On LLM Embeddings for Vulnerability Management

Rustam Talibzade, Francesco Bergadano, Idilio Drago
University of Turin
firstname.lastname@unito.it

***Abstract*—Effective vulnerability management is fundamental for cybersecurity, requiring significant manual effort to identify, classify, and prioritize reports. Automating this process could reduce analyst workload and improve response to threats. We explore the use of Large Language Models (LLMs) for representing and analyzing Common Vulnerabilities and Exposures (CVEs), specifically their ability to generate semantic embeddings that capture the nature of vulnerabilities from textual descriptions. Using a dataset of 4,710 CVEs, we generate vector embeddings with multiple LLMs. We then apply both unsupervised clustering and supervised classification to evaluate the quality of the embeddings. Our preliminary results show that some LLMs – in particular Llama 3.2 – map similar vulnerabilities together in the embedding space. Embeddings seen to position related vulnerabilities in nearby regions, with certain clusters showing strong correspondence to specific categories like SQL Injection (CWE-89) and Path Traversal (CWE-22). A simple KNN classifier using only the embeddings achieves around 50% accuracy when categorizing CVEs. This is a remarkable result, considering the very high number of classes in the problem. Our initial findings show potential for the use of LLMs in vulnerability management processes, calling for deeper analysis on how to better learn representations from vulnerability reports.**

***Index Terms*—LLM, Cybersecurity, Generative AI.**

I. Introduction

Effective vulnerability management is fundamental for cybersecurity, requiring significant manual effort to identify, classify, and prioritize reported vulnerabilities. Security teams face the challenging task of analyzing vulnerability reports to react to threats in a timely fashion. Automating parts of this process could substantially reduce analyst workload and improve the response to emerging threats.

Recent advances in Large Language Models (LLMs) offer promising opportunities for improving vulnerability management. These models demonstrate impressive capabilities in natural language understanding and processing, potentially enabling the automatic extraction of relevant information from incident reports.

We here take a first step towards understanding the potential of LLMs in this problem domain by focusing on automatically evaluating Common Vulnerabilities and Exposures (CVE) documents. We specifically study the LLM ability to generate semantic embeddings that capture the nature of vulnerabilities from textual CVE descriptions. Our research thus focuses on two questions:

- Can LLMs produce embeddings useful for the representation of CVEs from their textual descriptions?
- Are these embeddings informative for downstream machine learning tasks, such as identifying CVE categories, severity levels, and other meta-information?

To address these questions, we present a preliminary study relying on a dataset of 4,710 CVEs. We first generate vector embeddings using different LLMs, namely Llama, DeepSeek, Mistral and Qwen. We then apply both unsupervised clustering (DBSCAN) and supervised classification (kNN) to evaluate the quality of the embeddings for vulnerability analysis tasks.

Our preliminary results show that LLMs map similar vulnerability types together in the embedding space. The clustering analysis reveals that embeddings group related vulnerabilities, with certain clusters showing strong correspondence to specific vulnerability categories, such as SQL Injection (CWE-89) and Path Traversal (CWE-22). When considering the Common Weakness Enumeration (CWE) as target class for the CVE classification, a simple kNN classifier using only the embeddings achieves around 50% accuracy in CWE identification. Manual analysis shows promising results, with embeddings that seems to group together CVEs describing related vulnerability categories or products.

While not yet sufficient for full automation of vulnerability management, these initial findings demonstrate the potential of LLMs for assisting in vulnerability management processes and supporting security analysts in efficiently categorizing vulnerabilities. Our results call for deeper analysis on how to better learn representations from vulnerability reports.

II. Related Work

Several studies have focused on the use of machine learning for supporting vulnerability management. In particular, benchmarks have been proposed for the development and comparison of new models. CTIBench [1], for example, has been introduced for evaluating LLMs in the domain of Cyber Threat Intelligence (CTI). It includes tasks like CVE-to-CWE classification similar to one of the tasks we use in our analysis. CTIBench also includes tasks such as CVSS score prediction, MITRE ATT&CK technique extraction, and threat actor attribution. These benchmarks show that while models like GPT-4 generally performed best, smaller models like LLAMA3-70B offer competitive results in some tasks.

