Using LLMs for Security Advisory Investigations: How Far Are We?

1st Bayu Fedra Abdullah
Informatics Engineering
Universitas Muhammadiyah Surakarta
Surakarta, Indonesia
L200200115@student.ums.ac.id

2nd Yusuf Sulistyo Nugroho
Informatics Engineering
Universitas Muhammadiyah Surakarta
Surakarta, Indonesia
yusuf.nugroho@ums.ac.id

3rd Brittany Reid
Information Science
Nara Institute of Science and Technology
Nara, Japan
brittany.reid@naist.ac.jp

4th Raula Gaikovina Kula
Graduate School of IST
University of Osaka
Osaka, Japan
raula-k@ist.osaka-u.ac.jp

5th Kazumasa Shimari
Information Science
Nara Institute of Science and Technology
Nara, Japan
k.shimari@is.naist.jp

6th Kenichi Matsumoto
Information Science
Nara Institute of Science and Technology
Nara, Japan
matumoto@is.naist.jp

***Abstract*—Large Language Models (LLMs) are increasingly used in software security, but their trustworthiness in generating accurate vulnerability advisories remains uncertain. This study investigates the ability of ChatGPT to (1) generate plausible security advisories from CVE-IDs, (2) differentiate real from fake CVE-IDs, and (3) extract CVE-IDs from advisory descriptions. Using a curated dataset of 100 real and 100 fake CVE-IDs, we manually analyzed the credibility and consistency of the model's outputs. The results show that ChatGPT generated plausible security advisories for 96% of given input real CVE-IDs and 97% of given input fake CVE-IDs, demonstrating a limitation in differentiating between real and fake IDs. Furthermore, when these generated advisories were reintroduced to ChatGPT to identify their original CVE-ID, the model produced a fake CVE-ID in 6% of cases from real advisories. These findings highlight both the strengths and limitations of ChatGPT in cybersecurity applications. While the model demonstrates potential for automating advisory generation, its inability to reliably authenticate CVE-IDs or maintain consistency upon re-evaluation underscores the risks associated with its deployment in critical security tasks. Our study emphasizes the importance of using LLMs with caution in cybersecurity workflows and suggests the need for further improvements in their design to improve reliability and applicability in security advisory generation.**

***Index Terms*—advisory, chatgpt, cve id, security, vulnerability**

I. Introduction

In recent years, software vulnerabilities have become the main focus for software developers, especially due to the increase in cybercrimes [1]. Developers may attempt to mitigate vulnerabilities [2], but hackers continue exploiting weaknesses to attack users and steal sensitive data [3]. To aid the secure development of software, the Common Vulnerabilities and Exposures (CVE) system maintains a database[1] of publicly disclosed vulnerabilities in software packages, labeling each with a unique CVE-ID and providing advisory information. CVEs are a common way of disseminating security information within the software community, and thus recognizing and identifying them is important to the security of software.

With the rise of generative AI, developers increasingly turn to Large Language Models (LLMs) for assistance in various domains, such as health [4], education [5], programming [6], including software security [7]. Despite their growing popularity, concerns remain regarding the accuracy and reliability of outputs [8]. In cybersecurity, some practitioners use tools like ChatGPT to retrieve CVE-IDs or generate advisories, though LLMs' trustworthiness in this context is uncertain. While LLMs show promise in vulnerability detection [9] and automated security bug repair [10], their tendency to hallucinate information [11] raises risks. Relying on AI-generated advisories without verification may create false security assumptions or even delay critical mitigation efforts. Although LLMs are not designed as database query systems, our study reflects real-world misuse scenarios where users treat them as trusted security sources.

This study investigates the capabilities of LLMs in generating accurate and reliable security advisories, identifying inconsistencies in vulnerability data, and detecting fake CVE-IDs. Prior research has examined LLMs in vulnerability detection and code security analysis. However, limited work has explored their reliability in security advisory generation. To address this gap, we formulate the following research questions:

- *RQ_1: How trustworthy are the LLM-generated security advisories, given a known CVE-ID as input?*
- *RQ_2: Do LLMs have the capability to detect fake CVE-IDs?*
- *RQ_3: Can LLMs accurately and consistently produce CVE-ID from a provided advisory description?*

To answer these questions, we constructed a dataset com-

[1]https://www.cve.org/

prising 100 real CVE-IDs and their corresponding security advisories from the GitHub Security Advisory (GHSA) database,[2] capturing key metadata such as CVE-ID, title, severity, and affected products. To evaluate the LLM's ability to detect fabricated vulnerabilities, we generated an additional 100 fake CVE-IDs using a Python script and verified their absence in public vulnerability databases. Both real and fake CVE-IDs were provided to ChatGPT, and the generated advisories were analyzed for accuracy, consistency, and reliability.

