerability in se
CVE-2023-29403	HIGH	stdlib	v1.20	1.19.10, 1.20.5	registry.k8s.io/coredns/coredns:v1.10.1	golang: runtime: unexpected behavior of se
CVE-2025-30204	HIGH	github.com/golang-jwt/jwt/v4	v4.2.0	4.5.2	registry.k8s.io/coredns/coredns:v1.10.1	golang-jwt/jwt: jwt-go allows excessive mer
CVE-2025-22868	HIGH	golang.org/x/oauth2	v0.3.0	0.27.0	registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/oauth2/jws: Unexpected mem
CVE-2023-39325	HIGH	golang.org/x/net	v0.4.0	0.17.0	registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, x/net/http2: rapid stream
CVE-2022-41723	HIGH	golang.org/x/net	v0.4.0	0.7.0	registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/net/http2: avoid quadratic cor
CVE-2025-22869	HIGH	golang.org/x/crypto	v0.0.0-20221010152910-d6f0a8c073c2	0.35.0	registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/crypto/ssh: Denial of Service i
CVE-2022-41724	HIGH	stdlib	v1.20	1.19.6, 1.20.1	registry.k8s.io/coredns/coredns:v1.10.1	golang: crypto/tls: large handshake records
CVE-2022-41725	HIGH	stdlib	v1.20	1.19.6, 1.20.1	registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, mime/multipart: denial of
CVE-2023-24534	HIGH	stdlib	v1.20	1.19.8, 1.20.3	registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, net/textproto: denial of se
CVE-2022-41722	HIGH	stdlib	v1.20	1.19.6, 1.20.1	registry.k8s.io/coredns/coredns:v1.10.1	golang: path/filepath: path-filepath filepath
CVE-2023-24536	HIGH	stdlib	v1.20	1.19.8, 1.20.3	registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, net/textproto, mime/mult
CVE-2023-24537	HIGH	stdlib	v1.20	1.19.8, 1.20.3	registry.k8s.io/coredns/coredns:v1.10.1	golang: go/parser: Infinite loop in parsing
CVE-2023-45288	HIGH	stdlib	v1.20	1.21.9, 1.22.2	registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, x/net/http2: unlimited nun
CVE-2023-45283	HIGH	stdlib	v1.20	1.20.11, 1.21.4, 1.20.12, 1.21.5	registry.k8s.io/coredns/coredns:v1.10.1	The filepath package does not recognize pat
CVE-2024-34156	HIGH	stdlib	v1.20	1.22.7, 1.23.1	registry.k8s.io/coredns/coredns:v1.10.1	encoding/gob: golang: Calling Decoder.Decc
CVE-2023-24539	HIGH	stdlib	v1.20	1.19.9, 1.20.4	registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: improper sanitizatio
CVE-2023-29400	HIGH	stdlib	v1.20	1.19.9, 1.20.4	registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: improper handling o
CVE-2025-47907	HIGH	stdlib	v1.20	1.23.12, 1.24.6	registry.k8s.io/coredns/coredns:v1.10.1	database/sql: Postgres Scan Race Condition

- **Calcul du score de priorité**

Le score de priorité est calculé selon la formule suivante :

Score de Priorite´=(CVSS×10×Facteur_Actif)+(EPSS×100)+(KEV×50)

Avec :

- • **CVSS** ∈ [0,10] : mesure la gravité technique intrinsèque de la vulnérabilité.
- • **EPSS** ∈ [0,1] : estime la probabilité d'exploitation dans un contexte réel.
- • **KEV** ∈ {0,1} : indique si la vulnérabilité figure dans le catalogue des vulnérabilités activement exploitées.
- • **Facteur_Actif** : coefficient reflétant la criticité métier de l'actif, déterminée à partir de la cible (*target*) analysée.

Niveau FIPS	Description de l'actif	Facteur appliqué
HIGH	Actif critique (production)	1.5
MODERATE	Actif important	1.0
LOW	Actif de test ou non critique	0.5

Le facteur FIPS est déterminé à partir du **target extrait des rapports Trivy**, permettant d'associer chaque vulnérabilité à son environnement réel.

Cette formule permet de combiner :

- la **gravité technique** (CVSS),
- la **probabilité d'exploitation effective** (EPSS),
- l'**urgence opérationnelle** liée à une exploitation active (KEV).

