A Study on Vulnerability Scoring using Large Language Models

Ing. Andrés Tito
M.Sc Com. and Electronics
Technische Universität München
Munich, Germany
andres.tito@tum.de

Dr.-Ing. Mohammad Hamad
Leader, Security for IoT and Automotive Systems Group
Technische Universität München
Munich, Germany
mohammad.hamad@tum.de

Dipl.-Inf. Lukas Braune
Senior Product Security Expert
Rohde & Schwarz GmbH&Co. KG
Munich, Germany
lukas.braune@rohde-schwarz.com

***Abstract*—Automating vulnerability scoring presents a significant challenge in cybersecurity, where accurate assessments are essential for prioritizing risks and allocating resources effectively. Traditional methods, often limited by delayed responses and inaccuracies, hinder timely evaluations. This study investigates the feasibility of using pre-trained Large Language Models (LLMs) to generate scores following the Common Vulnerability Scoring System (CVSS), drawing on structured vulnerability data such as Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE). We evaluate the accuracy and reliability of LLM-generated scores by benchmarking them against established databases, including the National Vulnerability Database (NVD), GitHub Advisory, and RedHat. Findings reveal that, although LLMs can contribute to the scoring process, notable discrepancies remain between model outputs and official records, underscoring limitations in the models' reliability. This study highlights the importance of integrating automated LLM technologies with traditional assessment methodologies to enhance both the comprehensiveness and dependability of vulnerability evaluations in cybersecurity.**

***Index Terms*—Vulnerability scoring, Large Language Models, CVSS, CVE**

I. Introduction

Vulnerability scoring is a complex and often tedious task requiring extensive knowledge of the FIRST Common Vulnerability Scoring System (CVSS) [1] standard and the specific characteristics of the vulnerable system. In the cybersecurity industry, it is common practice to rely on scores and information provided by the National Vulnerability Database (NVD) due to the simplicity and accessibility of its database [2]. However, it is well-known that the NVD has certain limitations in its reports, leading to inaccurate assessments when automating analysis processes [3] [4].

One significant issue is the response time of the NVD, which may be slow in reacting to new vulnerabilities and releasing corresponding information. For organizations, having a CVSS score available as quickly as possible is crucial to begin mitigation efforts [5].

While previous research has explored the use of Large Language Models (LLMs) in tasks like vulnerability detection and code analysis [6] [7], few studies have focused on vulnerability scoring itself [8]. This research aims to fill that gap by investigating the feasibility of using pre-trained LLMs to automate the scoring of security vulnerabilities using descriptive parameters such as CVE data and CWE tags. By comparing the results with established systems like CVSS from sources such as NVD, GitHub Advisory (GitHub), and RedHat, we aim to evaluate the accuracy of LLM-based scoring and assess how well these models perform in generating reliable vulnerability scores.

II. Methodology

A. Large Language Models Selection

For this study, we utilized two specific Large Language Models (LLMs): GPT-4o-mini [9] and Llama-8b [10]. These models were chosen due to their availability and low cost, making them suitable for our budget constraints.

B. Datasets

We employed three primary datasets for evaluating the vulnerability scoring models: the National Vulnerability Database (NVD) CVE dataset, the GitHub dataset [11], and the RedHat dataset [12]. The NVD dataset contains over 100,000 CVEs, while the GitHub dataset includes approximately 18,000 CVEs. The RedHat dataset was also incorporated to provide additional data points. We removed CVEs that lacked descriptions, CWEs, or scores during preprocessing. It is important to note that this research aims to evaluate pre-trained general LLMs on scoring vulnerabilities rather than training the LLMs from scratch.

C. Evaluation Metrics

The performance of GPT-4o-mini and Llama-8b was evaluated using the following criteria and metrics:

- **Mean Deviation**: The average difference between the LLM-generated and original scores.

$$MD = \frac{1}{n}\sum_{i=1}^{n}|x_i - \mu|$$

where x_i represents the individual LLM-generated score, μ is the original score, and n is the total number of scores.

- **Mean Deviation Percentage**: The average percentage difference between the LLM-generated and original scores.

$$MDP = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|x_i - \mu|}{\mu} \times 100\right)$$

where x_i is the LLM-generated score, μ is the original score, and n is the total number of scores.

