A Temporal Paradox in Software Vulnerability Prioritization: Why Do Large Language Models Perform Better Post-Knowledge Cutoff Date?

Osama Al Haddad
Macquarie University, Sydney, Australia
osama.alhaddad@hdr.mq.edu.au

Muhammad Ikram, and Young Choon Lee
Macquarie University, Sydney, Australia
{muhammad.ikram, young.lee}@mq.edu.au

Muhammad Ejaz Ahmed
Data61 CSIRO, Sydney, Australia
ejaz.ahmed@data61.csiro.au

***Abstract*—As Security Operations Center (SOC) teams face challenges analyzing disparate threat feeds with varying amounts of information, Large Language Models (LLM) are a promising technology that can scale vulnerability prioritization efforts. However, critical to LLMs generating accurate responses is high-quality data on which the LLMs are trained. Recent literature suggests that a small and near-constant number of compromised training data can affect performances of LLMs of varying sizes. To investigate this possible phenomena in a SOC environment, we evaluated LLM and Prompting Technique (PT) combinations to prioritize software vulnerabilities, using the Cybersecurity Infrastructure and Security Agency's Stakeholder-Specific Vulnerability Categorization (SSVC) framework. OpenAI ChatGPT 4o-mini, Anthropic Claude 3 Haiku, and Google Gemini Flash 1.5, across 12 PTs, were instructed to analyse 384 real-world vulnerability samples over three trials, and to return values for the four SSVC decision points (SDP). These vulnerabilities were classed pre- or post-Knowledge Cutoff Date (KCD) – pre-KCD where the vulnerabilities are within the cutoff dates of all the investigated LLMs, and post-KCD are beyond the all LLM cutoff dates. For each trial, F1-scores were calculated for each LLM-PT-SDP-KCD combination. A harmonic mean was then calculated across the three trials to yield a single performance score for each LLM-PT-SDP-KCD combination. We found that LLMs tended to perform stronger on post-KCD vulnerabilities than on pre-KCD, with Gemini Flash 1.5 the strongest performer overall in conjunction on the Chain of Thought and Few Shot PTs, particularly for the Exploitation SDP. To explain this observation, we posit that the revisions of the vulnerability prioritization life cycle amount to a type of data compromise in the training dataset, such that LLMs are hindered by older and interim reports and classifications of vulnerabilities, hence impacting their ability to provide accurate software vulnerability classifications. In conclusion, we call for greater transparency in LLM training datasets for vulnerability prioritization tasks, as well as further exploration of methods to generate LLM training datasets optimized for vulnerability prioritization. Code, prompt templates and data are available here.**[1]

***Index Terms*—Large Language Models, Software Vulnerabilities, Stakeholder-Specific Vulnerability Categorization, Prompting Techniques**

I. Introduction

Effective vulnerability management is critical for organizations aiming to mitigate cybersecurity threats and minimize risk exposure. A core challenge within this process is *vulnerability prioritization* – the task of determining which vulnerabilities require immediate attention and which can be deferred. Traditionally, this task has relied on manual or semi-automated processes within the broader vulnerability management lifecycle. These methods often require specialized tools and expert judgment to interpret alerts and assess potential impacts, making the process both resource-intensive and time-consuming [1]. Under high-pressure conditions, this burden can lead to misclassifications, delayed responses, or missed critical threats.

To address these limitations, researchers have turned to machine learning and deep learning techniques [2]–[5]. These models use historical data to classify and prioritize vulnerabilities based on statistical patterns. However, three key limitations remain: (1) a scarcity of large-scale, high-quality labeled datasets [6]; (2) high dependency on the distribution and completeness of training data, which hinders generalization to novel vulnerabilities [7]; and (3) difficulty adapting to evolving threats, including zero-day exploits and emerging attack vectors [8].

Recently, LLMs like ChatGPT, Claude, and Gemini, have emerged as promising tools for automating vulnerability analysis. Pre-trained on vast textual corpora, these models can interpret unstructured and semi-structured data with minimal reliance on manual labeling [9]. LLMs have shown strong performance in related security domains, including source code analysis [10] and threat intelligence classification [11], suggesting their potential for vulnerability prioritization.

[1]https://github.com/LLM-Vulnerability-Prioritization/Temporal-Paradox

Workshop on LLM Assisted Security and Trust Exploration (LAST-X) 2026
27 February 2026, San Diego, CA, USA
ISBN 978-1-970672-05-3
https://dx.doi.org/10.14722/last-x.2026.23016
www.ndss-symposium.org

While a promising development, concerns have been raised on the training datasets of these LLMs hampering performance. One notable paper by the developers of the Claude LLM, Anthropic, observed that a small and near-constant amount of data can influence LLM performance, even for larger LLMs where one would intuit that this small amount of poisoned data would have a negligible impact [12]. Questions also arise from the use of synthetic data for training data to optimize LLM performance, with recent research challenging a current trend using synthetic data for training in the hopes of improved performances [13], [14].

Contribution. Based on this emerging body of literature, we seek to determine if a similar phenomena is observed in using LLMs for vulnerability prioritization. We do so by asking the following research question:

- *For vulnerability prioritization tasks, do LLMs perform better on vulnerabilities published pre-KCD or post-KCD, and why?*

The hypothesis is that if an LLM is unaffected by compromised training data, its performance in vulnerability prioritization of pre-KCD vulnerabilities would outperform its prioritization performance of post-KCD vulnerabilities. This would result from the LLM having seen the vulnerability in its high quality dataset and returning accurate responses. However, if performances post-KCD are stronger, this would suggest that training sets are interfering with an LLM's ability to accurately prioritize vulnerabilities pre-KCD, resulting in a stronger post-KCD performance in comparison.

II. Related Works

In this section, we present a summary of the literature in the field of artificial intelligence for vulnerability management, which are summarized in Table I.

Machine Learning for Vulnerability Analysis and Prioritization. Security researchers have extensively leveraged machine learning and deep learning models to perform vulnerability analysis and prioritization. Ghaffarian et al. [15] categorize these approaches into three main types: (1) Supervised Learning, where models are trained to infer specific functions by pairing inputs with desired outputs; (2) Unsupervised Learning, where models infer relationships within datasets without predefined labels; and (3) Reinforcement Learning, where models learn and adjust behavior based on reward functions to achieve desired outcomes. For instance, Harer et al. [16] demonstrated that machine learning models could effectively detect vulnerabilities in C and C++ code using static analysis results as ground truth, achieving an ROC score of 0.87. Similarly, Tanga et al. [17] found that unsupervised learning models were effective in discriminating between benign and malicious network traffic, highlighting their potential for detecting novel and zero-day vulnerabilities due to their independence from predefined ground truth datasets. Hore et al. [18] developed Deep VULMAN, a reinforcement learning-based framework for dynamically prioritizing security threats and allocating resources, addressing the limitation of prior research that treated vulnerability triaging as a one-time decision process, which is suboptimal in the dynamic real-world context of vulnerabilities.

Vulnerability categorization with LLMs. The use of LLMs for vulnerability prioritization has also become a burgeoning field of research [19]. One line of research focuses on using Bidirectional Encoder Representations from Transformers (BERT) to process text by analysing the context, namely, preceding and following words. This comprises two components: Masked Language Modelling, which predicts masked words (i.e., the words to be predicted) using the surrounding context, and Next Sentence Prediction, which predicts the sequence in which text should occur [20]. Shahid et al. developed CVSS-BERT, where the authors trained several BERT systems to map vulnerability descriptions to Common Vulnerability Scoring System (CVSS) vector string values, achieving F1-scores between 0.83 and 0.95 [21]. Yin et al. [22] developed ExBERT, a classification model capable of determining the likelihood that a vulnerability will be exploited from available descriptions, with F1-scores of up to 0.91. Das et al. [23] developed V2W-BERT, mapping Common Vulnerabilities and Exposures (CVE) to Common Weakness Enumeration (CWE) based on reports from MITRE and the National Vulnerability Database, achieving Accuracy scores of up to 0.97. Aghaei et al. [24] developed CVEDrill, a custom LLM model designed to analyze vulnerability data, recommend a CVSS vector string, and classify the vulnerability to the correct CWE, using a pre-trained SecureBERT model, achieving F1 scores of greater than 0.9 on select CVSS metrics.

