Context-Aware Vulnerability Management Using Large Language Models

AnmolikaSingh*
Data Scientist
Independent Researcher
singh.anmolika@gmail.com

Mojtaba Alfardan*
DevOps Engineer
Independent Researcher
alfardanmojtaba@gmail.com

*Both authors have had an equal contribution

***Abstract*—Organizations are frequently overwhelmed by the sheer volume of alerts about vulnerabilities discovered within their systems. These alerts are typically prioritized based on severity levels categorized by Common Vulnerabilities and Exposures (CVE) [2], a standard glossary used in Vulnerability Management Systems. However, this severity classification often fails to consider the specific operational context of the systems, leading to misaligned priorities and the potential oversight of more critical vulnerabilities that demand immediate attention. This paper investigates whether Large Language Models (LLMs)[25] can offer a solution by integrating contextual awareness into the vulnerability management process, thus enhancing the efficiency and effectiveness of organizational responses to cybersecurity threats.**

Keywords: Common Vulnerabilities and Exposures (CVE), Vulnerability Management System (VMS), Large Language Models (LLM), Cybersecurity, Severity Classification

I. Introduction

VULNERABILITY management remains a cornerstone of strategic cybersecurity practices within organizations globally. It involves the identification, prioritization, and mitigation of software vulnerabilities that emerge due to failures or flaws in computer systems, networks, databases, and applications. These vulnerabilities, if exploited, can lead to significant losses—both financial and reputational [26]. Effective management of these vulnerabilities is thus crucial to maintaining the security and integrity of organizational IT infrastructure.

A. The Nature of Software Vulnerabilities

Software vulnerabilities are essentially errors or flaws in the source code of applications and systems that can be exploited by attackers to gain unauthorized access or cause harm [8]. These vulnerabilities vary widely in their nature and severity, ranging from minor issues that have little impact on system functionality to critical flaws that can compromise entire networks or data integrity. The process of identifying and addressing these vulnerabilities forms the core of vulnerability management practices.

B. Challenges in Vulnerability Management

Despite the critical importance of managing vulnerabilities effectively [26], organizations face several challenges in this domain:

1) Volume and Complexity The sheer number of vulnerabilities discovered in software systems can be overwhelming [19]. Each day, new vulnerabilities are identified, adding to the already extensive list of issues that need to be addressed.
2) Prioritization Determining which vulnerabilities pose the greatest risk and should therefore be prioritized is a complex task. Traditional vulnerability management systems utilize scoring systems like the Common Vulnerability Scoring System (CVSS) [5] to assign severity levels to vulnerabilities. However, these systems often lack the context-specific insight needed to accurately assess the actual risk posed by each vulnerability in an operational environment.
3) Resource Allocation Given the high volume of vulnerabilities and the varying levels of threat they pose [19], efficiently allocating resources to address these vulnerabilities is a significant challenge. Organizations must balance the need to fix critical vulnerabilities quickly with the practical limitations of their Cybersecurity resources.
4) Generic Assessments Many vulnerability management systems provide assessments that are too generic. These systems typically assign a 'High' severity level to a large number of vulnerabilities [9], which can lead to inefficient prioritization. As a result, less critical vulnerabilities may receive undue attention, while more critical ones may be overlooked.
5) Manual Efforts and Human Error The need for further manual prioritization due to the generic nature of assessments introduces human error and delays. Manual review processes are not only time-consuming but also prone to mistakes, as security teams must make quick decisions under pressure.

C. The Role of Large Language Models in Vulnerability Management

In response to these challenges, this paper proposes the use of Large Language Models (LLMs) [25] as a novel approach to enhance the vulnerability management process. LLMs, such as OpenAI's GPT-4 [14], represent a significant advancement in artificial intelligence and natural language processing technologies. These models have shown remarkable capabilities in understanding and generating human-like text based on the input provided to them.

The hypothesis driving this research is that LLMs can be trained and utilized to provide context-aware assessments of vulnerabilities. By understanding the specific context in which a software system operates, LLMs could potentially offer more precise vulnerability prioritization, thereby reducing the reliance on generic severity scores and minimizing the need for manual intervention.

D. Objectives

The primary objectives of this paper are to:

1) Explore the feasibility of using LLMs to improve the accuracy and efficiency of vulnerability assessments in organizational IT environments.
2) Evaluate the performance of LLMs in identifying and prioritizing vulnerabilities based on contextual relevance rather than generic severity scores.
3) Assess the potential of LLMs to reduce manual efforts and human error in the vulnerability management process.