- **Accuracy**: Measured in three ways:
  1) **Severity Level Accuracy**: Whether the severity level assigned by the LLM matches the original severity level (as shown in Table I), which categorizes vulnerabilities into None, Low, Medium, High, or Critical levels [13].
  2) **Action Needed Accuracy**: Whether the LLM correctly identifies if the original score is higher or lower than 7.
  3) **Threshold-Based Accuracy**: Whether the LLM-generated score falls within a specified threshold (e.g., 2 points) of the original score.

TABLE I
CVSS SEVERITY LEVELS

Severity Level	CVSS Score Range
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

D. Experimental Setup

The experiment is conducted within a virtual Python environment designed to integrate multiple components for data collection, processing, and evaluation.

Data is gathered from various CVE sources using specialized scripts tailored to each source's specific API formats. These scripts extract relevant vulnerability information in a structured format for further analysis.

To interact with the LLM, the script chat.py handles the prompt generation and LLM requests. This script includes key tasks such as adding system prompts, setting optimization parameters, and crafting the input messages for the LLM. It also extracts essential outputs from the LLM's responses, including the base CVSS score and the corresponding vector.

The data cleansing and experimental analysis are carried out using a Jupyter notebook, where data is preprocessed and prepared for evaluation.

For data visualization, the script frontend.py is employed. This script assumes that the LLM queries and the CVE source data collection have been completed. It presents the evaluation metrics—such as mean deviation, mean deviation percentage, and accuracy—in a user-friendly format, making it easier to interpret the results.

E. Evaluation Process

The evaluation process was conducted using Python and the Pandas library. The methodology involved the following steps:

1) **System Prompt Crafting**: A system prompt was crafted to instruct the LLMs on their task: *"You are an expert in CVSS 3.1 scoring. Provide the correct base score and a valid CVSS 3.1 vector. Ensure the vector is complete, well-formed, and adheres to the CVSS 3.1 standard."*
2) **Parameter Optimization**: Specific LLM parameters were adjusted to ensure deterministic answers. The temperature was set to 0.1, and the top_p parameter was set to 0.9. Additionally, since only one answer was expected, the parameter "n" was set to 1.
3) **Data Input**: The CVE ID, description, and CWE of each CVE were provided to the LLMs.
4) **LLM Output**: The LLMs generated vectors and scores based on the provided information.
5) **Score Calculation**: The LLM-generated vectors were input into a CVSS Calculator to obtain the scores, then compared against the original scores.
6) **Data Merging**: The LLM results were merged with the original score sources to measure deviation and accuracy.

F. Challenges and Limitations

Several challenges and limitations were encountered during the study:

- **Score Discrepancies**: Scores varied depending on the source, with NVD and RedHat sometimes providing different scores for the same CVE [4].
- **Vector Compatibility**: The vectors and scores provided by the LLMs were not always compatible, leading us to use only the vectors and calculate the scores separately.
- **Vector Validity**: Some LLM-generated vectors were invalid in format, requiring additional validation steps.
- **Model Constraints**: Budget constraints limited us to using smaller models such as GPT-4o-mini and Llama-8b.

III. EXPERIMENTAL RESULTS

This section presents the results of our experiments using the GPT-4o-mini and Llama-8b models for vulnerability scoring. We evaluate the models based on the metrics described in the Methodology section: mean deviation, mean deviation percentage, and accuracy. Additionally, we compare the results across different score sources: NVD Primary Source, NVD Secondary Source, GitHub, and RedHat.

A. Single Case Scenarios

To provide more illustrative insights into the performance of the models, we evaluate specific CVEs, comparing their scores across multiple sources. This helps identify inconsistencies and understand how LLMs perform compared to traditional sources. Additionally, we explore potential reasons behind score variations.

1) Inconsistencies in CVE-2020-0779: Different models and CVE sources are expected to generate varying scores, even when adhering to a standard such as CVSS [4]. However, discrepancies between major sources can be significant in some cases, and LLM-generated scores further complicate the evaluation.