Another line of research focuses on Generative Pre-trained Transformers (GPT). GPT-based models are trained in an unsupervised manner on large text corpora. In combination with self-attention mechanisms, this allows GPTs to process input texts and predict subsequent words for their outputs when prompted [25]. Oniagbi [26] found LLMs beneficial in analysing Security Information and Event Management (SIEM) data and classifying events as *Interesting* or *Not Interesting*, achieving an Accuracy of 0.914. On threat intelligence, Shaswata et al. [27] developed LocalIntel, a GPT-3.5-turbo-based framework combining global knowledge of vulnerabilities with local context knowledge of an organization's systems to contextualise vulnerabilities, achieving a moderate F1-score of 0.68. The researchers used a prompting approach which, in part, uses Chain of Thought to improve response accuracy, as well as setting Temperature to zero to remove variation from responses, though the capacity for LLMs to be truly deterministic is debated [28].

III. Methodology

Our study seeks to explore how LLMs perform for vulnerability prioritization tasks on vulnerabilities perform pre- and post-KCD. Figure 1 explains our data collection and analysis pipeline.

TABLE I
SUMMARY OF ML AND LLM RESEARCH FOR VULNERABILITY ANALYSIS

Author(s)	Research Description	Metric	Score
Machine Learning Approaches			
Harer et al. [16]	ML for C/C++ vulnerability detection	ROC	0.87
BERT-based Approaches			
Shahid et al. [21]	CVSS-BERT for CVSS vector mapping	F1	0.83-0.95
Yin et al. [22]	ExBERT for exploitation likelihood prediction	F1	0.91
Das et al. [23]	V2W-BERT for CVE to CWE mapping	Accuracy	0.97
Aghaei et al. [24]	CVEDrill: Custom LLM for vulnerability analysis	F1	>0.9
GPT-based Approaches			
Oniagbi [26]	LLM for SIEM event classification	Accuracy	0.914
Shaswata et al. [27]	LocalIntel: GPT-3.5 with local context	F1	0.68

A. Sampling Approach

Our study used the Vulnrichment repository, which contained 45,621 distinct CVE IDs as of November 22, 2024. Given computational and cost constraints, we employed statistical sampling with a 95% confidence interval, 5% margin of error, and 50% proportion assumption [29]. This yielded an initial minimum required sample size of 380. To facilitate other research, we rounded the number of samples to 384, resulting in a sample size of 192 per class (i.e. pre-KCD and post-KCD).

B. Analysis Dataset: VulZoo

Per Figure 1 Step 1, VulZoo is a GitHub repository of open-source vulnerability datasets, curated by Ruan et al. [30], from which the vulnerability details are sourced. Given LLM strengths in analyzing semi-structured and unstructured data, the fields as listed in Table II have been extracted where a CVE ID is present in the record, to be parsed with PTs and submitted for querying to LLMs. The VulZoo dataset comprises Git-based repositories, direct file downloads, security advisory mail crawlers, and patch codes. The authors addressed common data quality challenges of vulnerability datasets through [30]: (1) validating URL links; (2) running SHA256-based de-duplication; and (3) removing irrelevant files.

C. Knowledge Cutoff Date Stratification

The KCDs of the selected LLMs are shown in Table III. To compare LLM performances pre-cutoff and post-cutoff,

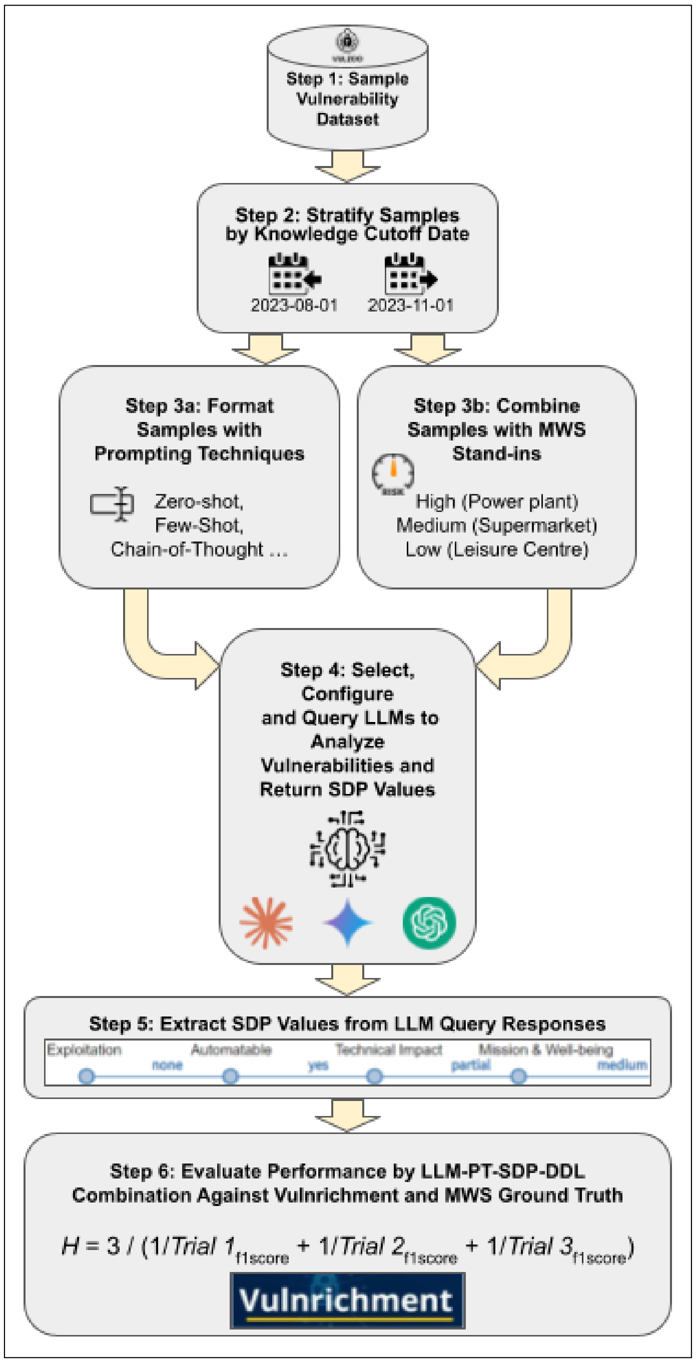
Step 1: Sample Vulnerability Dataset
Step 2: Stratify Samples by Knowledge Cutoff Date
2023-08-01
2023-11-01
Step 3a: Format Samples with Prompting Techniques
Zero-shot, Few-Shot, Chain-of-Thought ...
Step 3b: Combine Samples with MWS Stand-ins
High (Power plant)
Medium (Supermarket)
Low (Leisure Centre)
Step 4: Select, Configure and Query LLMs to Analyze Vulnerabilities and Return SDP Values
Step 5: Extract SDP Values from LLM Query Responses
Exploitation
none
Automatable
yes
Technical Impact
partial
Mission & Well-being
medium
Step 6: Evaluate Performance by LLM-PT-SDP-DDL Combination Against Vulnrichment and MWS Ground Truth
H = 3 / (1/Trial 1 f1score + 1/Trial 2 f1score + 1/Trial 3 f1score)
Vulnrichment

Fig. 1. Overview of our methodology showing how stratified vulnerability samples, prompting strategies, and MWSs are processed and queried across multiple LLMs to extract SSVC decision values and evaluate performance using harmonic-mean F1-scores.

Table III indicates that all the knowledge bases of the LLMs cover dates prior to 2023-08-01, while none of the knowledge bases cover dates after 2023-11-01. Therefore, per Figure 1 Step 2, to generate pre-cutoff and post-cutoff samples, vulnerabilities before 2023-08-01 are tagged as pre-cutoff, while vulnerabilities after 2023-11-01 are tagged as post-cutoff.

TABLE II
VULZOO DATASETS AND SEMI-STRUCTURED AND UNSTRUCTURED FIELDS SELECTED FOR STUDY, AS WELL AS THE COUNT OF SAMPLE CVE IDS IN THE DATASET

Dataset	# of CVEs	Fields
AttackerKB	256	Document Description; MITRE Tactics; Timeline; Vulnerable Versions; Vendor Product Names; Tags
Bugtraq	8	Advisory Email
CISA KEV	20	Vendor; Product; Vulnerability Name; Short Description; Required Action; Known Ransomware Campaign Use
CVE	384	Description
Exploit DB	8	Description; Content
Full Disclosure	10	Advisory Email
Linux Vulns	23	Descriptions; Title; Email Advisory
OSS Security	20	Advisory Email
Patch Database	11	Code
ZDI Advisory	109	Title; Vendors; Products; Description; Additional Details; Timeline

TABLE III
LLM KNOWLEDGE CUTOFF DATES.