The findings highlight both the potential and the risks of utilizing LLMs for security advisory generation. While LLMs show impressive generative capabilities, their tendency to produce misinformation raises concerns about their deployment in critical cybersecurity tasks. Our study highlights the importance of validating AI-generated security advisories before adoption and serves as a foundation for future research aimed at improving LLM reliability in cybersecurity applications.

II. Related Work

A. LLMs for Software Engineering

Since the first release of the LLM, numerous studies have been conducted on its implementation in software engineering. Studies reveal that LLMs or generative AI tools like Google Bard (now called Gemini), ChatGPT, or CoPilot have proven that they can improve productivity in software engineering [12], [13]. The use of LLMs has also benefited many software engineering tasks. For example, LLMs can accelerate the development cycles and reduce the time spent on repetitive coding tasks [14], increasing productivity [15], improving software quality and development efficiency [16], and even the LLMs can help in identifying the software vulnerability-related subtle patterns [17].

Despite the advantages of the use of LLMs in software engineering, it has been unknown how far the LLMs can distinguish between real or fake vulnerabilities and their descriptions.

B. Identifying Software Vulnerabilities

As one of the weaknesses in computer security, vulnerabilities are often used by an attacker to exploit the system to perform unauthorized actions [18], [19]. To mitigate this, developers sometimes notify other programmers by citing a specific vulnerability identifier in code comments to indicate that the code may contain vulnerability [20]. Other strategies were also proposed in several researches, such as the development of automatic identification of vulnerable versions for CVE [21], the implementation of TF-IDF and Doc2Vec for automatically tracing the related CAPEC-IDs from CVE-ID [22], and a machine-learning based technique to assign pertinent CWE identifiers to new CVE entries [23]. These trigger software developers to make efforts and help them in addressing security issues consistently.

[2]https://github.com/advisories

In addition, some prior works were conducted by focusing on the vulnerability descriptions. For example, a study proposed a novel method to compose CVE descriptions by extracting some vulnerability aspects from ExploitDB [24]. Their results indicate that the method can achieve high accuracy in composing CVE descriptions. Another study augmented the vulnerability description by scrapping third-party references (hyperlinks) [25]. The findings have shown potential regarding summary fluency, completeness, correctness, and understanding. Due to the increasing of AI implementations in the software engineering field, the CVE description can also be potentially written with the help of AI, such as ChatGPT. Thus, in this paper, we study to what extent an LLM can generate the security advisory description.

III. Study Setting

A. Research Questions

To guide our study, we formulated the following three research questions and their motivations.

RQ_1: How trustworthy are the LLM-generated security advisories, given a known CVE-ID as input?

Motivation: This RQ arises from the increasing use of LLMs like ChatGPT in cybersecurity. Many developers and security practitioners rely on these models to generate security advisories and identify vulnerabilities. However, the accuracy of AI-generated advisories remains uncertain. Since cybersecurity decisions require precise and reliable information, any incorrect or misleading advisory could result in overlooked vulnerabilities or a false sense of security.

In addition, LLMs are known to hallucinate [11], generating plausible but false information. In the case of security advisories, this could mean fabricated vulnerability details or incorrect mitigation steps. Since non-expert users may not have the expertise to validate AI-generated advisories, it is crucial to assess how trustworthy these outputs are. By investigating the trustworthiness of LLM-generated security advisories, we aim to evaluate whether AI models can serve as a reliable source of cybersecurity information or if their use introduces risks that could undermine software security efforts.

RQ_2: Do LLMs have the capability to detect fake CVE-IDs?

Motivation: While databases like CVE and GHSA provide authoritative records of known vulnerabilities, developers and security practitioners may still turn to AI tools like ChatGPT for quick information retrieval. However, the ability of LLMs to differentiate real vulnerabilities from fabricated ones remains unclear. If an LLM fails to detect fake CVE-IDs, it could generate misleading security advisories, leading to confusion or even security misconfigurations. This is particularly concerning because attackers or misinformed users could exploit AI-generated misinformation to spread false vulnerabilities or obscure real threats.