For instance, when analyzing CVE-2020-0779, we observe differing scores from two major vulnerability databases, NVD [14] and RedHat [15]:

”A heap-based buffer overflow in coders/tiff.c may result in program crash and denial of service in ImageMagick before 7.0.10-45.” CWE-122: Heap-based Buffer Overflow

TABLE II
CVE-2020-0779 CVSS3.1 SCORES

Source	Base Score	Vector
NVD	5.5	AV:L/**AC:L**/PR:N/UI:R/S:U/**C:N/I:N**/A:H
Red Hat	7	AV:L/**AC:H**/PR:N/UI:R/S:U/**C:H/I:H**/A:H

The LLM-generated scores for the same vulnerability exhibit even more variation:

TABLE III
CVE-2020-0779 LLM CVSS3.1 SCORES

Source	Base Score	Vector
gpt-4o-mini	5.1	AV:L/**AC:H/PR:N**/UI:N/S:U/**C:N/I:H/A:N**
Llama-8b	4.4	AV:L/**AC:L/PR:H**/UI:N/S:U/**C:N/I:H/A:N**

In this case, Red Hat assigns a significantly higher CVSS score of 7.0, indicating a high impact on confidentiality, integrity, and availability. This assessment seems to overestimate the vulnerability’s potential impact, particularly in ImageMagick, where the primary risk is a denial of service. NVD’s score of 5.5 focuses solely on availability, which appears more accurate.

The LLM-generated results further diverge. GPT-4o-mini assigns a score of 5.1, which aligns more closely with NVD’s evaluation, but fails to acknowledge the availability impact as a major concern. Llama-8b’s score of 4.4 introduces additional inconsistencies, such as requiring privileges to exploit the vulnerability—an assertion not supported by the NVD or RedHat vectors. Moreover, neither LLM properly identifies availability as a significant impact, which is critical for this CVE. This misjudgment highlights the challenges of relying on LLMs for vulnerability scoring, as their assessments can deviate significantly from trusted sources, especially in nuanced cases.

2) Consistencies in CVE-2014-0160: Conversely, when evaluating well-known vulnerabilities such as CVE-2014-0160 (Heartbleed) [16], the LLM-generated results are much more consistent and accurate, reflecting the high quality of the training data for widely recognized vulnerabilities.

TABLE IV
CVE-2014-0160 CVSS3.1 SCORES

Source	Base Score	Vector
NVD	7.5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
gpt-40-mini	7.5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Llama-8b	7.5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

In this instance, both LLM models perfectly match the CVSS score and vector provided by the NVD, demonstrating that their performance improves significantly when the models have access to comprehensive training data for well-documented vulnerabilities. This consistency likely stems from the widespread inclusion of Heartbleed-related data in cybersecurity datasets used for LLM training. Therefore, while LLMs can struggle with lesser-known or more nuanced vulnerabilities, they perform well when evaluating widely recognized issues.

B. Performance Evaluation of LLM-Generated Vulnerability Scores Using NVD as a Benchmark

We will start with the results obtained from the National Vulnerability Database (NVD), which serves as one of the primary sources of vulnerability data. NVD’s comprehensive and authoritative dataset is widely regarded in the cybersecurity community, making it an essential benchmark for evaluating Large Language Models (LLMs) in vulnerability scoring. Furthermore, our analysis revealed that NVD-based scores align more closely with LLM-generated outputs than other sources, such as RedHat and GitHub, further solidifying its central role in this study.

1) Mean Deviation and Mean Deviation Percentage: The mean deviation and mean deviation percentage were calculated to measure the average difference and average percentage difference between the LLM-generated scores and the original scores.

TABLE V
MEAN DEVIATION AND MEAN DEVIATION PERCENTAGE

Model	Mean Deviation	Mean Deviation Percentage
GPT-4o-mini	1.076	16.78%
Llama-8b	1.314	20.21%

As shown in Table V, GPT-4o-mini achieved a mean deviation of 1.076 and a mean deviation percentage of 16.78%, while Llama-8b had a mean deviation of 1.314 and a mean deviation percentage of 20.21%.

2) Accuracy: The accuracy of the models was evaluated in three ways: severity level accuracy, action needed accuracy, and threshold-based accuracy.

a) Severity Level Accuracy: Severity level accuracy measures whether the LLM-assigned severity level matches the original severity level.

TABLE VI
SEVERITY LEVEL ACCURACY

Model	Severity Level Accuracy
GPT-4o-mini	59.6%
Llama-8b	52.7%

Table VI shows that GPT-4o-mini achieved a severity level accuracy of 59.6%, while Llama-8b achieved 52.7%.

b) Action Needed Accuracy: Action needed accuracy measures whether the LLM correctly identifies if the original score is higher or lower than 7.