LLM	Cutoff Date
OpenAI ChatGPT 4o-mini	2023-10-01
Anthropic Claude 3 Haiku	2023-08-01
Google Gemini Flash 1.5	2023-11-01

D. Selecting Prompting Techniques

To evaluate LLMs under diverse interaction paradigms, per Figure 1 Step 3a, we utilize a set of PTs informed by the taxonomy proposed by Tony et al. [31]. PTs which are grouped into five major categories:

1) **Root Techniques:** Basic, standalone prompts used as a baseline for LLM performance (e.g., ZS, OS).
2) **Refinement-Based Techniques:** Iterative prompting techniques where model outputs are revised based on feedback.
3) **Decomposition-Based Techniques:** Techniques that segment complex problems into smaller tasks to improve reasoning (e.g., step-by-step).
4) **Reasoning-Based Techniques:** Prompts that encourage logical inference, such as CoT.
5) **Priming Techniques:** Techniques that include cultural memes to orient the model towards desirable outputs (e.g., role-play).

Some PTs–such as *Progressive Hint* and *Least-to-Most*–require users to provide feedback or structure sub-tasks interactively across multiple prompts [32]. While these methods have shown promise in guiding LLMs toward more accurate outputs, they place a significant burden on the user to act as a facilitator of the model's reasoning process. This is suboptimal in a high-pressure SOC environment and as such, were excluded from the study. See Appendix A for the list of PTs used in this study.

E. Selecting Vulnerability Prioritization Framework

To facilitate other ongoing research, we selected the SSVC framework with which to instruct LLMs to perform their analyses, given its aim to provide context-aware prioritizations. This is in contrast to the other vulnerability prioritization frameworks which are generic in nature and do not take into account an organization's context. Further, given research has already been conducted in using LLMs for vulnerability prioritization frameworks such as the CWE and CVSS, the relative lack of research using LLMs for SSVC prioritization is an opportunity for this study to contribute to the field. The SSVC comprises four decision points:

1) **Exploitation:** Assesses whether the vulnerability is currently being exploited or has credible reports of exploitation in the wild.
2) **Automatable:** Determines the extent to which the exploitation process can be automated, which may impact how quickly the threat can spread or be weaponized.
3) **Technical Impact:** Evaluates the potential damage the vulnerability could cause, such as loss of confidentiality, integrity, or availability of critical systems.
4) **Mission & Wellbeing:** Reflects the potential harm to the organization's mission or human wellbeing if the vulnerability is exploited, emphasizing contextual impact beyond technical considerations.

. These decision points values, when parsed through the decision tree, culminate in a prioritization outcome. Given requirements to provide justifiable decisions [33], per Figure 1 Step 4, we instructed LLMs to provide values to the four decision points in their response, rather than provide a final outcome. Doing so provides SOC teams some insight into how an LLM has analyzed a given vulnerability in a manner that is organization-specific.

F. Selecting, Configuring and Querying LLMs

Given their ability to analyze semi-structured and unstructured textual data in an automated manner, LLMs appear to be the appropriate tool to analyze vulnerabilities and recommend responses. Newer proprietary LLM models, such as ChatGPT o1, Claude Opus 3, and Gemini Pro 1.5 have high input and output per million token costs. At the time of writing, the total cost per million input and output tokens ranges from US$6.25 to US$90 for these premium models [34], making them prohibitively expensive for this study. As such, for Figure 1 Step 4, we selected models that were cost-effective and exhibited good performances per commonly-accepted benchmarks [35], listed in Table IV. Based on the subsequent LLM queries that were run in this study, this yielded costs as shown in Table V.

TABLE IV
LLM CONTEXT WINDOWS AND PRICING [34].

LLM	Context Window	Input ($/M)	Output ($/M)
ChatGPT 4o-mini	128K	0.15	0.60
Claude 3 Haiku	200K	0.25	1.25
Gemini Flash 1.5	1M	0.075	0.30

TABLE V
TOTAL COST OF LLM TOKEN USAGE. CALCULATED BY OPENROUTER PRICES [34] AND THE SUM OF THE 'TOTAL_TOKENS' FOR ALL QUERY RESPONSES FROM OPENROUTER.

LLM	Total Tokens	Cost/M ($)	Total Cost ($)
Claude 3 Haiku	113,283,270	1.50	169.92
Gemini Flash 1.5	98,495,136	0.375	36.94
ChatGPT 4o-mini	91,151,176	0.75	68.36
		Total Cost	**275.22**

Further, these models are already accessible to many organizations globally. Industry statistics suggest that 66% of the cloud computing market share globally comprises Amazon Web Services, Microsoft Azure, and Google Cloud [36], all of which offer at least one of these LLMs, making it easier to integrate these models into existing IT infrastructure [37]–[39]. This would make this study more applicable to organizations in this situation. For this study, these LLMs are accessed via OpenRouter, an online platform that provides API access to a wide variety of LLMs [34]. The API allows for parameters such as Temperature, Top-K and Top-P to be set. We set these parameters as listed in Table VI.

TABLE VI
LLM PARAMETERS.

Parameter	Value
Temperature	0.7
Top-K	50
Top-P	0.7

While these figures are arbitrary, they do fall within ranges that guides for configuring LLMs deemed balanced [40], [41]. The intention is that the LLMs will process inputs and provide recommendations that do not excessively deviate from the task of vulnerability prioritization, while still having the freedom to consider a range of prioritization options for a given vulnerability. In total, we sent 124,416 queries to OpenRouter, as detailed in Table VII.

TABLE VII
TOTAL NUMBER OF QUERIES SENT TO OPENROUTER.

Parameter	Value
Sample Size	384
Number of PTs	12
Number of LLMs	3
Number of Trials	3
Number of MWSs	3
Total Queries	**124,416**

G. Evaluation Dataset: Vulnrichment

The Vulnrichment dataset [42] is an annotated repository of real-world software vulnerabilities maintained by the U.S. Cybersecurity and Infrastructure Security Agency (CISA). As of November 22, 2024, Vulnrichment contained 45,621 curated vulnerability records, each indexed by a unique CVE identifier. The dataset is specifically designed to support SSVC-based risk assessment workflows, offering security analysts a pre-annotated subset of CVEs with fields aligned to the SSVC decision-making model. Given the alignment between these fields and the SSVC framework, Vulnrichment serves as a suitable, albeit partial, ground truth dataset for training and evaluating SDP predictions. However, it does not include values for the Mission & Wellbeing (M&W) decision point, which is inherently stakeholder-specific and context-dependent.

1) Handling Missing Mission & Wellbeing Data: The absence of stakeholder-specific M&W values in Vulnrichment poses a challenge for complete SSVC evaluation. To approximate real-world usage, per Figure 1 Step 3b, we introduced *Mission & Wellbeing Stand-ins* (MWSs) to simulate different organizational risk profiles. This approach is consistent with prior SSVC use cases [43], where analysts assume a M&W level based on organizational mission criticality. We designed three MWSs representing common organizational risk scenarios; high, medium and low. Each MWS was embedded into the system prompt during LLM queries to guide contextual reasoning. This enabled LLMs to simulate tailored M&W decisions based on organizational characteristics. For example, in a power plant context, an LLM should reasonably infer a high M&W classification, reflecting elevated safety and operational risks. For Figure 1 Step 6 therefore, while the ground truth for the other SDPs is Vulnrichment, the ground truth for the M&W SDP is the MWS that we assigned.