Authors of [2] discuss using LLMs for vulnerability assessment in medical devices. Their CVE-LLM model generates

978-3-903176-74-4

vulnerability assessments based on CVE descriptions, asset metadata, Common Attack Pattern Enumeration and Classification (CAPEC) and CWEs. The model shows good accuracy.

In contrast to these works, we aim at evaluating embeddings quality instead of proposing benchmarks or evaluating end-to-end classifiers that rely on LLMs.

More broadly, a vast number of recent works focus on the use of LLMs for cybersecurity. In particular, the inspection of source code using LLMs for proactive bug search has received attention from different authors.

In [3] authors looked at 16 LLMs and tested them across a mix of Java and C/C++ datasets with 25 CWEs. LLMs detect better the vulnerabilities that need intra-procedural reasoning, outperforming popular static analysis tools like CodeQL. Even when LLMs make wrong predictions, they can point out relevant sources, sinks, and sanitizers, helping follow-up human analysis. The problem of identifying vulnerability in code is also approached in [4]. This work proposes VFFinder, a tool that uses in-context learning to predict attack vectors and root causes from CVE descriptions. VFFinder matches descriptions against the source code using semantic retrieval, reporting promising results.

In an opposite direction, authors of [5] argue that LLMs are not yet reliable for vulnerability detection. This work presents a framework that evaluates LLMs with respect to their ability to identify security-related bugs. Authors point out high false positive rates by LLMs: even when the target systems are already patched, LLMs still flag systems as vulnerable. Sometimes, despite identifying the vulnerabilities correctly, the reasoning behind the decision is flawed. LLMs get confused by small changes like addition or removal of spaces and minor function name changes.

Our work shares goals with those efforts, targeting the evaluation of LLMs for cybersecurity application. We however focus on a first step towards understanding how cybersecurity knowledge is intrinsically encoded in models pre-trained with general textual data.

III. Methodology

A. Data Collection

We use the National Vulnerability Database (NVD) maintained by the National Institute of Standards and Technology (NIST) to obtain CVE documents. The NVD is a repository of standardized vulnerability information that contains CVE descriptions, vulnerability metrics, and impact ratings. A subset of 4710 CVEs reported in 2022 is used in this preliminary study. As a cleanup step, we consider only CVEs reported by NIST with vulnerability statuses *Analyzed* or *Modified*. The rationale is to avoid CVE descriptions coming from different cybersecurity teams. Our manual analysis suggests that the writing style of CVEs is strongly dependent on the team reporting the CVE. As an additional cleanup step, we ignore all CVEs associated with the CWE *NVD-CWE-noinfo*. CVEs in this category are usually vague and do not give much information about the vulnerability type even for human experts.

TABLE I: Models used for embeddings creation. [6]

Model	Parameters	Embeddings Size	Context Length
Llama 3.1	8B	4096	131072
Llama 3.2	3.2B	3072	131072
Llama 3.3	70.6B	8192	131072
Qwen 2.5	7.6B	3584	32768
QwQ	32.8B	5120	131072
Mistral	7.2B	4096	32768
DeepSeek-R1	7.6B	3584	131072

B. Embedding Creation and LLM Models

Our main goal is to evaluate how LLMs encode CVE descriptions. We are particularly interested in understanding whether LLM embeddings capture more advanced cybersecurity concepts beyond simple natural language patterns. We select 7 LLMs for this preliminary work which are summarized in Table I.

We selected popular open-source LLMs with variable parameter counts for our evaluation. As shown in Table I, all models have billion parameters and produce embeddings of similar dimension. All models offer an extended context length (32768 and 131072 tokens), which is largely sufficient to keep the complete CVE description in the model context. These models have been chosen because they represent widely-used models that can be run locally, allowing for efficient embedding generation. For the sake of space, we will describe results for the Llama 3.2 model in the following, as this model has proven to deliver excellent results. A comprehensive comparison of all models will be presented in our future work.