Moreover, LLMs are trained on large amounts of publicly available text but lack direct access to up-to-date vulnerability databases. Without real-time verification mechanisms, they

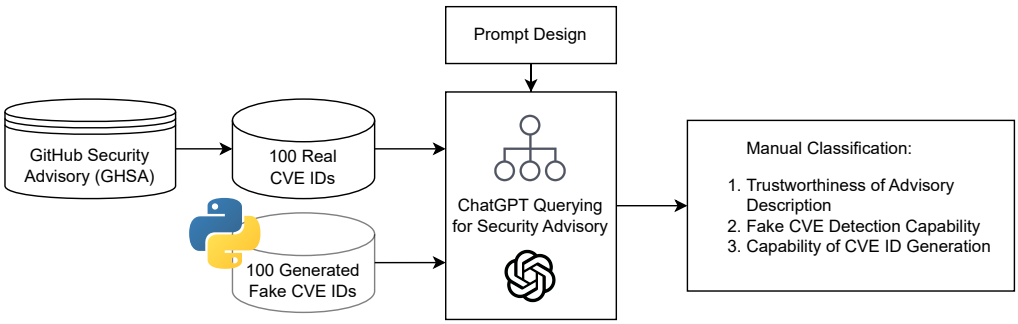

Fig. 1. Overview of the research procedures, covering data collection, prompt design, and manual analyses.

TABLE I
DATASET OVERVIEW

Types of CVE ID	# CVE IDs
Real CVE-IDs	100
Generated CVE-IDs	100
Total	200

may produce CVE-IDs that sound reasonable but are entirely fake or fail to recognize invalid ones. By investigating whether LLMs can effectively detect fake CVE-IDs, this study aims to assess their reliability as a security resource and highlight potential weaknesses in AI-driven vulnerability identification.

RQ$_3$: Can LLMs accurately and consistently produce CVE-ID from a provided advisory description?

Motivation: When analyzing vulnerabilities, correctly associating an advisory with its corresponding CVE-ID is essential for tracking and mitigating security risks. If LLMs can accurately generate CVE-IDs from advisory descriptions, they could serve as useful tools for automating parts of the vulnerability management process. This would benefit developers, security teams, and organizations by improving efficiency in identifying and responding to threats. However, if LLMs generate incorrect CVE-IDs, they risk spreading misinformation and creating confusion within the security community.

This limitation raises concerns about their ability to consistently match advisories with the correct CVE-IDs, especially when dealing with newly disclosed vulnerabilities. If an LLM frequently generates incorrect CVE-IDs, this could undermine trust in the security insights generated by AI. By evaluating the accuracy of LLMs in generating CVE-IDs, this study aims to determine their usefulness in cybersecurity workflows and highlight areas where improvements are necessary to ensure their reliability as an information source.

B. Data Collection

As illustrated in Fig. 1, the entire process of this study includes data collection, fake CVE IDs generation, prompt design, ChatGPT querying, and manual classification.

The dataset used in this study, as presented in Table I, was built through the following 2 main steps:

```
cve_id: CVE-2016-3722
title: Incorrect Authorization in Jenkins Core
affected: Maven org.jenkins-ci.main:jenkins-core
severity: MODERATE
cwe_id: CWE-863
published: 2022-05-14
details: Jenkins before 2.3 and LTS before 1.651.2
allow remote authenticated users with multiple
accounts to cause a denial of service (unable to
login) by editing the "full name".
```

Fig. 2. Example of extracted security advisory from GHSA.

```
cve-2021-6365
cve-2016-13580
cve-2018-8981
cve-2022-13790
cve-2022-6453
```

Fig. 3. Example of fake CVE-IDs.

1) Extracting Security Advisory info from GHSA: To build our dataset, we initially downloaded all GitHub Security Advisory (GHSA) data. We then randomly selected 100 real CVE-IDs and their advisory descriptions by specifying the attributes including CVE-ID, title, affected product, severity, CWE-ID, published date, and details, as exemplified in Fig. 2.