H. Evaluation Metric: Harmonic Mean of F1-scores

To assess how effectively each LLM, PT and KCD combination can classify SDPs, as per Figure 1 Step 6, we evaluated model outputs against the Vulnrichment dataset [44] ground truth and our MWSs. Each LLM response was compared on a per-SDP basis: *Exploitation*, *Automatable*, *Technical Impact*, and *Mission & Wellbeing*. For this evaluation, we adopted the F1-score metric, widely used in classification tasks due to its ability to balance *precision* and *recall*. It is particularly valuable in contexts where class imbalance exists or where false positives and false negatives are of equal concern. Precision measures the proportion of correctly predicted positive instances among all predicted positive instances, while recall measures the proportion of correctly predicted positive instances among all actual positive instances. The F1-score provides a harmonic mean of precision and recall, penalizing extreme imbalances between the two. Based on commonly-accepted machine learning practices [45], scores ≥ 0.8 are indicative of strong performance, while scores between 0.5 and 0.8 suggest moderate performance. To ensure reliability across the inherent stochastic nature of LLM outputs, we conducted three independent trials for each LLM–PT–SDP-KCD combination. We then computed the harmonic mean of the F1-scores across these trials. The harmonic mean is particularly sensitive to low outlier values and less influenced by high values [46], making it an appropriate choice to identify consistently high or low-performing combinations. This methodology provides a robust comparative analysis of LLM capability in classifying SDPs, highlighting not only peak performance but also consistency across multiple runs.

IV. Results

To answer the research question, the responses for each LLM-PT-SDP combination were grouped by KCD status, namely pre-KCD and post-KCD. A harmonic mean of F1-scores from the three trials was then calculated for each LLM-PT-SDP-KCD combination. Per Figure 2, on the Automable SDP, Claude 3 Haiku performed poorly on vulnerabilities published both pre- and post-KCD. The strongest performances were yielded by the PTs with at least one exemplar. In the case of the Self Planning, One Shot, Few Shot with Explanation and Few Shot PTs, performances improved post-KCD while other PTs saw degradations. Gemini Flash 1.5 performed stronger on all PTs for vulnerabilities published after the KCD, with scores ranging from approximately 0.63 to 0.69. ChatGPT 4o-mini also saw moderate performances post-KCD, with scores ranging from approximately 0.60 to 0.68.

On the Exploitation SDP, Claude 3 Haiku performed poorly across all PTs and pre- and post-KCD, with Self Planning the best performing with approximately 0.16. All PTs except Self Consistency saw improvements post-KCD. For Gemini Flash 1.5, all PTs saw improvements post-LLM KCD. Performances ranged from moderate to good, with PTs with at least one exemplar scoring approximately 0.75 or higher. For ChatGPT 4o-mini, there was a clear distinction in performance between PTs with an exemplar and those without, as well as in pre- and post-KCD status. PTs with at least one exemplar had higher scores, both pre- and post-KCD, with performances improving to moderate or near moderate post-cutoff. The remaining PTs, while also seeing performance improvements post-KCD, all performed poorly, with the strongest performance from this group the Self Planning PT with approximately 0.34.

On the Mission & Wellbeing SDP, Claude 3 Haiku performs poorly. While most PTs saw improvements post-KCD, no PT exceeded 0.28. For Gemini Flash 1.5, all PTs performed worse post-KCD than pre-KCD, with the Memetic Proxy PT pre-KCD scoring approximately 0.47. ChatGPT 4o-mini also saw poor performance, with mixed performances within PTs pre- and post-KCD. Few-Shot with Explanation pre-KCD achieved the best score with approximately 0.27.

On the Technical Impact SDP, Claude 3 Haiku saw improvements from pre- to post-KCD across all PTs. These improvements saw PT performances move from mostly poor to mostly moderate. Gemini Flash 1.5 saw slight improvements from pre-KCD to post-KCD, with Self Consistency, One Shot, and Chain of Thought seeing slight declines. All PTs across both pre-KCD and post-KCD performed moderately. With ChatGPT 4o-mini, performance was mixed; most PTs saw significant gains while a minority of PTs saw deterioration. Most PTs saw improvements from pre-KCD to post-KCD, solidifying moderate performances. However, for the PTs with at least one exemplar, performance remained the same or worsened, with three out of four cases seeing performances fall from moderate to poor.

V. Discussion

An unexpected observation was LLM performance on vulnerabilities post-KCD, with Gemini Flash 1.5 using Chain of Thought scoring best, achieving a harmonic mean F1-score of 0.82 on the Exploitation SDP. In contrast, the same LLM, PT, and SDP combination was also the best performing for the pre-KCD period, but only scored 0.77. Statistical significance tests for the results are listed in Table VIII, suggesting a small effect size as well as a borderline significance depending on alpha.

Research explaining this phenomenon is limited, with observations tending to support improved performance on pre-KCD data. Kim et al. [47] noted the importance of including relevant data in an LLM's training dataset for improving LLM performance across a range of training strategies. Kandpal et al. [48] observed a positive correlation between LLM performance and the frequency count of data in a given dataset, which would suggest an improved performance on pre-KCD data that an LLM may have been trained on. In a study on the performance of ChatGPT 3.5 and ChatGPT 4 on coding benchmarks pre- and post-KCD, Roberts et al. [49] noted performance degradations on both LLMs for the coding benchmarks post-KCDs. One study that did find an improvement post-KCD was Alam et al. [50]. Studying LLM abilities to map attack descriptions to MITRE ATT&CK technique IDs, the authors found that Gemini 1.5 performed better on descriptions post-KCD than pre-KCD, but commented that the differences were generally insignificant and that overall LLMs performed better on samples pre-KCD. Another study by Rahman et al. [51] on LLM ability to generate accurate code for familiar code bases (pre-KCD) versus novel code bases (post-KCD) found that

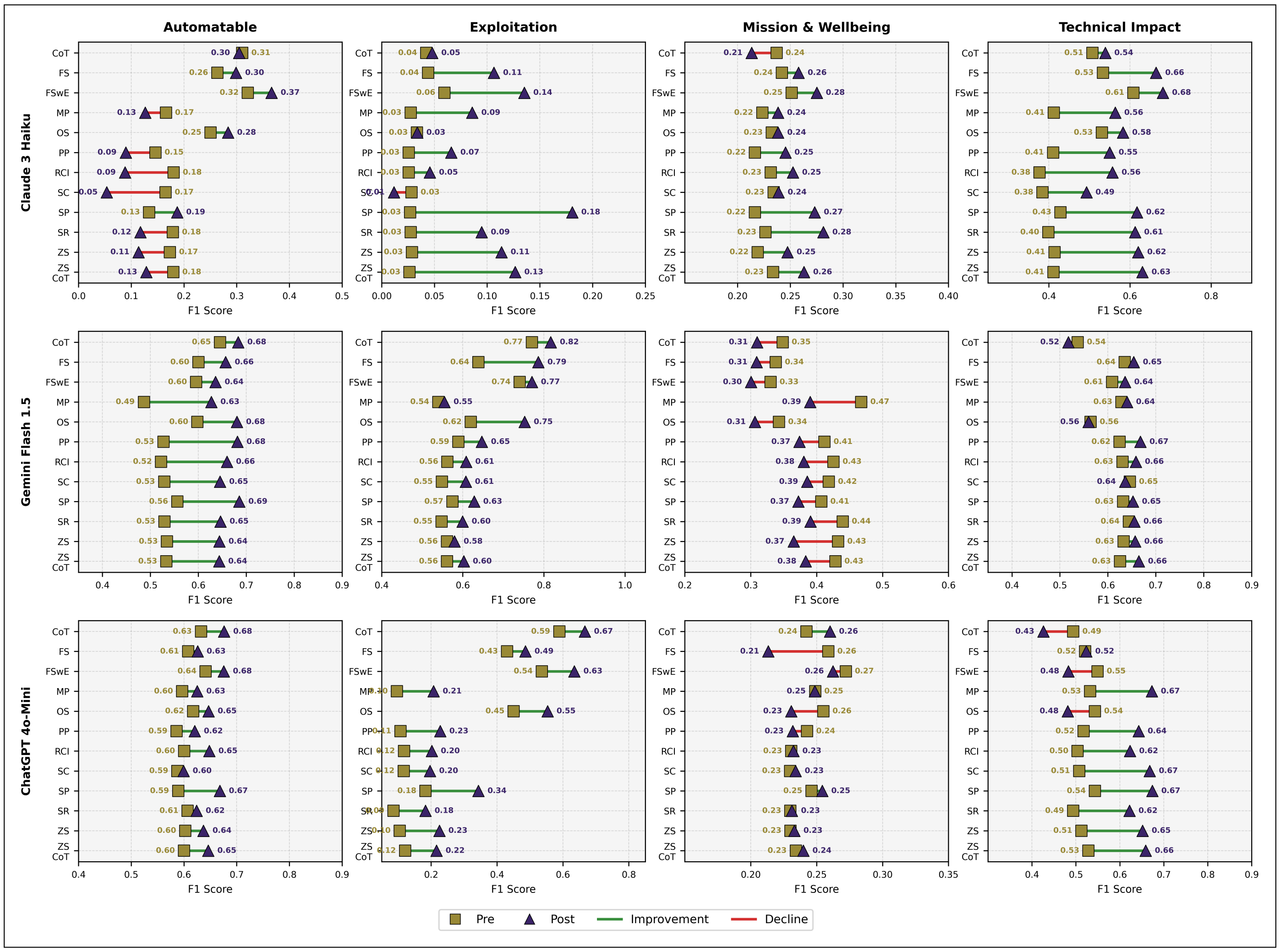

Fig. 2. Dumbbell Chart of Harmonic Means of Trial F1-scores for LLM-PT-SDP-KCD. See Appendix B for full dataset.