II. Literature Review

A. Introduction to Current Vulnerability Scoring Systems

Vulnerability management is a pivotal aspect of cybersecurity, essential for assessing and mitigating risks associated with software vulnerabilities. The Common Vulnerability Scoring System (CVSS) provides a standardized framework to rate the severity of security vulnerabilities, utilizing metrics divided into Base, Temporal, and Environmental categories. Although widely adopted, CVSS has limitations in context sensitivity and often fails to account for unique environmental factors that could significantly alter a vulnerability's impact on an organization [5].

B. The Emergence of Large Language Models in Cybersecurity

Recent advancements in artificial intelligence, particularly the development of Large Language Models (LLMs) [25], offer new potentials for refining traditional vulnerability management systems. LLMs are adept at processing and generating human-like text, enabling them to analyze unstructured data like security reports and software documentation extensively. This capability suggests that LLMs could play a crucial role in re-evaluating vulnerability severity more contextually and dynamically.

C. Insights from Penetration Testing

The application of LLMs in penetration testing provides valuable insights into their potential for broader cybersecurity roles. The study "PENTESTGPT: An LLM-empowered Automatic Penetration Testing Tool" illustrates that while LLMs excel at specific sub-tasks within penetration testing—such as utilizing tools and interpreting outputs—they struggle with maintaining an integrated understanding of complex security environments [3]. These findings underscore the challenges LLMs face in context retention and holistic scenario analysis, which are critical for effective vulnerability management.

D. Integrating LLMs into Vulnerability Severity Re-assessment

This research explores the feasibility of using LLMs to reassess the severity of Common Vulnerabilities and Exposures (CVEs) [2] based on specific codebase contexts. By leveraging the nuanced text processing capabilities of LLMs, it is possible to analyze detailed descriptions of vulnerabilities against the backdrop of an organization's specific operational environment. This approach aims to adjust CVSS scores to better reflect the actual risks posed to an organization, considering factors such as asset criticality, existing security measures, and prior incident data.

The integration of LLMs into vulnerability management represents a significant step towards more context-aware and dynamic cybersecurity practices. Although challenges such as context loss and the integration of complex environmental data remain, the potential for LLMs to transform vulnerability assessment practices is promising. Further research is needed to develop effective methodologies for integrating these models into existing security infrastructures, potentially leading to a more accurate and responsive approach to managing security risks.

III. Methodology

A. Preparation

1) Data Source: The primary source of vulnerability data for this study is the National Vulnerability Database (NVD) [12], curated by the National Institute of Standards and Technology (NIST) [11]. The NVD provides a comprehensive catalog of information about security vulnerabilities in publicly released software packages.

2) Dataset Construction: The dataset, termed the CVE-Context Dataset, includes vulnerabilities filtered based on specific criteria:

Severity Filtering: Only vulnerabilities rated as 'High' or 'Critical' were included in order to focus on those with potentially severe impacts. Technology Filtering: The vulnerabilities selected pertain to Python [7], Node.js [6], Linux [18], and OpenSSL [20] — technologies that are prevalent in the scenarios considered for this study.

3) Dataset: This study uses a dataset that comprises of 860 common Vulnerabilities and Exposures (CVE) against their evaluations made by Large Language Model. The dataset includes the correct responses along with the Model's responses across the 5 different cases. Each row of the dataset

corresponds to a unique CVE and it's Severity. In this study we have 360 critical severity cases and 500 High severity cases. Each row in the dataset also has the correct response and the LLM's response to the 5 different cases.

The full dataset and cases are published publicly [1].

B. Model Selection and Setup

1) Model Description: GPT-4 [14] by OpenAI [16] is an advanced generative pre-trained transformer capable of understanding and generating natural language responses [15]. Its architecture is designed to handle a wide range of AI tasks, making it ideal for interpreting complex and nuanced cyber security data.

We used the latest gpt-4 model in 2023 [14]. The model was selected because it had the highest reasoning benchmarks at the time we ran the experiment [14].

2) Interaction with the Model: Prompt Engineering: The prompts were crafted to clearly define the context of each CVE [2] within specific operational scenarios, asking the model to decide on the potential impact (Yes, No, Unsure).

3) Response Handling: : The responses from GPT-4 were meticulously recorded, forming a structured output that was later analyzed against expert evaluations.