2) Generating Fake CVE-IDs: To understand how far we can rely on LLMs in generating security advisories, we generated 100 random fake CVE-IDs automatically using Python code by following the format `CVE-YYYY-NNNN(N)`. To ensure that the generated CVE-IDs were fake, we cross-checked the identifiers on the NVD Database.[3] If the CVE IDs exist in the database, they were ignored. Otherwise, they were kept for further analysis of fake security advisories. Some generated fake CVE-IDs are shown in Fig. 3.

C. Prompt Design

Prompts using CVE-IDs replicate actual developer behavior observed on platforms like Stack Overflow and Reddit. For instance, a Reddit discussion[4] illustrates how users query ChatGPT regarding known and even reserved CVE details,

[3]https://nvd.nist.gov/

[4]https://www.reddit.com/r/artificial/comments/1345ay8/chatgpt_leaks_reserved_cve_details_should_we_be/

```
You will provide information about the given CVE ID.
Your output should follow these rules:
- Provide the title of the CVE in the "Title" label
- Mention the affected product explicitly and its version
  in the "Affected" label
- Define the severity level in the "Severity" label
- Provide the CWE ID in the "CWE-ID" label
- Give the "Published date" in 'YYYY-MM-DD' format in the
  "Published" label
- Write the description of the given CVE ID in the
  "Description" label
- Return label only without other text
```

Fig. 4. Example of input to LLM, asking for vulnerability advisory descriptions.

TABLE II
THE TRUSTWORTHINESS OF THE LLM-GENERATED ADVISORY DESCRIPTIONS

Category	Descriptions
Reliable	If the generated security advisory is consistent and looks credible.
Unreliable	If the generated security advisory exhibits inconsistencies and seems unreliable.

highlighting its usage in real-world security contexts. Prompts were entered manually into ChatGPT using a structured template, as shown in Fig. 4. In this case, no files or fine-tuning were used. We also observed that ChatGPT's responses were highly sensitive to the phrasing of prompts; for example, including explicit instructions such as "mention the affected product explicitly" produced more detailed information, while omitting such cues often resulted in vague or generic outputs. These findings emphasize the importance of careful prompt design, which we plan to explore further in future work.

D. Manual Classification

The manual classification in this study was conducted by a single evaluator following a structured rubric. For trustworthiness, advisories were labeled as 'Reliable' if they appeared internally consistent and plausible, and 'Unreliable' if they had inconsistencies, such as date mismatches or implausible information. For similarity, the evaluator used a five-level scale ranging from 'Totally Different' to 'Similar,' based on both lexical overlap and semantic coherence. In detail, we describe them in each RQ in Section IV.

E. Appendix

To facilitate the reproducibility of this work, we made our replication package publicly available on https://github.com/bayufedra/Research-NAIST-SE-2024.

IV. RESULTS AND DISCUSSIONS

In this section, we describe the findings of each research question and discuss them comprehensively.

A. RQ_1: How trustworthy are the LLM-generated security advisories, given a known CVE-ID as input?

To address this question, we asked ChatGPT to generate security advisory details according to the given CVE-ID. By using the designed prompt in Fig. 4, we input both real and fake CVE-IDs. All LLM outputs were then manually analyzed to evaluate the accuracy of the generated descriptions.

TABLE III
TRUSTWORTHINESS OF CHATGPT IN GENERATING SECURITY ADVISORY DESCRIPTIONS BASED ON THE GIVEN CVE-IDS

Type of CVE-ID	# CVE-IDs	
	Reliable	Unreliable
Real	96%	4%
Fake	97%	3%

TABLE IV
COMPLIANCE CATEGORIES BETWEEN LLM-GENERATED AND ORIGINAL SECURITY ADVISORY DESCRIPTIONS

Category	Descriptions
Totally Different	If the generated description is entirely unrelated.
Quite Different	If the generated output has some connections but most of the information is different.
Somewhat Different	If the description generated by LLM has significant connections but there are differences in details or structure
Quite Similar	If the output is mostly the same, with only minor or insignificant differences
Similar	If the generated description is exactly the same or fully aligned without any noticeable differences

As described in Table II, the LLM-generated outputs were classified into two categories: *Reliable* and *Unreliable*. We labeled the output *Reliable* if there is a possibility for someone to believe that the security advisory is real. Otherwise, it was labeled *Unreliable*, including inconsistencies. For example, the published year of the advisory is different from the year indicated in the CVE-ID code.