- **Classification des vulnérabilités**

À partir du score de priorité calculé, chaque vulnérabilité est classée selon les seuils suivants :

Niveau	Seuil du score	Description
P1	priority_score ≥ 80	Critique – action immédiate requise
P2	50 ≤ priority_score < 80	Priorité élevée
P3	priority_score < 50	Priorité modérée

Cette classification permet de hiérarchiser efficacement les vulnérabilités

5.2.2 Délais de correction

De plus Dans ce projet, les délais de traitement sont définis selon une **politique de sécurité opérationnelle** (Service Level Agreements – SLA), appliquée automatiquement par le script Python

Il est important de souligner que le script ne déduit pas ces délais de manière autonome.
Il est conçu pour **appliquer une règle de gestion prédéfinie**, établie par moi

- **Principe général**

Le processus repose sur deux étapes distinctes :

1. **Mesure et catégorisation du risque**
    - Calcul du priority_score à partir des indicateurs CVSS, EPSS et KEV.
    - Conversion de ce score en un niveau de priorité symbolique (P1, P2 ou P3).
2. **Application de la politique de correction**
    - Association automatique d'un délai de correction à chaque niveau de priorité.
    - Génération d'une recommandation opérationnelle via la fonction generate_recommendation.

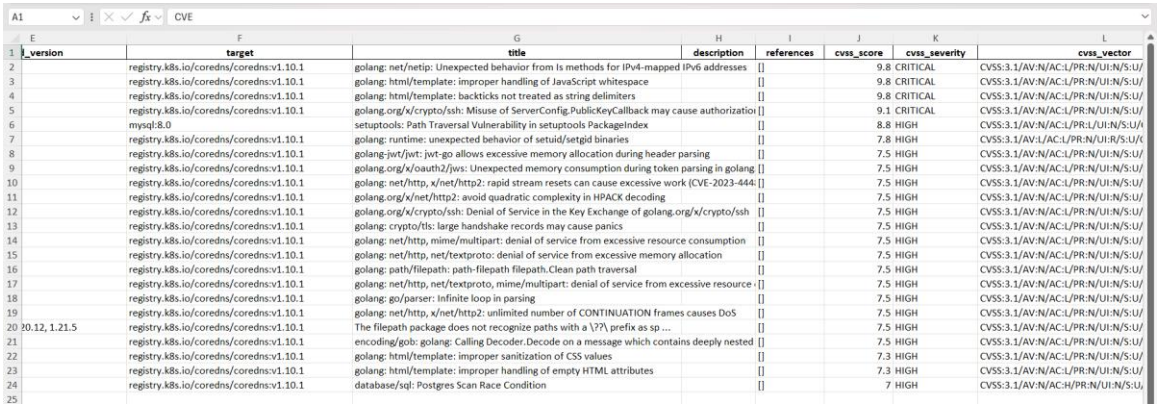
- **Correspondance Priorité – Délais de correction**

Les délais de correction appliqués dans le script sont les suivants :

➢ **Niveau de priorité Délai de correction (SLA) Interprétation opérationnelle**

P1 – Critique	**48 heures maximum**	Correction immédiate requise
P2 – Élevée	**7 à 15 jours**	Correction rapide recommandée
P3 – Modérée	**30 jours**	Correction planifiée

Ces délais sont intégrés dans le rapport final sous forme de recommandations textuelles générées automatiquement par la fonction generate_recommendation.