As shown in Table VII, GPT-4o-mini achieved an action-needed accuracy of 75.9%, while Llama-8b achieved 71.8%.

TABLE VII
ACTION NEEDED ACCURACY

Model	Action Needed Accuracy
GPT-4o-mini	75.9%
Llama-8b	71.8%

TABLE VIII
THRESHOLD-BASED ACCURACY (THRESHOLD = 2)

Model	Threshold-Based Accuracy
GPT-4o-mini	81.8%
Llama-8b	73.4%

c) Threshold-Based Accuracy: Threshold-based accuracy measures whether the LLM-generated score falls within a specified threshold (e.g., 2 points) of the original score.

Table VIII shows that GPT-4o-mini achieved a threshold-based accuracy of 70.3% when the threshold was set to 2, while Llama-8b achieved 73.4%.

C. Comparison Across Different Score Sources

To further evaluate the performance of the models, we compared the results across different score sources: NVD, GitHub, and RedHat.

TABLE IX
MEAN DEVIATION AND MEAN DEVIATION PERCENTAGE BY SOURCE

Source	Model	Mean Deviation	Mean Deviation %
NVD	GPT-4o-mini	1.076	16.78%
NVD	Llama-8b	1.314	20.21%
GitHub	GPT-4o-mini	1.308	22.41%
GitHub	Llama-8b	1.505	26.54%
RedHat	GPT-4o-mini	1.699	36.34%
RedHat	Llama-8b	1.609	30.61%

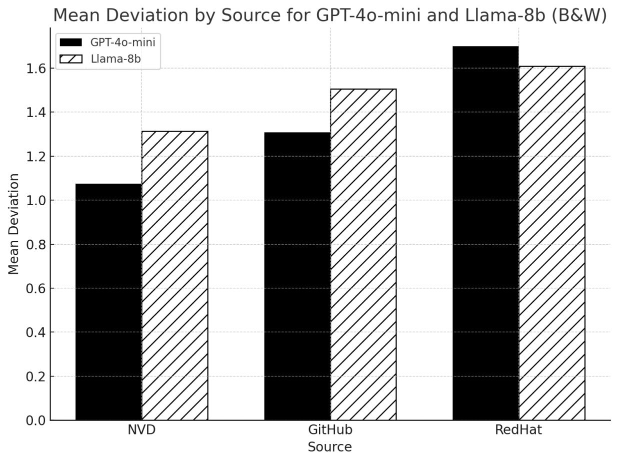

Fig. 1. Mean Deviation and Mean Deviation Percentage by Source

1) Mean Deviation and Mean Deviation Percentage by Source: Table IX and fig. 1 shows each model's mean deviation and mean deviation percentage across different score sources. GPT-4o-mini consistently outperformed Llama-8b across every source except for Red Hat.

TABLE X
ACCURACY BY SOURCE

Source	Model	Severity lvl	Action Needed	Threshold
NVD	GPT-4o-mini	59.6%	75.9%	81.8%
NVD	Llama-8b	52.7%	71.8%	73.4%
GitHub	GPT-4o-mini	53.8%	72.3%	71.6%
GitHub	Llama-8b	49.2%	69.3%	66.1%
RedHat	GPT-4o-mini	40.9%	58.6%	64.3%
RedHat	Llama-8b	43.7%	64.8%	67.7%

2) Accuracy by Source:

D. Discussion of Results

The results indicate that GPT-4o-mini and Llama-8b perform reasonably well in scoring vulnerabilities. GPT-4o-mini slightly outperforms Llama-8b in all metrics. However, both models show room for improvement, particularly in generating valid CVSS vectors.

The discrepancies in scores depending on the source (NVD, GitHub, RedHat) [17] and the occasional invalid vectors generated by the LLMs highlight some of the challenges faced in this study. Despite these challenges, the models demonstrated an excellent ability to approximate vulnerability scores, suggesting that LLMs can be valuable in vulnerability assessment.

IV. CONCLUSION

This research evaluated different LLMs for vulnerability scoring by comparing their performance against various CVE sources, including NVD, GitHub Advisory, and RedHat.