while GPT-4.1, GPT-5 and GPT-OSS showed stronger post-cutoff performances, these differences were deemed negligible. Cao et al. [52] observed increased performance in some LLMs on a code completion task post-KCD, but posit that given LLMs now generate a large segment of recent code, the LLMs in their study were able to replicate coding patterns and hence scored higher when compared to LLM-generated ground truth snippets, as opposed to human-generated code from prior years. Overall, these findings should support the notion that the LLMs would have performed stronger with the pre-KCD vulnerabilities.

One possible factor explaining the stronger post-KCD performance could be the quality of the data used in the training sets of the LLMs investigated in this study. Training dataset quality plays a key role in LLM performance, with Iskander et al. [53] noting that LLMs trained on a higher quality dataset outperform LLMs trained on an unvalidated dataset, even when the higher quality data is smaller. Dodge et al.'s [54] investigation of the Colossal Clean Crawled Corpus, a web-scraped dataset commonly used in natural language processing research, found that a significant proportion of the data present was obtained from select sources, such as patent repositories, Wikipedia, and news outlets, as well as noting the use of block-lists excluding clusters of scientific and legal content. Such selectivity could lead to distortions in the assessment of vulnerabilities pre-KCD. Cheng et al. [55] note that within training datasets, outdated, modified, or incorrect data may be included, even when newer versions of training datasets are generated. Further, Klang et al. [56] noted in their research that LLMs struggled most when presented with temporal-based questions. In software vulnerabilities, details can and do change over time as more information becomes available, such as attackers developing and refining exploits, and software developers and security researchers implementing patches [57]. For example, Jahanshahi and Mockus [58] found that 17.3% of code in a training dataset had newer versions that could have been incorporated. Therefore, for vulnerabilities with publication dates within an LLM's KCD, the training set of LLMs may be exposed to selective data or conflicting data about a given vulnerability. For vulnerabilities post-KCD, LLMs may not

TABLE VIII
PRE- VS POST-CUTOFF WELCH'S T-TEST SUMMARY (WELCH'S); SIGNIFICANT AT $\alpha = 0.10$ BUT NOT SIGNIFICANT AT $\alpha = 0.05$.

Metric	**Value**
Unique LLMs	3
Unique Prompting Techniques	12
Unique SSVC Decision Points	4
Unique LLM-PT-SDP combinations	144
Pre-cutoff LLM-PT-SDP combinations (n)	144
Post-cutoff LLM-PT-SDP combinations (n)	144
Pre-KCD harmonic mean F1-score mean (SD)	0.3926 (0.1999)
Post-KCD harmonic mean F1-score mean (SD)	0.4367 (0.2191)
Mean diff (post − pre)	0.0441
Welch's t (post − pre)	1.7848
p-value	0.07536697
95% CI for mean diff	[−0.0045, 0.0928]
Cohen's d (independent)	0.2103

face this problem of conflicting neural network inferences, and can solely focus on making inferences for a new vulnerability based upon prompt data to which the LLM has not been previously exposed. This would be consistent with Khare et al. [59] who noted that LLMs performed well identifying vulnerable code when the given vulnerability did not require further context, attributing the stronger performance "to the fact that these are fairly self-contained and little additional context is needed to detect them".

Therefore, further research is needed to determine the composition of the training datasets of these LLMs. While closed-source LLMs were investigated in this study, there are some insights which suggest Gemini may benefit from a high-quality and domain specific-training dataset, namely the datasets available to its developer Google, such as VirusTotal, Mandiant, and the web directory cache [60], [61]. Nevertheless, the LLMs in this study are proprietary and closed-source and as such, further research would only be possible with greater transparency of the training datasets, allowing researchers to review these datasets to identify instances of corruption and improve these datasets [62]. An alternative approach would be to train an open-source LLM on a training dataset that adheres to best practice principles for training LLMs, that include: (1) a wide and diverse range of reliable and relevant data; (2) accurate, balanced and timely data; (3) accessible, consistent and documented data; and (4) compliant data according to applicable licensing, copyright and data protection legislation [63]. This would facilitate observations of an LLM's ability to assess a vulnerability without the risk of its neural network inferences being hindered by selective and conflicting data. In conjunction, insights into the activation of the pathways within the LLM neural networks would be beneficial. For example, research by Azaria et al. [64] developed SAPLMA, a method for determining the truthfulness of LLMs by analysing neural network patterns. Research in determining LLM confidence in its own answers may help with workloads for SOC analysts. In Gligorić et al. [65], the authors noted that LLMs can provide reliable confidence scores of their responses. In a SOC setting, such insights could allow analysts to rely on LLM responses with high truthfulness and confidence scores, while further investigating vulnerabilities where the LLMs indicated low truthfulness or confidence scores.

Takeaway: LLMs tended to perform better on vulnerabilities published after LLM knowledge cutoff dates. Results show that Gemini Flash 1.5 is the strongest performer overall, with Chain of Thought and Few Shot PTs generally yielding the best outcomes, particularly for the Exploitation SDP.

VI. LIMITATIONS

Determining Ground Truth. As noted by Foody [66], ground truth data is assumed to be correct for studies, even though this may not be the case. Industry research indicates that vulnerability data may suffer from a range of issues, including: (1) misattribution of a vulnerability to the wrong part of the software; (2) disagreements between security researchers and vendors on the validity of a vulnerability; and (3) unavailable vulnerability data [67]. When evaluating a model's performance therefore, a model's responses may be labeled as incorrect when they are correct and vice versa. While using Vulnrichment for this study may be an acceptable ground truth, given it is curated by a reputable authority like CISA, it may not be free from errors. An example of the divergence between the Vulnrichment ground truth used for this study and publicly available data curated by other reputable entities in the cybersecurity field is CVE-2019-2861. While Vulnrichment assigns an Exploitation value of *None*[2], the entry in Exploit-DB indicates at minimum a *Proof of Concept*[3]. Disagreements between security researchers are not uncommon [68], with researchers reasonably arriving at different conclusions for a given vulnerability [69], and appear to be an inherent limitation of the vulnerability prioritization field.

Data Licensing. Vulnerability datasets tend to be permissive, allowing use for research purposes. For example, the Vulnrichment dataset is Creative Commons [70], the NVD takes a custom "As Is" license [71], and the Exploit-DB dataset is made available under GNU General Public License (GPL) Version 2 [72]; requirements typically include attribution and release of derivative works. However, some of the vulnerability datasets have restrictive terms of use [73] and as such, would not be suitable for a published academic study. If access to

[2]https://github.com/cisagov/Vulnrichment/blob/fb7d3feaabb7e8b4a2de53be5908088c2ee06d40/2019/2xxx/CVE-2019-2861.json

[3]https://www.exploit-db.com/exploits/47196

these proprietary datasets could be secured, repeating this study to include these proprietary, and ideally high-quality, datasets may also improve LLM performance.

VII. Conclusion and Future Research

When evaluating LLM and PT performance by KCD, a surprising result was the performance of vulnerabilities published after LLM KCD. On the F1-score metric, Gemini Flash 1.5 scored 0.82 on the Exploitation SDP with Chain of Thought. One explanatory hypothesis is selective or conflicting vulnerability data, which could result in lower performance of LLMs for vulnerabilities published pre-KCD. As such, while select LLM and PT combinations have shown moderate performance on recommending prioritizations, SOC teams need to be aware of the limitations of LLMs for vulnerability prioritization, namely the potential for compromised training data to hamper response efforts. Future work needs to focus on optimizing LLM training datasets for vulnerability prioritization. In conjunction with this, methods that facilitate transparency in neural network inferences would be key, as this would provide insights on what sections of the data are contributing to correct outcomes and conversely, which are degrading LLM performance.