	E	F	G	H	I	J	K	L
1	l_version	target	title	description	references	cvss_score	cvss_severity	cvss_vector
2		registry.k8s.io/coredns/coredns:v1.10.1	golang: net/netip: Unexpected behavior from Is methods for IPv4-mapped IPv6 addresses		[]	9.8	CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
3		registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: improper handling of JavaScript whitespace		[]	9.8	CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
4		registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: backticks not treated as string delimiters		[]	9.8	CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
5		registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/crypto/ssh: Misuse of ServerConfig.PublicKeyCallback may cause authorizatio		[]	9.1	CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
6		mysql:8.0	setuptools: Path Traversal Vulnerability in setuptools PackageIndex		[]	8.8	HIGH	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/
7		registry.k8s.io/coredns/coredns:v1.10.1	golang: runtime: unexpected behavior of setuid/setgid binaries		[]	7.8	HIGH	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C
8		registry.k8s.io/coredns/coredns:v1.10.1	golang-jwt/jwt: jwt-go allows excessive memory allocation during header parsing		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
9		registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/oauth2/jws: Unexpected memory consumption during token parsing in golang		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
10		registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, x/net/http2: rapid stream resets can cause excessive work (CVE-2023-444		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
11		registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/net/http2: avoid quadratic complexity in HPACK decoding		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
12		registry.k8s.io/coredns/coredns:v1.10.1	golang.org/x/crypto/ssh: Denial of Service in the Key Exchange of golang.org/x/crypto/ssh		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
13		registry.k8s.io/coredns/coredns:v1.10.1	golang: crypto/tls: large handshake records may cause panics		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
14		registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, mime/multipart: denial of service from excessive resource consumption		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
15		registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, net/textproto: denial of service from excessive memory allocation		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
16		registry.k8s.io/coredns/coredns:v1.10.1	golang: path/filepath: path-filepath filepath.Clean path traversal		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
17		registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, net/textproto, mime/multipart: denial of service from excessive resource		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
18		registry.k8s.io/coredns/coredns:v1.10.1	golang: go/parser: Infinite loop in parsing		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
19		registry.k8s.io/coredns/coredns:v1.10.1	golang: net/http, x/net/http2: unlimited number of CONTINUATION frames causes DoS		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
20	20.12, 1.21.5	registry.k8s.io/coredns/coredns:v1.10.1	The filepath package does not recognize paths with a \??\ prefix as sp ...		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
21		registry.k8s.io/coredns/coredns:v1.10.1	encoding/gob: golang: Calling Decoder.Decode on a message which contains deeply nested		[]	7.5	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
22		registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: improper sanitization of CSS values		[]	7.3	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
23		registry.k8s.io/coredns/coredns:v1.10.1	golang: html/template: improper handling of empty HTML attributes		[]	7.3	HIGH	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/
24		registry.k8s.io/coredns/coredns:v1.10.1	database/sql: Postgres Scan Race Condition		[]	7	HIGH	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/
25								

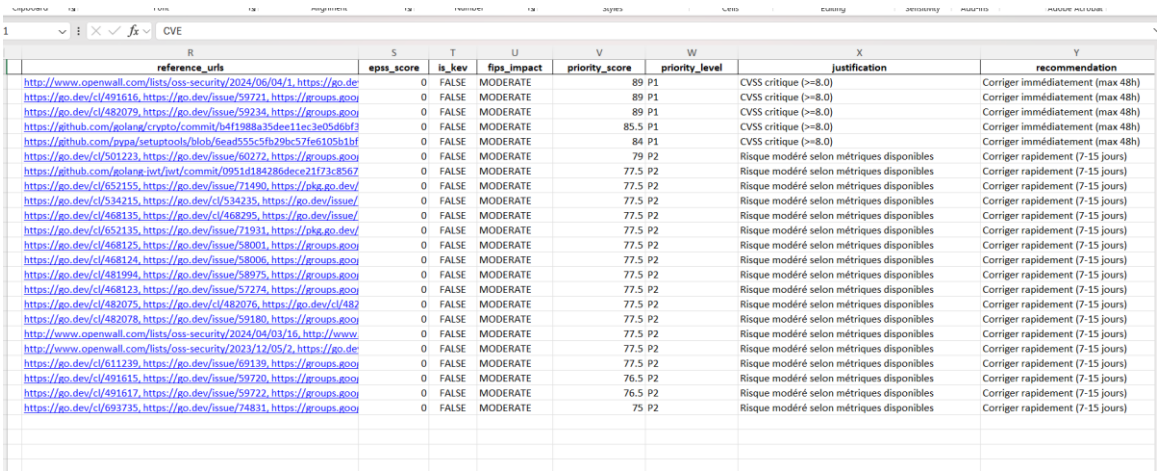
➢ **Justification des délais**

- **P1 (48 heures)**
  Une vulnérabilité classée P1 combine généralement :
  - une gravité élevée (CVSS important),
  - une forte probabilité d'exploitation (EPSS),
  - et/ou une exploitation active connue (KEV).

Ce type de vulnérabilité représente un **risque immédiat et critique**, justifiant une intervention prioritaire sans délai.

- **P2 (7 à 15 jours)**
  Les vulnérabilités P2 présentent un risque élevé mais moins immédiat.
  Elles nécessitent une correction rapide, tout en permettant une planification minimale afin de limiter l'impact opérationnel.