C. Downstream Tasks

To evaluate the quality of embeddings, we define two different downstream tasks related to cybersecurity: an unsupervised clustering problem and a supervised classification task.

First, we apply DBSCAN clustering to the various LLM embeddings. The same parameter setting is used for all LLMs. To determine optimal cluster formation, we select an epsilon value of 0.1 using the elbow method, checking the several independent clustering results simultaneously. The resulting clusters are then manually analyzed for their correspondence to similar CWEs, involved products, and other aspects to assess how well the embeddings group similar vulnerabilities without labels.

For our second task, we explore the classification of CVEs using CWE categories as in previous work [1]. Using the embeddings generated by the selected LLMs, we verify whether CVEs belonging to the same CWE are mapped in a similar embedding space region by setting up a K-Nearest Neighbors (KNN) classification problem. We set up a leave-one-out validation experiment, in which each CVE embedding is compared to all others to obtain the nearest neighbor CVEs. The CWEs of the K-neighboring CVEs are used as prediction labels. We evaluate performance using multiple K values (K=1, K=3, and

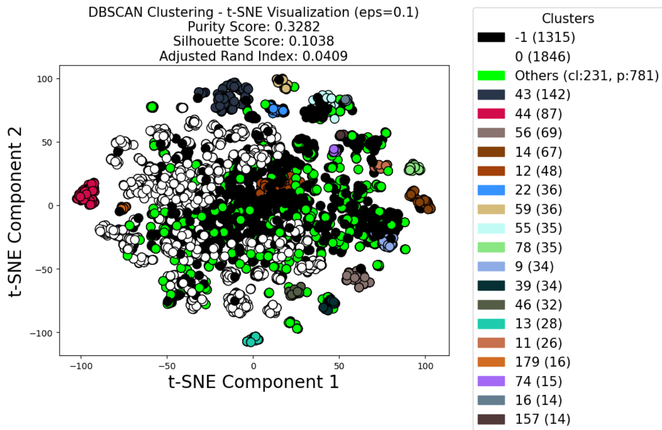

Fig. 1: DBSCAN clustering with Llama3.2 embeddings – the legend reports the cluster ID, and cluster cardinality (between brackets). Label -1 represents unclustered points; top-20 cluster are shown, others are combined in ”Others”.

TABLE II: Performance metrics for the 5-KNN classifier with Llama3.2 embeddings (Top-1 accuracy).

Metric	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Macro avg	20.14	15.97	16.59
Weighted avg	47.90	49.15	47.41
Accuracy	49.15		

K=5) and report both Top-1 and Top-2 accuracy metrics. Top-1 accuracy measures correct classification of the exact CWE, while Top-2 accuracy considers whether the correct CWE appears among the top two predictions.

IV. Preliminary Results

Figure 1 presents a visualization of clustering obtained with DBSCAN for the Llama 3.2 embeddings. Since the embeddings are high-dimensional, we use t-SNE for visualization which helps to project the embeddings into two dimensions while preserving the structure of the data. DBSCAN divides our subset of CVEs to more than 250 clusters – only the top-20 clusters are shown in the figure; -1 are the unclustered points. We then manually investigate the cluster in the search for possible explanations for they compositions.

Some clusters appear to include CVEs related to the same software product, whereas some other clusters are related to the type of vulnerability (CWE). For example, cluster 43 (dark blue) which is situated on top center in the t-SNE plot has all but one point corresponding to CWE-89 (SQL Injection), the exception belonging to CWE-79 (Cross-site Scripting), which is also a vulnerability related to web products. Cluster 44 (red) situated on the left center has all points belonging to CWE-22 (Path Traversal). Clusters 11, 39, 46 contain CVEs with CWE-787 (Out-of-bounds Write) and CWE-125 (Out-of-bounds Read), which is an interesting observation considering the similarity of vulnerabilities. Thus, their descriptions share