References

[1] CrowdStrike, "2024 state of application security report," 2024, accessed on March 24, 2025. [Online]. Available: https://www.crowdstrike.com/en-us/2024-state-of-application-security-report/

[2] M. Fu, C. Tantithamthavorn, T. Le, Y. Kume, V. Nguyen, D. Phung, and J. Grundy, "Aibughunter: A practical tool for predicting, classifying and repairing software vulnerabilities," *Empirical Software Engineering*, vol. 29, no. 1, p. 4, 2024.

[3] S. Elder, M. R. Rahman, G. Fringer, K. Kapoor, and L. Williams, "A survey on software vulnerability exploitability assessment," *ACM Computing Surveys*, vol. 56, no. 8, pp. 1–41, 2024.

[4] S. Zhao, J. Zhu, and J. Peng, "Software vulnerability mining and analysis based on deep learning," *Computers, Materials and Continua*, vol. 80, no. 2, pp. 3263–3287, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1546221824005319

[5] J. Jisoo, S. Jung, M. Ahn, D. Kim, J. Youn, and D. Shin, "Research on quantitative prioritization techniques for selecting optimal security measures," *IEEE Access*, 2024.

[6] L. Alzubaidi, J. Bai, A. Al-Sabaawi, J. Santamaría, A. S. Albahri, B. S. N. Al-Dabbagh, M. A. Fadhel, M. Manoufali, J. Zhang, A. H. Al-Timemy *et al.*, "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications," *Journal of Big Data*, vol. 10, no. 1, p. 46, 2023.

[7] Y. Guo, S. Bettaieb, and F. Casino, "A comprehensive analysis on software vulnerability detection datasets: trends, challenges, and road ahead," *International Journal of Information Security*, vol. 23, no. 5, pp. 3311–3327, 2024.

[8] Z. Kan, S. McFadden, D. Arp, F. Pendlebury, R. Jordaney, J. Kinder, F. Pierazzi, and L. Cavallaro, "Tesseract: Eliminating experimental bias in malware classification across space and time (extended version)," *arXiv preprint arXiv:2402.01359*, 2024.

[9] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[10] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn, "A systematic evaluation of large language models of code," in *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, 2022, pp. 1–10.

[11] Y. Chen, M. Cui, D. Wang, Y. Cao, P. Yang, B. Jiang, Z. Lu, and B. Liu, "A survey of large language models for cyber threat detection," *Computers & Security*, vol. 145, p. 104016, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824003213

[12] A. Souly, J. Rando, E. Chapman, X. Davies, B. Hasircioglu, E. Shereen, C. Mougan, V. Mavroudis, E. Jones, C. Hicks, N. Carlini, Y. Gal, and R. Kirk, "Poisoning attacks on llms require a near-constant number of poison samples," 2025. [Online]. Available: https://arxiv.org/abs/2510.07192

[13] H. Bansal, A. Hosseini, R. Agarwal, V. Q. Tran, and M. Kazemi, "Smaller, weaker, yet better: Training llm reasoners via compute-optimal sampling," *arXiv preprint*, October 2024, google DeepMind, UCLA, Mila.

[14] J. Zhang, D. Qiao, M. Yang, and Q. Wei, "Regurgitative training: The value of real data in training large language models," 2024. [Online]. Available: https://arxiv.org/abs/2407.12835

[15] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey," *ACM computing surveys (CSUR)*, vol. 50, no. 4, pp. 1–36, 2017.

[16] J. A. Harer, L. Y. Kim, R. L. Russell, O. Ozdemir, L. R. Kosta, A. Rangamani, L. H. Hamilton, G. I. Centeno, J. R. Key, P. M. Ellingwood *et al.*, "Automated software vulnerability detection with machine learning," *arXiv preprint arXiv:1803.04497*, 2018.

[17] C. Tanga, D. A. Madhukar Mulpuri, P. Hemalatha, G. Singh, T. Pattewar, and N. Parveen, "Advanced deep learning techniques for information security vulnerability detection using machine learning."

[18] S. Hore, A. Shah, and N. D. Bastian, "Deep vulman: A deep reinforcement learning-enabled cyber vulnerability management framework," *Expert Systems with Applications*, vol. 221, p. 119734, 2023.

[19] D. M. Divakaran and S. T. Peddinti, "Large language models for cybersecurity: New opportunities," *IEEE Security Privacy*, pp. 2–9, 2024.

[20] S. Shreyashree, P. Sunagar, S. Rajarajeswari, and A. Kanavalli, "A literature review on bidirectional encoder representations from transformers," in *Inventive Computation and Information Technologies*, S. Smys, V. E. Balas, and R. Palanisamy, Eds. Singapore: Springer Nature Singapore, 2022, pp. 305–320.

[21] M. R. Shahid and H. Debar, "Cvss-bert: Explainable natural language processing to determine the severity of a computer security vulnerability from its description," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 1600–1607.

[22] J. Yin, M. Tang, J. Cao, and H. Wang, "Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description," *Knowledge-Based Systems*, vol. 210, p. 106529, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120306584

[23] S. S. Das, E. Serra, M. Halappanavar, A. Pothen, and E. Al-Shaer, "V2w-bert: A framework for effective hierarchical multiclass classification of software vulnerabilities," in *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2021, pp. 1–12.

[24] E. Aghaei, E. Al-Shaer, W. Shadid, and X. Niu, "Automated CVE analysis for threat prioritization and impact prediction." [Online]. Available: http://arxiv.org/abs/2309.03040

[25] G. Yenduri, M. Ramalingam, G. C. Selvi, Y. Supriya, G. Srivastava, P. K. R. Maddikunta, G. D. Raj, R. H. Jhaveri, B. Prabadevi, W. Wang, A. V. Vasilakos, and T. R. Gadekallu, "Gpt (generative pre-trained transformer)— a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions," *IEEE Access*, vol. 12, pp. 54 608–54 649, 2024.

[26] O. Oniagbi, A. Hakkala, and I. Hasanov, "Evaluation of llm agents for the soc tier 1 analyst triage process," Master's thesis, University of Turku, Department of Computing, 2024.

[27] S. Mitra, S. Neupane, T. Chakraborty, S. Mittal, A. Piplai, M. Gaur, and S. Rahimi, "LOCALINTEL: Generating organizational threat intelligence from global and local cyber knowledge." [Online]. Available: http://arxiv.org/abs/2401.10036

[28] B. Atil, A. Chittams, L. Fu, F. Ture, L. Xu, and B. Baldwin, "LLM stability: A detailed analysis with some surprises." [Online]. Available: http://arxiv.org/abs/2408.04667

[29] "Sample size calculator help," [Online; accessed 2025-01-16]. [Online]. Available: https://www.abs.gov.au/websitedbs/D3310114.nsf/home/Sample+Size+Calculator+Help

[30] B. Ruan, J. Liu, W. Zhao, and Z. Liang, "Vulzoo: A comprehensive vulnerability intelligence dataset," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2334–2337. [Online]. Available: https://doi.org/10.1145/3691620.3695345

[31] C. Tony, N. E. Díaz Ferreyra, M. Mutas, S. Dhif, and R. Scandariato, "Prompting techniques for secure code generation: A systematic investigation," *ACM Trans. Softw. Eng. Methodol.*, Mar. 2025, just Accepted. [Online]. Available: https://doi.org/10.1145/3722108

[32] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, and E. Chi, "Least-to-most prompting enables complex reasoning in large language models." [Online]. Available: http://arxiv.org/abs/2205.10625

[33] L. Viganò and D. Magazzeni, "Explainable security," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, Sep. 2020, pp. 293–300.

