# Rapport sur l'Amélioration de l'Analyse des CVE à l'aide des Modèles LLM et la Génération Automatisée des Vecteurs CVSS

***Introduction***

Ce rapport décrit les différentes étapes suivies pour améliorer l'analyse des CVE (Common Vulnerabilities and Exposures) en utilisant des modèles de langage (LLM) afin de générer automatiquement les vecteurs et scores CVSS (Common Vulnerability Scoring System).

L'objectif principal de ce projet était de développer un système robuste capable de stabiliser la génération des vecteurs CVSS tout en exploitant la capacité des LLM pour classifier et enrichir les informations sur les CVE. Nous avons rencontré plusieurs défis, en particulier en ce qui concerne la stabilité des résultats du LLM, mais les itérations et optimisations apportées ont permis d'améliorer la fiabilité du processus.

Lien de Repo Git : https://github.com/alaeddine-hash/cve\_asses.git

***1. Extraction initiale des CVE***

Dans un premier temps, nous avons utilisé la base de données de l'application pour récupérer les CVE directement depuis la base NVD (National Vulnerability Database). À ce stade, un script a été développé afin de permettre la récupération des CVE selon deux paramètres : la date et le nombre de CVE à traiter. Cela nous a permis de structurer la collecte des données de manière contrôlée et progressive.

**Sous le fichier : get\_cve.py**

***2. Prétraitement des Données***

Une fois les CVE récupérées, la deuxième étape consistait à effectuer un prétraitement des données. Ce processus impliquait la suppression de la section CVSS originale des CVE pour ensuite la concaténer avec les informations provenant des *advisory links* (liens de conseils et recommandations). Cette étape a permis de préparer les données pour leur analyse ultérieure par le modèle LLM.

***3. Analyse des CVE et Génération des Vecteurs CVSS avec LLM***

La troisième étape visait à intégrer un modèle LLM dans le pipeline d'analyse afin de générer automatiquement les vecteurs CVSS et leurs scores associés. Pour cela, une boucle *for* a été mise en place pour traiter chaque CVE individuellement, avec pour objectif d'analyser les informations disponibles et de produire un vecteur CVSS conforme aux standards de classification de la vulnérabilité.

Cependant, les premiers tests ont révélé une instabilité dans les résultats fournis par le LLM. Lorsqu'une CVE comportait des informations incomplètes ou peu claires, le modèle générait des résultats incohérents, modifiant fréquemment le vecteur CVSS et son score d'une itération à l'autre. J'ai dû ajuster les *prompts* en plusieurs itérations en ajoutant des formules spécifiques à la tâche pour obtenir des réponses plus précises. Bien que ces ajustements aient permis de stabiliser quelque peu les réponses, des incohérences persistaient, notamment lorsque le modèle générait des vecteurs similaires mais des scores différents, ce qui n'était pas logique.

**Sous le fichier : main\_cve.py**

***4. Stabilisation de la Génération des Vecteurs CVSS***

Dans la deuxième phase de ce projet, j'ai concentré mes efforts sur la stabilisation des vecteurs CVSS générés par le LLM. Après avoir approfondi l’analyse du processus de traitement des CVE, j'ai identifié que la principale cause des incohérences provenait d’un problème de classification des métriques. Le LLM avait des difficultés à identifier correctement certaines catégories et valeurs des métriques.

Pour résoudre ce problème, j’ai exploité la capacité des LLM à classifier et catégoriser les données. En utilisant la même approche logique que dans la première phase, avec des itérations successives de *prompting*, j'ai réussi à stabiliser les vecteurs CVSS sur un ensemble de 100 CVE datées de 2024. Sur cet ensemble, le modèle a modifié 60 vecteurs CVSS, mais les modifications apportées restaient limitées à un ou deux paramètres dans les métriques d’exploitabilité ou d’impact, assurant ainsi une meilleure cohérence globale des résultats.

**Sous le fichier : cvss\_vector.py**

***5. Validation et Analyse Manuelle***

Pour valider les résultats obtenus, j'ai analysé manuellement un échantillon de CVE. Après une analyse approfondie, il est apparu que les modifications proposées par le LLM étaient justifiées et correctes, confirmant ainsi l’efficacité du modèle dans la génération de vecteurs CVSS précis et cohérents.

***6. Enrichissement des CVE avec des Données Supplémentaires***

Dans la troisième phase, j'ai exploré l'utilisation du LLM pour enrichir les CVE en recherchant des informations complémentaires telles que le système d'exploitation (OS) affecté, les versions des logiciels, ainsi que les composants vulnérables. Lorsque toutes les données nécessaires sont disponibles, le LLM s'est révélé particulièrement efficace dans cette tâche. Il a démontré une grande précision et cohérence dans l'enrichissement des CVE, apportant des résultats fiables et exploitables. Le modèle excelle dans ce type de classification et de recherche d'information. Toutefois, dans les cas où les informations manquent, le LLM peut rencontrer des difficultés à fournir des réponses complètes. Cependant, lorsque les données sont complètes, le processus d'enrichissement avec le LLM est optimal et représente une solution robuste pour cette tâche.

***Conclusion***

Ce projet a permis d'améliorer significativement la génération des vecteurs et scores CVSS en exploitant les capacités des modèles de langage LLM. Les résultats obtenus ont démontré une amélioration de la cohérence dans l'analyse des CVE, en particulier après la stabilisation du processus de classification des métriques. Cependant, des défis subsistent concernant la disponibilité des données pour certaines CVE, et des recherches futures pourraient se concentrer sur l'enrichissement des informations disponibles pour optimiser davantage les performances du LLM.

Ce travail met en lumière le potentiel des modèles de langage pour la classification des vulnérabilités, tout en identifiant les domaines où des ajustements sont nécessaires pour améliorer la précision et la stabilité des résultats.