The result, as presented in Table III, shows that ChatGPT mostly generates *reliable* advisories, accounting for 96% and 97% for real and fake CVE-IDs, respectively. This indicates that ChatGPT is remarkably good at generating reliable output, regardless of whether the input is real or not. Thus, it may not be obvious to developers if the output or input is misleading since ChatGPT does not detect whether the CVE-IDs are fake.

Since the trustworthiness of security advisories generated by ChatGPT does not necessarily reflect the accuracy of the associated CVE-IDs, a deeper analysis was required. In this case, we only focused on the 100 randomly chosen real CVE-IDs. Therefore, we examined the similarity between the advisory descriptions of these real CVE-IDs and the descriptions generated by ChatGPT. In this analysis, we manually classified the descriptions into five classes based on the similarity, as described in Table IV.

As can be seen in Fig. 5, it shows that ChatGPT produces

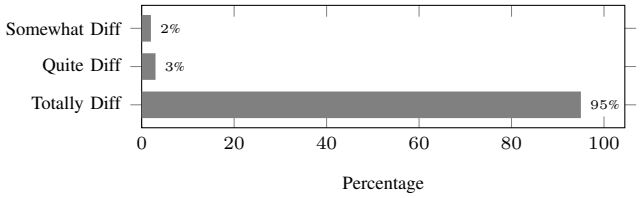

Fig. 5. Similarity distribution of ChatGPT-generated advisories to the original ones.

TABLE V
FAILURE RATE OF CHATGPT TO DETECT FAKE CVE-IDS

Detection Result	# Fake CVE-IDs
Detected	0%
Not detected	100%

95% outputs that totally different from the original advisory descriptions, 3% are quite different, and 2% are somewhat different. Notably, none of the generated advisories are classified as "Quite Similar" or "Similar," highlighting the inability of the model to accurately replicate original advisory content.

These results suggest that although ChatGPT can generate security advisories, its descriptions are often unreliable and lack fidelity to the original CVE advisories. This poses a significant risk, as practitioners relying on LLM-generated advisories may receive misleading or entirely incorrect security information. The high rate of "Totally Different" advisories also underscores the model's tendency to hallucinate content rather than align with verified vulnerability data. Given these findings, it is critical for security professionals to verify AI-generated advisories against trusted sources before relying on them for decision-making in cybersecurity contexts.

B. RQ_2: Do LLMs have the capability to detect fake CVE-IDs?

In this RQ, we examined the ability of ChatGPT to detect its own generated fake CVE-IDs. We inputted each fake ID of the CVE into the model and asked it to detect whether the IDs were real or fake.

As described in Table V, ChatGPT failed to flag any fake CVE-IDs as invalid. However, this does not imply complete model failure but rather reflects a lack of uncertainty indication in LLM responses. This result demonstrates that ChatGPT cannot validate CVE-IDs against an authoritative database. Users who rely on ChatGPT to verify CVE-IDs may unknowingly accept and propagate fabricated vulnerabilities, which could have serious implications for cybersecurity decision-making.

The inability of ChatGPT to differentiate real CVE-IDs from fake ones highlights a fundamental limitation in its knowledge retrieval and fact-checking processes. This limitation poses a significant risk, for example, bad actors could exploit the inability of ChatGPT to detect fakes by spreading false vulnerability reports, potentially causing unnecessary security issues or misleading developers into believing non-existent threats are real. These findings emphasize the importance of integrating external validation mechanisms, before relying on LLM-generated security advisories.

C. RQ_3: Can LLMs accurately and consistently produce CVE-ID from a provided advisory description?

In this question, we used both 100 real CVE advisories extracted from GHSA database, 100 generated advisories of these original real CVE-IDs resulting from RQ_1, and 100 generated advisories of fake CVE-IDs. Subsequently, we asked ChatGPT whether it could identify the CVE-ID code from the advisory text using the designed prompt, as shown in Fig. 6.

```
You will show a CVE-ID based on the given Security
Advisory. Your output must follow these rules:
- Give CVE-ID of the Security Advisory in the "CVE-ID"
  label.
- Return label only without other text.
```

Fig. 6. LLM input to ask for CVE-ID from a given advisory.