[34] Models. [Online]. Available: https://openrouter.ai

[35] "Gpt 4-o mini vs claude 3 haiku vs gemini 1.5 flash: Small language model pricing considerations," [Online; accessed 2025-01-31]. [Online]. Available: https://www.vantage.sh/blog/gpt-4o-small-vs-gemini-1-5-flash-vs-claude-3-haiku-cost

[36] M. S. Kingsley, *Comparing AWS, Azure, and GCP*. Cham: Springer International Publishing, 2024, pp. 381–393. [Online]. Available: https://doi.org/10.1007/978-3-031-33669-0_12

[37] Anthropic claude - models in amazon bedrock - AWS. [Online]. Available: https://aws.amazon.com/bedrock/claude/

[38] Azure OpenAI service – advanced language models | microsoft azure. [Online]. Available: https://azure.microsoft.com/en-us/products/ai-services/openai-service

[39] Gemini for google cloud: your AI-powered assistant. [Online]. Available: https://cloud.google.com/products/gemini

[40] "Api parameters — configure openrouter api requests — openrouter — documentation," [Online; accessed 2025-04-17]. [Online]. Available: https://openrouter.ai/docs/api-reference/parameters#temperature

[41] Vellum, "Llm parameter guide," https://www.vellum.ai/llm-parameters-guide.

[42] "cisagov/vulnrichment," original-date: 2024-04-24T16:15:29Z. [Online]. Available: https://github.com/cisagov/vulnrichment
[43] B. Edwards. Do we need yet another vulnerability scoring system? for SSVC, that's a YASS. [Online]. Available: https://www.bitsight.com/blog/do-we-need-yet-another-vulnerability-scoring-system-ssvc-thats-yass
[44] G. Baran. CISA announced vulnrichment : Project to enrich CVE records. [Online]. Available: https://cybersecuritynews.com/cisa-announced-vulnrichment/
[45] F-score: What are accuracy, precision, recall, and f1 score? — klu. [Online]. Available: https://klu.ai/glossary/accuracy-precision-recall-f1
[46] J. Komić, *Harmonic Mean*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 622–624. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_645
[47] J. Kim and J. Lee, "Strategic data ordering: Enhancing large language model performance through curriculum learning," *arXiv preprint arXiv:2405.07490*, 2024.
[48] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," ser. ICML'23. JMLR.org, 2023.
[49] M. Roberts, H. Thakur, C. Herlihy, C. White, and S. Dooley, "To the cutoff... and beyond? a longitudinal perspective on llm data contamination," in *The Twelfth International Conference on Learning Representations*, 2023.
[50] M. T. Alam, D. Bhusal, L. Nguyen, and N. Rastogi, "Ctibench: A benchmark for evaluating llms in cyber threat intelligence," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 50 805–50 825. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/5acd3c628aa1819fbf07c39ef73e7285-Paper-Datasets_and_Benchmarks_Track.pdf
[51] M. Rahman, S. Khatoonabadi, and E. Shihab, "Beyond synthetic benchmarks: Evaluating llm performance on real-world class-level code generation," 2025. [Online]. Available: https://arxiv.org/abs/2510.26130
[52] J. Cao, W. Zhang, and S.-C. Cheung, "Concerned with data contamination? assessing countermeasures in code language model," 2024. [Online]. Available: https://arxiv.org/abs/2403.16898
[53] S. Iskander, S. Tolmach, O. Shapira, N. Cohen, and Z. Karnin, "Quality matters: Evaluating synthetic data for tool-using LLMs," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 4958–4976. [Online]. Available: https://aclanthology.org/2024.emnlp-main.285/
[54] J. Dodge, M. Sap, A. Marasović, W. Agnew, G. Ilharco, D. Groeneveld, M. Mitchell, and M. Gardner, "Documenting large webtext corpora: A case study on the colossal clean crawled corpus," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1286–1305. [Online]. Available: https://aclanthology.org/2021.emnlp-main.98/
[55] J. Cheng, M. Marone, O. Weller, D. Lawrie, D. Khashabi, and B. V. Durme, "Dated data: Tracing knowledge cutoffs in large language models." [Online]. Available: http://arxiv.org/abs/2403.12958
[56] E. Klang, D. Apakama, E. E. Abbott *et al.*, "A strategy for cost-effective large language model use at health system-scale," *npj Digital Medicine*, vol. 7, p. 320, 2024. [Online]. Available: https://doi.org/10.1038/s41746-024-01315-1
[57] A. M. Algarni, "The historical relationship between the software vulnerability lifecycle and vulnerability markets: Security and economic risks," *Computers*, vol. 11, no. 9, 2022. [Online]. Available: https://www.mdpi.com/2073-431X/11/9/137
[58] M. Jahanshahi and A. Mockus, "Cracks in the stack: Hidden vulnerabilities and licensing risks in llm pre-training datasets," in *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, 2025, pp. 104–111.
[59] 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.
[60] T. Plumb, "New gemini-powered google threat intelligence platform fuses data from mandiant, virustotal — venturebeat," 5 2024, [Online; accessed 2025-04-14]. [Online]. Available: https://venturebeat.com/security/new-gemini-powered-google-threat-intelligence-platform-fuses-data-from-mandiant-virustotal/
[61] F. Ferrag and I. Bentounsi, "The use of artificial intelligence in academic translation tasks case study of chat gpt, claude and gemini," *Ziglobitha,(2)*, pp. 173–192, 2024.
[62] O. Sainz, J. Campos, I. García-Ferrero, J. Etxaniz, O. L. de Lacalle, and E. Agirre, "NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 10 776–10 787. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.722/
[63] X. Yu, Z. Zhang, F. Niu, X. Hu, X. Xia, and J. Grundy, "What makes a high-quality training dataset for large language models: A practitioners' perspective," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 656–668.
[64] A. Azaria and T. Mitchell, "The internal state of an LLM knows when it's lying," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 967–976. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.68/
[65] K. Gligorić, T. Zrnic, C. Lee, E. J. Candès, and D. Jurafsky, "Can unconfident llm annotations be used for confident conclusions?" *arXiv preprint arXiv:2408.15204*, 2024.
[66] G. M. Foody, "Ground truth in classification accuracy assessment: Myth and reality," *Geomatics*, vol. 4, no. 1, pp. 81–90, 2024. [Online]. Available: https://www.mdpi.com/2673-7418/4/1/5
[67] 2024 dependency management report | endor labs. [Online]. Available: https://www.endorlabs.com/lp/2024-dependency-management-report#full-research
[68] H. Holm and K. K. Afridi, "An expert-based investigation of the common vulnerability scoring system," *Computers Security*, vol. 53, pp. 18–30, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404815000620
[69] V. Badhwar, "How should i prioritize software vulnerabilities? — blog — endor labs," July 2023, [Online; accessed 2025-01-29]. [Online]. Available: https://www.endorlabs.com/learn/cve-vulnerability-epss-ssvc-reachability-vex#svcc-a-decision-tree-tool
[70] "cisagov/vulnrichment," original-date: 2024-04-24T16:15:29Z. [Online]. Available: https://github.com/cisagov/vulnrichment
[71] NVD legal disclaimer. [Online]. Available: https://nvd.nist.gov/general/legal-disclaimer
[72] LICENSE.md · main · exploit-DB / exploits + shellcode + GHDB · GitLab. [Online]. Available: https://gitlab.com/exploit-database/exploitdb/-/blob/main/LICENSE.md
[73] Terms of service. [Online]. Available: https://snyk.io/policies/terms-of-service/
[74] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=gEZrGCozdqR
[75] Prompt engineering for generative AI | machine learning. [Online]. Available: https://developers.google.com/machine-learning/resources/prompt-eng
[76] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html
[77] G. Kim, P. Baldi, and S. McAleer, "Language models can solve computer tasks," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2023.
[78] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-refine: Iterative refinement with self-feedback," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 46 534–46 594. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf
[79] X. Jiang, Y. Dong, L. Wang, Z. Fang, Q. Shang, G. Li, Z. Jin, and W. Jiao, "Self-planning code generation with large language models," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 7, Sep. 2024. [Online]. Available: https://doi.org/10.1145/3672456
[80] Z. Yu, L. He, Z. Wu, X. Dai, and J. Chen. Towards better chain-of-thought prompting strategies: A survey. [Online]. Available: https://arxiv.org/abs/2310.04959v1
[81] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.
[82] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.
[83] J. Chen, L. Chen, H. Huang, and T. Zhou, "When do you need chain-of-thought prompting for chatgpt?" 2023. [Online]. Available: https://arxiv.org/abs/2304.03262
[84] T. Ahmed and P. Devanbu, "Better patching using llm prompting, via self-consistency," in *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '23. IEEE Press, 2024, p. 1742–1746. [Online]. Available: https://doi.org/10.1109/ASE56229.2023.00065
[85] A. Lampinen, I. Dasgupta, S. Chan, K. Mathewson, M. Tessler, A. Creswell, J. McClelland, J. Wang, and F. Hill, "Can language models learn from explanations in context?" in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Association for Computational Linguistics, pp. 537–563. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.38

Appendix A
Prompting Categories & Techniques

TABLE IX
Prompting Techniques and their categories used in this study. Adapted from Tony et al. [31] (cf. §III-D).