While LLMs show great potential for automating vulnerability scoring, our results indicate that their current accuracy levels must be improved for such a critical task. Given the high stakes in vulnerability assessment, even minor inaccuracies can have significant consequences, making relying solely on LLMs in this domain unfeasible.

To address this, we recommend a hybrid approach, where LLMs dispatch the scoring process while human experts provide oversight to ensure precision. Another possible direction for improvement is fine-tuning LLMs with domain-specific scoring data to improve their accuracy in vulnerability assessment.

ACKNOWLEDGMENTS

We want to express our sincere gratitude to Rohde & Schwarz for their invaluable support in this research. Their infrastructure provision, including access to the Llama-8b on-site server and comprehensive LLM documentation, was instrumental in our work. We also appreciate their expertise and consultations, which significantly contributed to the success of this project.

REFERENCES

[1] FIRST, "CVSS v3.1 user guide," 2023. [Online]. Available: https://www.first.org/cvss/v3.1/user-guide.
[2] National Institute of Standards and Technology, "National Vulnerability Database," https://nvd.nist.gov/, accessed: 2024-10-18.

[3] Afsah Anwar, Ahmed Abusnaina, Songqing Chen, Frank Li, and David Mohaisen, “Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 4255-4269, 2022. doi: 10.1109/TDSC.2021.3125270.

[4] Red Hat, “Differences Between NVD and Red Hat Scores,” *Red Hat Security Updates*, n.d. [Online]. Available: https://access.redhat.com/security/updates/classification/. [Accessed: 2024-10-18].

[5] National Institute of Standards and Technology, “Vulnerability Mitigation Time Summary Matrix (Table 1),” in *Guide to Enterprise Patch Management Technologies*, NIST Special Publication 800-40 Rev. 4, 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r4.pdf. [Accessed: 2024-10-18].

[6] R. I. T. Jensen, V. Tawosi, and S. Alamir, “Software vulnerability and functionality assessment using LLMs,” arXiv preprint arXiv:2403.08429, 2024. [Online]. Available: https://arxiv.org/abs/2403.08429.

[7] Y. Liu, L. Gao, M. Yang, Y. Xie, P. Chen, X. Zhang, and W. Chen, “VulDetectBench: Evaluating the deep capability of vulnerability detection with large language models,” arXiv preprint arXiv:2406.07595, 2024. [Online]. Available: https://arxiv.org/abs/2406.07595.

[8] X. Yin, C. Ni, and S. Wang, “Multitask-based evaluation of open-source LLM on software vulnerability,” arXiv preprint arXiv:2404.02056, 2024. [Online]. Available: https://arxiv.org/abs/2404.02056.

[9] OpenAI, “GPT-4o-mini: Advancing cost-efficient intelligence,” 2024. [Online]. Available: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

[10] Meta, “Llama 3.1: 8b,” 2024. [Online]. Available: https://ollama.com/library/llama3.1:8b.

[11] GitHub, “GitHub security advisories,” 2024. [Online]. Available: https://github.com/advisories.

[12] Red Hat, “Red Hat CVE security updates,” 2024. [Online]. Available: https://access.redhat.com/security/security-updates/cve.

[13] FIRST, “CVSS v3.1 specification document,” 2024. [Online]. Available: https://www.first.org/cvss/v3.1/specification-document#:~:text=5.%20Qualitative%20Severity%20Rating%20Scale.

[14] NIST, “CVE-2020-27829: Heap-based Buffer Overflow in ImageMagick,” National Vulnerability Database, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-27829. [Accessed: 2024-10-18].

[15] Red Hat, “CVE-2020-27829: Heap-based Buffer Overflow in ImageMagick,” Red Hat Security, 2020. [Online]. Available: https://access.redhat.com/security/cve/CVE-2020-27829. [Accessed: 2024-10-18].

[16] NIST, “CVE-2020-27829: Heartbleed,” National Vulnerability Database, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2014-0160. [Accessed: 2024-10-20].

[17] J. Wunder, A. Kurtz, C. Eichenmüller, F. Gassmann, and Z. Benenson, “Shedding Light on CVSS Scoring Inconsistencies: A User-Centric Study on Evaluating Widespread Security Vulnerabilities,” in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 1102-1121, 2024. doi: 10.1109/SP54263.2024.00058.