- **P3 (30 jours)**
  Les vulnérabilités P3 sont considérées comme modérées.
  Elles peuvent être intégrées dans un cycle de maintenance standard, sans urgence immédiate.



reference_urls	epss_score	is_kev	fips_impact	priority_score	priority_level	justification	recommendation
http://www.openwall.com/lists/oss-security/2024/06/04/1, https://go.de	0	FALSE	MODERATE	89	P1	CVSS critique (>=8.0)	Corriger immédiatement (max 48h)
https://go.dev/cl/491616, https://go.dev/issue/59721, https://groups.goo	0	FALSE	MODERATE	89	P1	CVSS critique (>=8.0)	Corriger immédiatement (max 48h)
https://go.dev/cl/482079, https://go.dev/issue/59234, https://groups.goo	0	FALSE	MODERATE	89	P1	CVSS critique (>=8.0)	Corriger immédiatement (max 48h)
https://github.com/golang/crypto/commit/b4f1988a35dee11ec3e05d6bf3	0	FALSE	MODERATE	85.5	P1	CVSS critique (>=8.0)	Corriger immédiatement (max 48h)
https://github.com/pypa/setuptools/blob/6ead555c5fb29bc57fe6105b1bf	0	FALSE	MODERATE	84	P1	CVSS critique (>=8.0)	Corriger immédiatement (max 48h)
https://go.dev/cl/501223, https://go.dev/issue/60272, https://groups.goo	0	FALSE	MODERATE	79	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://github.com/golang-jwt/jwt/commit/0951d184286dece21f73c8567	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/652155, https://go.dev/issue/71490, https://pkg.go.dev/	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/534215, https://go.dev/cl/534235, https://go.dev/issue/	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/468135, https://go.dev/cl/468295, https://go.dev/issue/	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/652135, https://go.dev/issue/71931, https://pkg.go.dev/	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/468125, https://go.dev/issue/58001, https://groups.goo	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/468124, https://go.dev/issue/58006, https://groups.goo	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/481994, https://go.dev/issue/58975, https://groups.goo	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/468123, https://go.dev/issue/57274, https://groups.goo	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/482075, https://go.dev/cl/482076, https://go.dev/cl/482	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/482078, https://go.dev/issue/59180, https://groups.goo	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
http://www.openwall.com/lists/oss-security/2024/04/03/16, http://www.	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
http://www.openwall.com/lists/oss-security/2023/12/05/2, https://go.de	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/611239, https://go.dev/issue/69139, https://groups.goo	0	FALSE	MODERATE	77.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/491615, https://go.dev/issue/59720, https://groups.goo	0	FALSE	MODERATE	76.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/491617, https://go.dev/issue/59722, https://groups.goo	0	FALSE	MODERATE	76.5	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)
https://go.dev/cl/693735, https://go.dev/issue/74831, https://groups.goo	0	FALSE	MODERATE	75	P2	Risque modéré selon métriques disponibles	Corriger rapidement (7-15 jours)

VI- Comment tester les différents scripts

6.1 Configurations de la clé d'api

La clé API NVD étant unique et sensible, elle n'est pas incluse dans le dépôt.
Le fichier de configuration est volontairement ignoré via. git ignore.
L'utilisateur doit créer son propre fichier de configuration et y insérer sa clé personnelle.

Le fichier api-config.py est donc volontairement exclu du dépôt (clé API sensible) et ne doit être expose.

Pour exécuter le projet :
1. Créer un fichier api-config.py : nano api-config.py
2. Ajouter votre clé API NVD personnelle avec la même variable :
   NVD_API_KEY = "VOTRE_CLE_ICI"

6.2 Commandes pour exécuter chaque étapes

- Pour enrichir le rapport, se placer a la racine du dossier, installer tous les prérequis
  sudo apt install python3 python3-pip -y
  pip install pandas requests numpy matplotlib seaborn fpdf2 xlsxwriter
  python nvd_client.py -i scan_results.csv -o scan_enriched.csv (pour lancer l'enrichissement)

- Repriorisation et génération des rapports
  python repriorise.py -i scan_enriched.csv -o rapports -f all

- Pour extraire les targets
  python extract_targets.py

Conclusion

Ce projet a permis de mettre en œuvre une approche complète d'analyse et de priorisation des vulnérabilités, allant du scan initial des environnements jusqu'à la génération d'un rapport enrichi et exploitable. L'intégration des données Trivy avec celles de la NVD a apporté une vision plus précise et contextualisée des risques. La méthodologie de priorisation basée sur CVSS, EPSS et KEV permet de dépasser une simple évaluation théorique pour se rapprocher des enjeux opérationnels réels. L'automatisation du processus facilite la prise de décision et améliore l'efficacité des actions de remédiation. Enfin, ce travail illustre l'importance de combiner outils open source, données externes et logique métier dans une démarche de cybersécurité défensive structurée.