TABLE VI
SUCCESS RATE OF CHATGPT IN GENERATING CVE-IDS BASED ON THE REAL ADVISORIES

Type of Correctness	# CVE-IDs
True CVE ID	94%
False CVE ID	6%

First, we asked ChatGPT to generate the CVE-ID based on 100 real advisories. We considered "True" if the generated CVE-ID was fully correct. Otherwise, we marked it as "False" even if it partially matched the true CVE-ID (e.g., correct year but wrong number). While this approach is strict, it ensures clarity, and future work may adopt graded scoring to capture partial correctness. As shown in Table VI, ChatGPT was incapable of generating 6% CVE-IDs.

Next, we asked ChatGPT to identify the CVE-ID based on the advisories generated from the same set of real CVE-IDs. This analysis was conducted to assess its knowledge consistency. We found that, out of 100 advisories, only one had a consistent answer, as shown in Table VII. This suggests that while ChatGPT is capable of generating realistic-sounding security advisories, it struggles to accurately generate unique and valid CVE-IDs.

To add to our understanding of its ability, we further asked ChatGPT with the same scenario as in the previous experiment using the same prompt as designed in Fig. 6. However, in this case, we utilized the 100 generated security advisories based on fake CVE-IDs yielded from RQ_1 as the input. The results, as shown in Table VIII, show that 10% of the CVE-IDs generated by ChatGPT are detected in the database even though the advisory given is false. This further highlights the limitation of the model in accurately generating unique and valid CVE-IDs, even when provided with fabricated input data.

TABLE VII
THE CONSISTENCY OF THE LLM IN GENERATING CVE-IDS FROM ITS OWN GENERATED SECURITY ADVISORY

Consistency	# CVE-IDs
Consistent CVE ID	1%
Inconsistent CVE ID	99%

TABLE VIII
FREQUENCY OF CVE-ID EXISTENCE DETECTION BASED ON FAKE ADVISORIES

Authenticity	# CVE-IDs
Existing CVE ID	10%
Fabricated CVE ID	90%

V. Threats to Validity

One potential threat to validity in this study relates to the temporal nature of LLM capabilities. Our research was conducted between April and June 2024, meaning that all results presented in this paper reflect the state of ChatGPT during that period. Since LLMs continuously evolve through updates and improvements, their performance in generating and verifying security advisories may change over time.

Another threat concerns the prompt design used to query ChatGPT. The effectiveness of LLM responses can be highly sensitive to the wording, structure, and specificity of prompts. Furthermore, our study relied on manual labeling to classify advisory similarities and assess the correctness of generated CVE-IDs. This process might be subjective, as human evaluators may interpret and categorize responses differently. Although we maintained consistency in our labeling approach, variations in judgment among different researchers could introduce bias into the results.

VI. Conclusion

This study evaluated the capabilities of an LLM, specifically ChatGPT in generating credible security advisories and identifying CVE-IDs. We investigated the performance of ChatGPT in identifying 100 real and 100 fake CVE-IDs to gain valuable insights into its limitations in cybersecurity applications.

Our findings reveal the limitations in the ability of ChatGPT to generate accurate and trustworthy security advisories, detect fake CVE-IDs, and correctly assign CVE-IDs to advisories. Although the model can produce realistic-sounding advisories, it shows that the generated descriptions frequently differ from the original ones. ChatGPT also fails to detect fake CVE-IDs, as it consistently identifies them as real. Furthermore, its ability to generate CVE-IDs from advisory descriptions is inconsistent, with a higher percentage of incorrect outputs. These findings suggest that while LLMs can assist in security-related tasks, they should not be relied upon to create or verify security advisory without human supervision. Future research should explore ways to improve the reliability of LLMs in security contexts, such as integrating external validation mechanisms, refining prompt engineering strategies, and improving dataset quality for training security-related models.

References

[1] M. C. Sanchez, J. M. C. de Gea, J. L. Fernandez-Aleman, J. Garceran, and A. Toval, "Software vulnerabilities overview: A descriptive study," *Tsinghua Science and Technology*, vol. 25, no. 2, pp. 270–280, 2019.

[2] H. Nina, J. A. Pow-Sang, and M. Villavicencio, "Systematic mapping of the literature on secure software development," *IEEE Access*, vol. 9, pp. 36852–36867, 2021.

[3] S. Mishra, M. A. Alowaidi, and S. K. Sharma, "Impact of security standards and policies on the credibility of e-government," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2021.