4) SDK Interactions: : We interacted with GPT-4 via the API using the Open AI Python SDK [13].

C. Operational Scenarios

The scenarios were chosen to reflect typical and critical use cases in modern IT operations, ranging from cloud-based services to traditional on-premise deployments.
Scenario Descriptions:

1) Debian-based Container [4] on AWS ECS [22]: Examines the implications of a Python web application that does not handle SSL termination, a common setup in cloud-native environments.
2) Ubuntu-based [10] Container [4] Running Node.js [6] on AWS Fargate [23]: Focuses on event-driven applications in a server less architecture, assessing the impact of node-specific vulnerabilities.
3) Python AWS Lambda Function Behind an API Gateway: Tests the security robustness of server less computing frameworks [24], which are increasingly used for scalable web applications.
4) Ubuntu-based [10] EC2 [21] Instance Hosting a Nodejs [6] Web App: Represents traditional cloud deployments, examining how node vulnerabilities affect web services.
5) Ubuntu-based Container Running CI Scripts on AWS: Addresses the security concerns in automated build and deployment processes, critical for maintaining the integrity of software development pipelines.

D. Experiment Overview

We pass the following template to the LLM for each case for each vulnerability:

```
f"This is a CVE {cve_id} detail:
↪ {cve_description}. Reply by only
↪ returning yes or no or unsure, would
↪ this affect me if {case_description}"
```

At the same timewe filled the correct answer as determined by a senior engineer. The results are recorded in a CSV with a row for each vulnerability and columns that have the LLM and Human result for each case [1].

E. Metrics

The key metrics used in the evaluation included:

1) Accuracy: Measures the proportion of total correct predictions made by the model.
2) Precision and Recall: Assess the model's effectiveness in identifying relevant CVEs without false positives or negatives.

F. Limitations and Other Considerations

1) Model Limitations: While GPT-4 is a robust model, it is constrained by the data on which it was trained, which may not fully encompass the specific technical nuances of cybersecurity threat assessment.

2) Expert Subjectivity: The interpretations by the panel of experts, while informed, are inherently subjective and could vary based on individual expertise and experience.

3) Generalizability: The findings from this study are based on a select set of technologies and scenarios, which may not be universally applicable across all possible CVEs or IT configurations.

G. Analysis

The data was analysed on across all the different dimensions. Figure 1 is a radar chart showing the performance of the

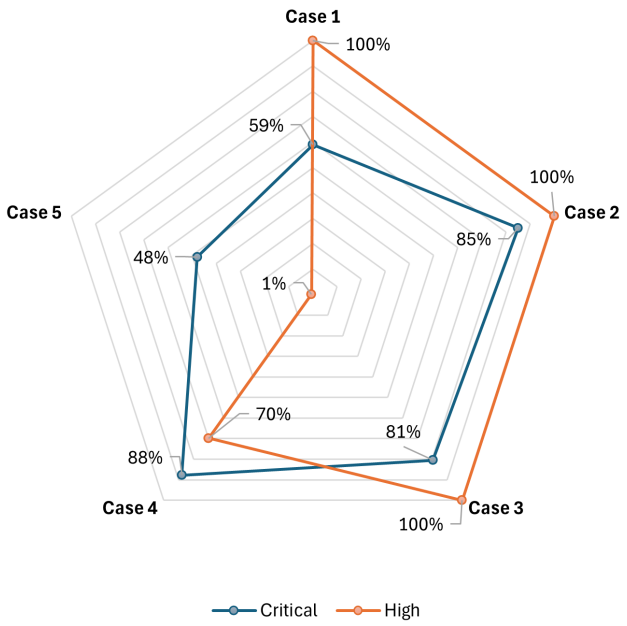


Fig. 1: Severity Radar Chart

LLM on severity levels amongst the 5 cases. The evaluation on the two severity levels tested reveals that the Large Language Model performs well on the 'High' severity cases achieving a 100% alignment in cases 1, 2 and 3 and then showing a extreme drop for case 4 to 70% and case 5 to a drastic 1%. For 'Critical' Severity the LLM shows high accuracy for case 2, 3 and 4 but plummeted considerably in cases 1 and 5 to 59% and 48% respectively.