Prompting Category & Technique	**Description**
Root	Instructs an LLM to complete a task with varying numbers of examples: Zero-shot (no examples), One-shot (one example), and Few-shot (multiple examples).
Zero-shot (ZS)	Provides no supporting information or examples, forcing the LLM to rely on prior training. Performance tends to be poorer than Few-shot prompting, but appropriate when vulnerability data is limited [74].
One-shot (OS) and Few-shot (FS)	Include examples as part of instruction; One-shot provides one example, Few-shot provides several [75]. Reduces need for task-specific data while still requiring some exemplars [76].
Refinement-based	Instructs an LLM to iterate, review, and provide feedback on responses before returning optimized output.
Recursive Criticism and Improvement (RCI)	Three-step approach: (1) original prompt, (2) critique prompt for critical analysis, (3) improvement prompt based on analysis [77]. Helps LLMs critique and regenerate responses.
Self-refine (SR)	Instructions to generate draft response, provide feedback, and improve iteratively until satisfied or reaching iteration limit [78]. Requires iteration limits for time-sensitive environments.
Decomposition-based	Breaks complex tasks into manageable subtasks, processes them separately, then reconstitutes for optimized response.
Self-planning (SP)	Two-part process: (1) LLM generates plan for task completion, (2) uses generated plan to complete task [79]. Minimal user input required as LLM handles problem breakdown.
Reasoning-based	Instructs LLM to generate, explain, and follow lines of reasoning to develop responses.
Chain-of-Thought (CoT)	Incorporates intermediate steps for progressive reasoning with demonstrations and instructions [80]. Improves performance on arithmetic, commonsense, and symbolic reasoning [81].
Zero-shot Chain-of-Thought (ZSCOT)	Encourages implicit reasoning process [82]. Simple instruction ”let’s think step by step” improved GPT-3 math reasoning from 17.7% to 78.7% [83].
Self-consistency (SC)	Generates multiple responses, uses Model Picks variant to determine most frequent response [84]. Helps analysts working to tight deadlines.
Few-shot with Explanation (FSWE)	Provides input/output exemplars with reasoning explanations. LLM generates answer with explanation using template [85]. Shows improved performance over Zero-shot and Few-shot.
Priming	Provides context for LLM to process subsequently provided tasks.
Persona Pattern (PP)	Instructs LLM to adopt specific person or role viewpoint before task completion [31]. May benefit security engineering environments for vulnerability prioritization.
Memetic Proxy (MP)	Instructs LLM to assume given scenario or situation rather than specific role [31]. Priming feature could improve vulnerability prioritization ability.

APPENDIX B
PERFORMANCE METRICS

TABLE X
HARMONIC MEANS OF F1-SCORES ACROSS THE THREE TRIALS PER LLM, PT, SDP AND KCD COMBINATION (CF. §III).

Prompting Technique	**SSVC Decision Point**	**Claude 3 Haiku**		**Gemini Flash 1.5**		**ChatGPT 4o-Mini**	
		Harmonic Mean of F1-scores		**Harmonic Mean of F1-scores**		**Harmonic Mean of F1-scores**	
		Pre	**Post**	**Pre**	**Post**	**Pre**	**Post**
Chain of Thought	Automatable	0.3106	0.3046	0.6454	0.6831	0.6324	0.6761
	Exploitation	0.0423	0.0479	0.7704	0.8169	0.5892	0.6666
	Mission & Wellbeing	0.2370	0.2134	0.3484	0.3095	0.2423	0.2601
	Technical Impact	0.5075	0.5391	0.5372	0.5176	0.4940	0.4262
Few Shot	Automatable	0.2633	0.2987	0.6003	0.6569	0.6082	0.6259
	Exploitation	0.0440	0.1065	0.6386	0.7862	0.4308	0.4863
	Mission & Wellbeing	0.2421	0.2578	0.3379	0.3090	0.2588	0.2133
	Technical Impact	0.5333	0.6644	0.6353	0.6541	0.5210	0.5234
Few Shot with Explanation	Automatable	0.3211	0.3662	0.5953	0.6358	0.6411	0.6756
	Exploitation	0.0595	0.1352	0.7404	0.7703	0.5360	0.6345
	Mission & Wellbeing	0.2513	0.2751	0.3300	0.3005	0.2721	0.2624
	Technical Impact	0.6084	0.6811	0.6091	0.6358	0.5494	0.4829
Memetic Proxy	Automatable	0.1659	0.1265	0.4865	0.6271	0.5966	0.6252
	Exploitation	0.0275	0.0858	0.5398	0.5547	0.0964	0.2076
	Mission & Wellbeing	0.2235	0.2383	0.4677	0.3902	0.2489	0.2486
	Technical Impact	0.4122	0.5635	0.6284	0.6399	0.5323	0.6728
One Shot	Automatable	0.2504	0.2836	0.5980	0.6805	0.6175	0.6467
	Exploitation	0.0334	0.0338	0.6199	0.7524	0.4504	0.5534
	Mission & Wellbeing	0.2325	0.2384	0.3428	0.3064	0.2551	0.2307
	Technical Impact	0.5307	0.5828	0.5641	0.5596	0.5434	0.4814
Persona Pattern	Automatable	0.1460	0.0897	0.5273	0.6815	0.5861	0.6204
	Exploitation	0.0256	0.0660	0.5891	0.6468	0.1070	0.2278
	Mission & Wellbeing	0.2165	0.2454	0.4118	0.3740	0.2427	0.2319
	Technical Impact	0.4105	0.5498	0.6245	0.6680	0.5175	0.6428
Recursive Criticism	Automatable	0.1805	0.0882	0.5223	0.6597	0.6005	0.6481
	Exploitation	0.0257	0.0457	0.5621	0.6083	0.1180	0.2021
	Mission & Wellbeing	0.2314	0.2525	0.4255	0.3804	0.2307	0.2323
	Technical Impact	0.3769	0.5568	0.6305	0.6584	0.5038	0.6232
Self Consistency	Automatable	0.1651	0.0536	0.5289	0.6452	0.5875	0.5993
	Exploitation	0.0283	0.0116	0.5485	0.6076	0.1169	0.1968
	Mission & Wellbeing	0.2342	0.2388	0.4184	0.3857	0.2299	0.2340
	Technical Impact	0.3842	0.4928	0.6458	0.6360	0.5075	0.6681
Self Planning	Automatable	0.1339	0.1871	0.5561	0.6855	0.5891	0.6680
	Exploitation	0.0270	0.1807	0.5746	0.6280	0.1829	0.3434
	Mission & Wellbeing	0.2164	0.2731	0.4070	0.3723	0.2461	0.2542
	Technical Impact	0.4286	0.6173	0.6318	0.6522	0.5433	0.6738
Self Refine	Automatable	0.1790	0.1174	0.5295	0.6463	0.6070	0.6240
	Exploitation	0.0276	0.0948	0.5477	0.5996	0.0859	0.1830
	Mission & Wellbeing	0.2265	0.2813	0.4395	0.3905	0.2300	0.2312
	Technical Impact	0.3988	0.6125	0.6438	0.6552	0.4940	0.6219
Zero Shot	Automatable	0.1734	0.1139	0.5345	0.6441	0.6026	0.6369
	Exploitation	0.0288	0.1137	0.5610	0.5796	0.1048	0.2251
	Mission & Wellbeing	0.2191	0.2474	0.4325	0.3651	0.2301	0.2331
	Technical Impact	0.4140	0.6202	0.6328	0.6570	0.5125	0.6518
Zero Shot Chain of Thought	Automatable	0.1797	0.1287	0.5333	0.6435	0.6002	0.6461
	Exploitation	0.0264	0.1266	0.5612	0.6024	0.1212	0.2165
	Mission & Wellbeing	0.2336	0.2630	0.4284	0.3832	0.2342	0.2399
	Technical Impact	0.4115	0.6308	0.6256	0.6646	0.5284	0.6587