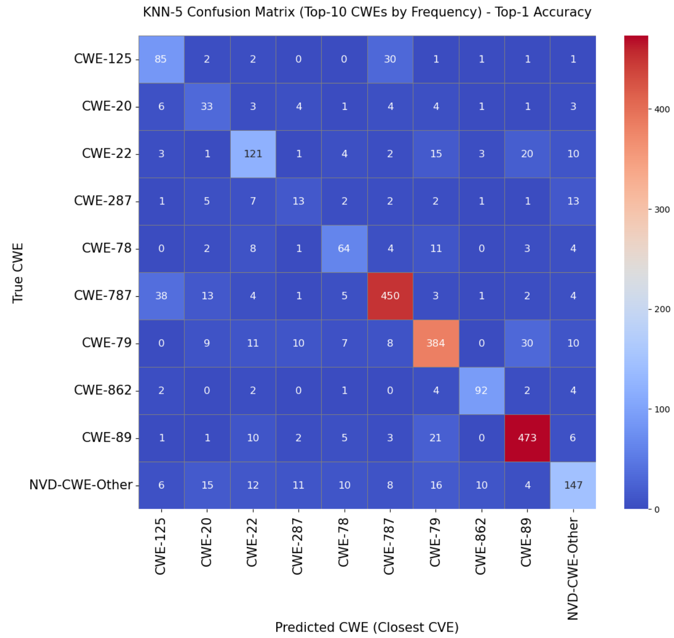

Fig. 2: Confusion Matrix of 5-KNN for Llama3.2 embeddings (Top-1 accuracy) – top-10 CWEs.

multiple terms and the embeddings appear to capture those similarities.

While not exhaustive, our manual analysis has identified clusters that group specific types of vulnerabilities as well as groups affecting the same product. In contrast to our initial intuition, we hardly find clusters that aggregate CVEs of similar severity level.

Table II reports the results of using KNN with 5 neighbors and Top-1 accuracy. We reach 49.15% accuracy. It is worth noticing that there are 125 different CWEs in this data sample, which we consider separate classes. As such, the classification presents somewhat promising results, even if accuracy is yet low. Figure 2 confirms that by reporting the confusion matrix of the same method for the 10 most popular CWEs. Most popular CWEs are CWE-787, CWE-79 and CWE-89. Notice here how CWE-787 (Out-of-bounds Write) and CWE-125 (Out-of-bounds Read) concentrate several classification mistakes, in line with conclusions of the DBSCAN analysis. Similar patterns appear for other less popular classes, not shown in the figure for brevity.

All in all, whereas the embeddings alone are not yet sufficient for building a precise classification pipeline, they seem to capture semantics of the description that are relevant for cybersecurity tasks.

V. Concluding Remarks

This work is an initial step towards understanding how cybersecurity concepts are encoded in LLM models. Our short-term goal is improving the comparison of LLMs for picking up the best representations. These representations will then be used to tackle downstream tasks, such as automatic CVE evaluation and identification of vulnerabilities in source code.

ACKNOWLEDGMENTS

This project was funded by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU, specifically by the cascate call project Q-CPS2: Quantitative models for Cyber Physical Systems Security. This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

REFERENCES

[1] M. T. Alam, D. Bhusal, L. Nguyen, and N. Rastogi, "Ctibench: A benchmark for evaluating llms in cyber threat intelligence," *arXiv preprint arXiv:2406.07599*, 2024.

[2] R. Ghosh, H.-M. von Stockhausen, M. Schmitt, G. M. Vasile, S. K. Karn, and O. Farri, "Cve-llm: Ontology-assisted automatic vulnerability evaluation using large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 28757–28765, 2025.

[3] A. Khare, S. Dutta, Z. Li, A. Solko-Breslin, R. Alur, and M. Naik, "Understanding the effectiveness of large language models in detecting security vulnerabilities," *arXiv preprint arXiv:2311.16169*, 2023.

[4] Y. Wu, M. Wen, Z. Yu, X. Guo, and H. Jin, "Effective vulnerable function identification based on cve description empowered by large language models," in *2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 393–405, 2024.

[5] S. Ullah, M. Han, S. Pujar, H. Pearce, A. Coskun, and G. Stringhini, "Llms cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 862–880, 2024.

[6] Ollama Inc., "Ollama: Run and customize large language models locally." https://ollama.com/, 2025. Accessed: 2025-04-28.