For each Case we compare the the large large model's judgement with the anticipated correct answer. For this purpose we construct a 3X3 confusion matrix with the x-axis as the LLM's answer and the Y-axis as the correct answer. The confusion matrix demonstrates the count of CVEs in 9 possible combinations. By examining these matrices we can determine the accuracy to identify specific areas the LLM performs better. High values along the diagonal suggest strong performance while the rest of the cells suggest possibility of improvement.

Figure 2 shows the distribution for Case 1 Debian-based Container [4] on AWS. The accuracy is 82.9%

- Precision:
  - Yes: 0.896
  - No: 0.822
  - Unsure: 0.795
- Recall:
  - Yes: 0.502
  - No: 0.944
  - Unsure: 1.00

Fig. 2: Case 1 Confusion Matrix

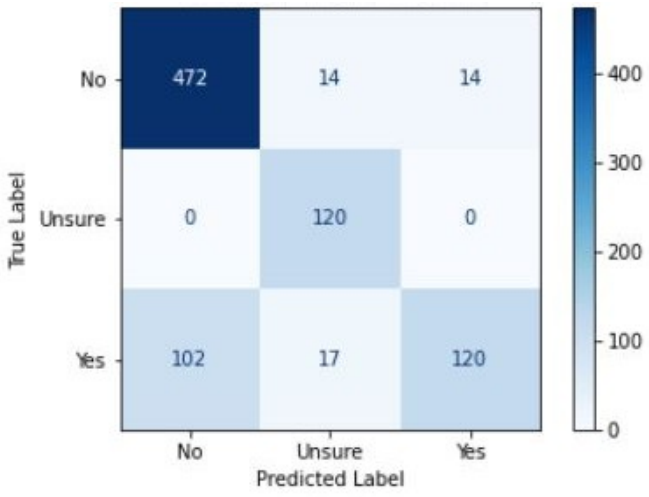
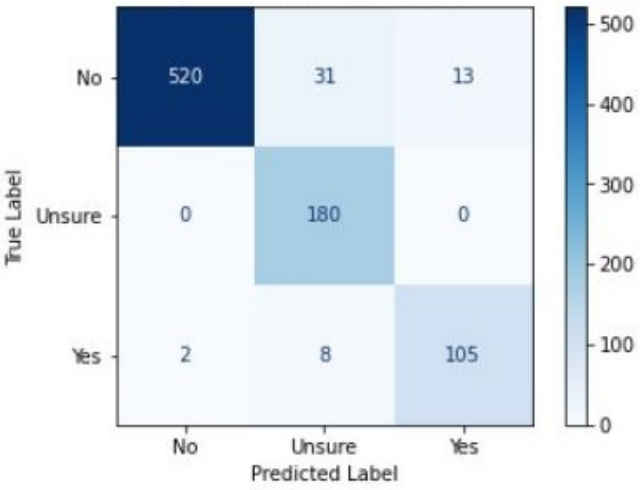


Figure 3 shows the distribution for Case 2 Ubuntu-based [10] Container [4] Running Node.js [6] on AWS Fargate [23].The accuracy is 93.7%

- Precision:
  - Yes: 0.890
  - No: 0.996
  - Unsure: 0.822
- Recall:
  - Yes: 0.913

Fig. 3: Case 2 Confusion Matrix



  - No: 0.922
  - Unsure: 1

Figure 4 shows the distribution for Case 3 Python AWS Lambda Function Behind an API Gateway.The accuracy is 91.9%

- Precision:
  - Yes: 0.807
  - No: 0.989
  - Unsure: 0.833
- Recall:
  - Yes: 0.977
  - No: 0.890
  - Unsure: 1