[4] H. Daungsupawong and V. Wiwanitkit, "An innovative approach for treating chronic vaginitis based on AI-driven drug repurposing," *Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika*, vol. 10, no. 1, pp. 30–35, 2024.

[5] T. Adiguzel, M. H. Kaya, and F. K. Cansu, "Revolutionizing education with AI: Exploring the transformative potential of ChatGPT." *Contemporary Educational Technology*, vol. 15, no. 3, 2023.

[6] Y. Liu, T. Le-Cong, R. Widyasari, C. Tantithamthavorn, L. Li, X. B. D. Le, and D. Lo, "Refining chatgpt-generated code: Characterizing and mitigating code quality issues," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 5, pp. 1–26, 2024.

[7] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, "From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy," *IEEE Access*, vol. 11, pp. 80218-80245, 2023.

[8] R. Williams, "Why Google's AI overview gets things wrong," *MIT Technology Review*. [Online]. Available: https://www.technologyreview.com/2024/05/31/1093019/why-are-googles-ai-overviews-results-so-bad/

[9] D. Noever, "Can large language models find and fix vulnerable software?" *arXiv preprint arXiv:2308.10345*, 2023.

[10] H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, "Examining zero-shot vulnerability repair with large language models," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2339–2356.

[11] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1-38, 2023.

[12] P. Brie, N. Burny, A. Sluyters, and J. Vanderdonckt, "Evaluating a large language model on searching for gui layouts," *Proceedings of the ACM on Human-Computer Interaction*, vol. 7, no. EICS, pp. 1–37, 2023.

[13] C. Ebert and P. Louridas, "Generative AI for software practitioners," *IEEE Software*, vol. 40, no. 4, pp. 30–38, 2023.

[14] F. Khomh, "Harnessing predictive modeling and software analytics in the age of LLM-Powered software development (invited talk)," in *Proceedings of the 19th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2023, pp. 1–1.

[15] S. I. Ross, F. Martinez, S. Houde, M. Muller, and J. D. Weisz, "The programmer's assistant: Conversational interaction with a large language model for software development," in *Proceedings of the 28th International Conference on Intelligent User Interfaces*, 2023, pp. 491–514.

[16] S. Yu, C. Fang, Y. Ling, C. Wu, and Z. Chen, "LLM for test script generation and migration: Challenges, capabilities, and opportunities," in *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*. IEEE, 2023, pp. 206–217.

[17] M. D. Purba, A. Ghosh, B. J. Radford, and B. Chu, "Software vulnerability detection using large language models," in *2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2023, pp. 112–119.

[18] N. Munaiah, A. Rahman, J. Pelletier, L. Williams, and A. Meneely, "Characterizing attacker behavior in a cybersecurity penetration testing competition," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–6.

[19] F. Piessens and I. Verbauwhede, "Software security: Vulnerabilities and countermeasures for two attacker models," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 990–999.

[20] Y. S. Nugroho, D. Gunawan, D. A. Puspa Putri, S. Islam, and A. Alhefdhi, "A study of vulnerability identifiers in code comments: Source, purpose, and severity," *Journal of Communications Software and Systems*, vol. 18, no. 2, pp. 165–174, 2022.

[21] L. Bao, X. Xia, A. E. Hassan, and X. Yang, "V-SZZ: Automatic identification of version ranges affected by CVE vulnerabilities," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 2352–2364.

[22] K. Kanakogi, H. Washizaki, Y. Fukazawa, S. Ogata, T. Okubo, T. Kato, H. Kanuka, A. Hazeyama, and N. Yoshioka, "Tracing capec attack patterns from CVE vulnerability information using natural language processing technique," in *Proceedings of the Annual Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2021.

[23] M. Aota, T. Ban, T. Takahashi, and N. Murata, "Multi-label positive and unlabeled learning and its application to common vulnerabilities and exposure categorization," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 988–996.

[24] J. Sun, Z. Xing, H. Guo, D. Ye, X. Li, X. Xu, and L. Zhu, "Generating informative CVE description from exploitdb posts by extractive summarization," *arXiv preprint arXiv:2101.01431*, 2021.

[25] H. Althebeiti and D. Mohaisen, "Enriching vulnerability reports through automated and augmented description summarization," in *International Conference on Information Security Applications*. Springer, 2023.