Fig. 4: Case 3 Confusion Matrix

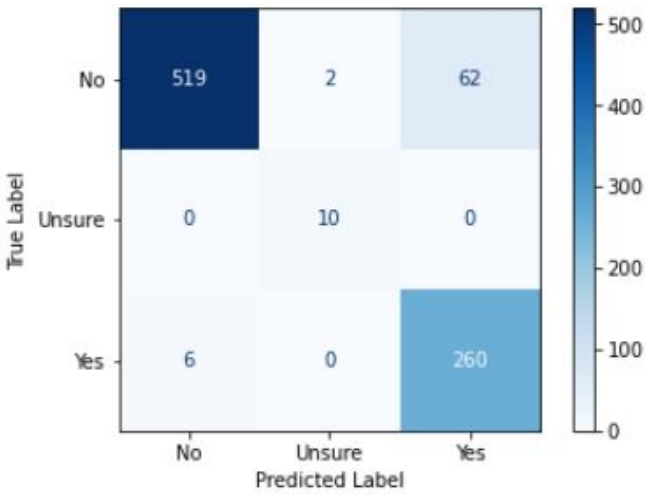
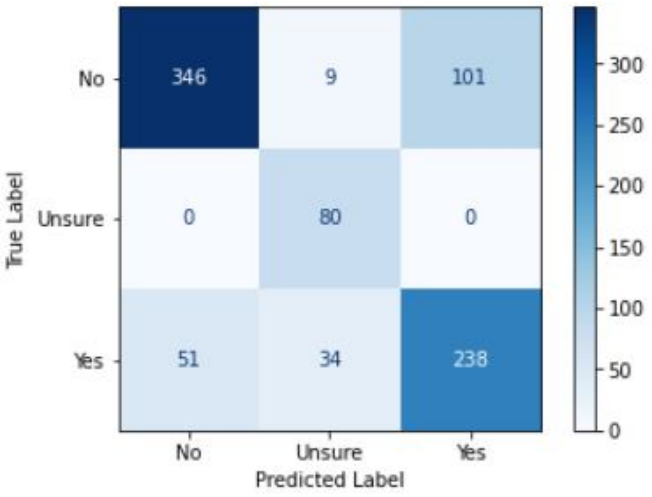


Figure 6 shows the distribution for Case 4 Ubuntu-based [10] EC2 [21] Instance Hosting a Nodejs [6] Web App. The accuracy is 77.3%

- Precision:
  - Yes: 0.702

Fig. 6: Case 4 Confusion Matrix



  - No: 0.872
  - Unsure: 0.650
- Recall:
  - Yes: 0.737
  - No: 0.759
  - Unsure: 1.00

Fig. 7: Case 5 Confusion Matrix

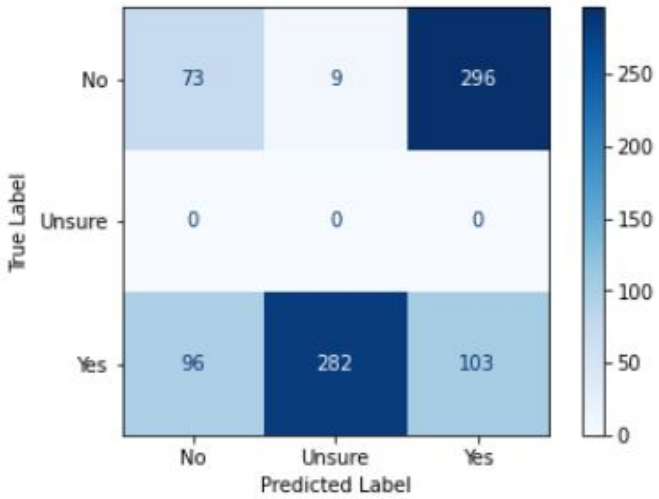


Figure 7 shows the distribution for Case 5 Ubuntu-based Container Running CI Scripts on AWS.The accuracy is 20.4%

- Precision:
  - Yes: 0.256
  - No: 0.432
  - Unsure: 0 *Undefined due to no true unsure predictions
- Recall:
  - Yes: 0.213
  - No: 0.193
  - Unsure: NaN *Undefined due to no actual unsure instances

H. Results

As a result of the analysis each case showed a different accuracy level. Fig 5 shows a bar graph of accuracy against each case as a comparison. The average accuracy across all was found to be 73.24%, however it is important to note that certain cases exhibited higher accuracy rates than others.

Case 1 demonstrated an accuracy rate of approximately 83%, indicating a relatively high level of accuracy in the large model's judgment. Similarly, Case 2 exhibited an impressive accuracy rate of 93.7%, while Case 3 achieved a commendable accuracy rate of 91.8%. In contrast, Case 4 displayed a lower accuracy rate of 77.3%. The most challenging case, Case 5, yielded the lowest accuracy rate of 20.5%.

The Precision and Recalls of the different cases showed that while the LLM acheived a high precision and recall for "No" and "Unsure" categories in most of the cases, the performance lacks for the "Yes" category.

IV. Further Work

The current study has exclusively utilized GPT-4 due to its leading performance in relevant benchmarks [17] at the time of our research. Recognizing the rapid advancement in the field of artificial intelligence, further work is proposed to expand our approach to include a broader range of Large Language Models. This will allow us to validate the robustness of our findings and explore the potential of newer or alternative models that may offer unique advantages. Specifically, future work should focus on the following areas:

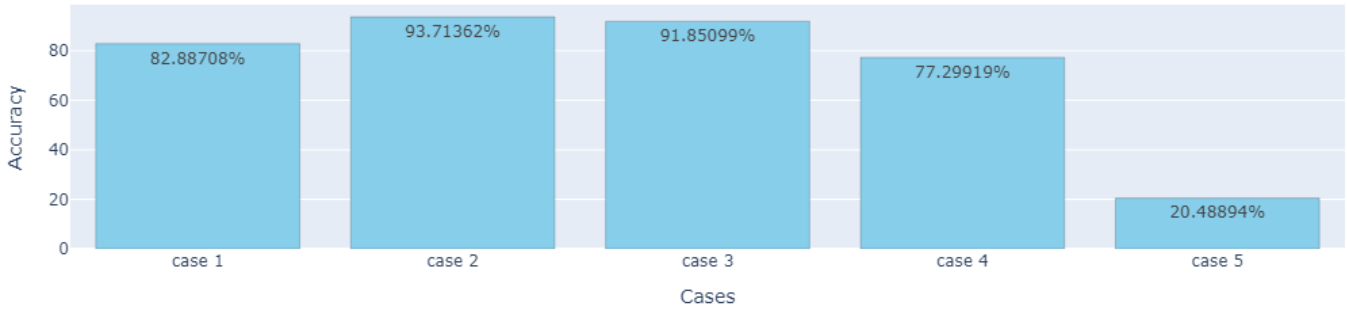


Fig. 5: Accuracy by Case

1) Model Comparison: Incorporating a comparative study of different LLMs to assess their effectiveness in vulnerability management. This will help identify whether newer or different models provide improved accuracy, contextual understanding, or operational efficiency compared to GPT-4.
2) Integration Guidelines: Developing detailed guidelines on how LLM outputs can be seamlessly integrated into existing vulnerability management systems. This involves creating standardized protocols for interpreting AI assessments and implementing these insights within the decision-making processes of IT security teams.
3) Case Studies: Conducting case studies in diverse organizational contexts to demonstrate the practical application of LLM assessments in real-world scenarios. These studies should document the steps from AI output to actionable insights, showcasing the tangible benefits and potential hurdles in adopting AI-driven vulnerability management.
4) Automation Tools: Investigating the development of automation tools that can directly ingest LLM outputs to adjust vulnerability prioritization and response strategies automatically. This would help in minimizing human intervention and accelerating response times, enhancing the overall efficacy of cybersecurity measures..
5) Feedback Loops: Establishing feedback loops within the AI models to refine their accuracy and relevance based on real-time data and user feedback. By continuously training the LLM on updated data and outcomes, the model can evolve to better meet the specific needs of different IT environments.
6) Feedback Loops: Establishing feedback loops within the AI models to refine their accuracy and relevance based on real-time data and user feedback. By continuously training the LLM on updated data and outcomes, the model can evolve to better meet the specific needs of different IT environments.

By expanding the scope of AI models tested and developing a comprehensive framework for their integration, future work will aim to solidify the role of LLMs in enhancing cybersecurity practices, ensuring that organizations can leverage the full potential of AI in managing vulnerabilities.

V. Conclusion

In conclusion, the research findings suggest that Large Language Models have the potential to overcome the limitations of Common Vulnerabilities and Exposures (CVE) classifications by incorporating project context and scenario details.

The results of the study highlights the varying performance levels of the large language model across different cases. While the average accuracy of 73.24% provides an overall assessment, it is evident that the model's accuracy is heavily influenced by the specific case being evaluated. The higher accuracy rates observed in Cases 1, 2, and 3 suggest that the model excels in certain scenarios, while the lower accuracy rates in Cases 4 and 5 indicate areas where improvement may be necessary.

The model is more reliable to determine when the CVE has "No" impact on the scenario or when the impact is uncertain i.e. "Unsure" compared to the "Yes" category when there is an impact.

Overall, these findings emphasize that while the Large Language Model shows promise, considerations should be made about the specific operational scenario when evaluating the usage of large language models for CVE classifications. Future work should focus on understanding the causes of the inconsistencies and developing targeted strategies to address them.

Author Contribution

Conceptualization, A.S. and M.A.; Methodology, A.S. and M.A.; Software, M.A.; Validation, A.S. and M.A.; Formal Analysis, A.S.; Investigation, A.S.; Resources, M.A.; Data Curation, M.A.; Writing – Original Draft Preparation, A.S. and M.A.; Writing – Review & Editing, A.S. and M.A.; Visualization, A.S.; All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data Availability Statement

The original data presented in the study are openly available on GitHub in [1].

Conflict of Interest

The authors declare that they have no competing interests. Therefore, no known competing financial interests or personal relationships could have appeared to influence the work reported in this paper.

References

[1] Mojtaba Alfardan. Llm vunerability classifications using an llm dataset. https://gist.github.com/mojtaba42/caec9c5d3fea4e7622e0ccecf34d0af9, 2024. Accessed: June 11, 2024.
[2] MITRE Corporation. Cve - common vulnerabilities and exposures. https://cve.mitre.org, 2024. Accessed: June 11, 2024.
[3] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. Pentestgpt: An llm-empowered automatic penetration testing tool, 2023.
[4] Inc. Docker. Understanding docker containers. https://docs.docker.com/get-started/overview/docker-containers, 2024. Accessed: June 11, 2024.
[5] FIRST.Org. Common vulnerability scoring system v3.1: Specification document, 2024.
[6] Node.js Foundation. Node.js documentation. https://nodejs.org/en/docs/, 2024. Accessed: June 11, 2024.
[7] Python Software Foundation. Python documentation. https://www.python.org/doc/, 2024. Accessed: June 11, 2024.
[8] Emanuele Iannone, Roberta Guadagni, Filomena Ferrucci, Andrea De Lucia, and Fabio Palomba. The secret life of software vulnerabilities: A large-scale empirical study. *IEEE Transactions on Software Engineering*, 49(1):44–63, 2023.
[9] Bill Jung, Yan Li, and Tamir Bechor. Cavp: A context-aware vulnerability prioritization model. *Computers & Security*, 116:102639, 2022.
[10] Canonical Ltd. Ubuntu. https://ubuntu.com/, 2024. Accessed: June 11, 2024.

[11] National Institute of Standards and Technology. National institute of standards and technology. https://www.nist.gov/. Accessed: June 11, 2024.
[12] National Institute of Standards and Technology. National vulnerability database. https://nvd.nist.gov/. Accessed: June 11, 2024.
[13] OpenAI. openai-python, version 1.30.2. https://github.com/openai/openai-python/tree/v1.30.2. Accessed: June 11, 2024.
[14] OpenAI. gpt-4-0613. https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4, 2023. Accessed: June 11, 2024.
[15] OpenAI. Gpt-4: Scaling language model performance. https://cdn.openai.com/reports/GPT-4.pdf, 2023. Accessed: June 11, 2024.
[16] OpenAI. Openai website. https://www.openai.com, 2024. Accessed: June 11, 2024.
[17] OpenCompass. Opencompass 2.0 large language model leaderboard. https://huggingface.co/spaces/opencompass/opencompass-llm-leaderboard, 2024. Accessed: April 11, 2024.
[18] Linux Kernel Organization. Linux kernel documentation. https://www.kernel.org/doc/html/latest/, 2024. Accessed: June 11, 2024.
[19] Ivan Pashchenko, Henrik Plate, Serena Ponta, Antonino Sabetta, and Fabio Massacci. Vulnerable open source dependencies: counting those that matter. pages 1–10, 10 2018.
[20] The OpenSSL Project. Openssl documentation. https://www.openssl.org/docs/, 2024. Accessed: June 11, 2024.
[21] Amazon Web Services. Amazon elastic compute cloud (ec2). https://aws.amazon.com/ec2/, 2024. Accessed: June 11, 2024.
[22] Amazon Web Services. Amazon elastic container service (ecs). https://aws.amazon.com/ecs/, 2024. Accessed: June 11, 2024.
[23] Amazon Web Services. Amazon fargate. https://aws.amazon.com/fargate/, 2024. Accessed: June 11, 2024.
[24] Amazon Web Services. Aws serverless computing. https://aws.amazon.com/serverless/, 2024. Accessed: June 11, 2024.
[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
[26] Michał Walkowski, Jacek Oko, and Sławomir Sujecki. Vulnerability management models using a common vulnerability scoring system. *Applied Sciences*, 11(18):8735, 2021.