Detecting Security Vulnerabilities Using a GPT Model

ABSTRACT

After the remarkable debut of ChatGPT and its related APIs, it has been a new area of study that constantly needs to be explored. Across diverse fields, experts are exploring ways to leverage GPT models. Yet, based on our survey, we ascertain that further research is required to find the potential of using the GPT models in detecting security vulnerabilities. In this paper, we present our empirical investigation into GPT-3.5's efficacy as a vulnerability detector. We have gathered 18,896 Java files from the 6 Java open Source Software projects. Using this data, we tasked GPT-3.5 with identifying hidden vulnerabilities related to CWE and CVE. To examine the performance of it, we performed additional comparative work using existing static analyzers. In our preliminary results, we observed both the potential and limitations of using the GPT models to detect vulnerabilities. Our findings show the early results discussing the possibility of the GPT models for security issues and lead to future research directions.

ACM Reference Format:

. 2023. Detecting Security Vulnerabilities Using a GPT Model. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The introduction of ChatGPT offered humanity a new way for advancement. After ChatGPT 3.5's astonishing debut on November 30, 2022., it impacted not only our daily lives but also our academic pursuits. A diverse range of experts is applying ChatGPT or its related GPT API in their respective fields of study [7, 12, 14, 22, 23].

After conducting a survey, we discovered that GPT is being employed in the fields of software assurance as well as other areas of study [14, 22, 23]. The results suggest that there are currently limited empirical studies in progress. The following three studies are not directly related to ensuring security in software but, are noticeable works that are similar to this topic. One study integrated GPT on solving challenging mathematical problems [22]. Their research centers around the integration of GPT-4 with Python to solve complex mathematical problems. Another study was conducted in the field of software engineering [7]. The core concept of this study was to tackle ethical issues concerning privacy, plagiarism, and data security. Moreover, a recent study by Xu et al. [23] suggests that GPT still lags behind human capabilities when answering questions across six distinct aspects: correctness, usefulness, diversity, readability, clarity, and conciseness. Apart from listed academic studies, there has been personal research that was conducted using various types of GPT, which are about figuring out the Myers-Briggs Type Indicator (MBTI) of GPT [12] and its political bias [14].

Meanwhile, limited studies have employed GPT as a tool to tackle vulnerability issues [9, 17]. Among them, one study was conducted by Cheshkov et al. [9]. It is about the evaluation of the ChatGPT model for vulnerability detection. They advanced their research by prompting ChatGPT and its related APIs to identify the top 5 vulnerabilities based on their frequency of occurrence. This narrowed their focus to specific CWEs chosen from their predefined dataset. Consequently, applying this approach to OSS platforms like Github, which inherits a broad spectrum of non-predefined CWEs, is not feasible.

We conducted an empirical study to examine the capability of the GPT model as a vulnerability detector. The model that we used for this experiment is gpt-3.5-turbo-16K. To broaden the scope of our data, we have gathered recent **3,586 independent commits** and **18,896 Java files** from **6 different Open Source Software (OSS) projects [5, 6, 10, 11, 20, 21]**. To ensure no bias from prior training contexts, we gathered commit history spanning from October 1, 2021, to July 18, 2023. Using these collected commits, we queried the GPT API with prepared prompts to gather responses associated with vulnerability detection. Subsequently, we compared these results with those from static analysis tools to validate their performance.

Based on our empirical study, we observed the potential of GPT as a vulnerability detection tool. It identified a substantial number of vulnerabilities within open-source software projects compared to static analyzers. Additionally, it was evident there still were limitations in the GPT model.

2 RELATED WORK

The emergence of GPT has had a significant impact on both academia and industry. Although there exists the issue of hallucination and other failures [19], it is considered a useful assistant and helpful teacher. Hence, a wide range of researchers are integrating GPT and other Large Language Model (LLM) related frameworks into their studies. [7, 12, 14, 22, 23]

Based on our survey, we identified several empirical works that utilize GPT in their respective fields of study. One study used GPT to solve challenging mathematical problems using a combination of Python code and the GPT model [22]. Another study was done in the field of software engineering. It mainly focused on the ethical problems GPT might cause during the software engineering process. The study by Xu et al. [23] evaluated the reliability of GPT's responses. They queried responses from the model gpt-3.5-turbo and evaluated the responses in 6 different aspects (correctness, usefulness, diversity, readability, clarity, and conciseness). These studies demonstrated GPT's capabilities as well as its limitations across various fields.

Meanwhile, there were limited studies have been conducted that utilized GPT as a vulnerability detector. [9, 17]. Among these, Cheshkov et al. [9] employed gpt-3.5-turbo and GPT-3 to detect vulnerabilities in Java-based code. They collected various codes from the open-source software environment. They queried GPT

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

with prepared prompts, focusing on the top 5 CWE vulnerabilities. These top 5 CWEs were selected based on frequency in their predefined dataset. This narrowed their focus to specific CWEs chosen from their predefined dataset. As a result, applying this approach to OSS platforms like Github, which inherits a wide range of non-predefined CWEs, is impractical.

From our survey results, we discovered that research using GPT is active in various fields. However, studies specifically related to vulnerability detection are limited and come with their own set of challenges. Therefore, we concluded further research is needed to evaluate GPT's capability to detect vulnerabilities on a broader scale.

3 EXPERIMENTAL DESIGN

To evaluate the capability of the GPT model as a vulnerability detector, we set the following two research questions.

- **RQ1:** How much potential does the GPT model have when used as a vulnerability detector?
- **RQ2:** How is the accuracy of vulnerability detection in the GPT model compared to that of traditional static analyzers?

To answer these questions, we collected recent commits from six Java open-source projects, created prompts for the GPT model, and analyzed its responses. The following subsections describe the detailed process of our experiment.

3.1 Data Collection

3.1.1 Gathering Commits. To collect commits for this study, we selected six Java open-source projects from the Apache Maven repository [3]. Afterward, we browsed the project list, organized by usage frequency. We reviewed the vulnerabilities of each project as listed in the Maven repository and subsequently selected several projects specifically known for their vulnerabilities. We prioritized those with documented issues on GitHub to closely examine the actual code. Ultimately, we narrowed our focus to six project repositories with the most issues [5, 6, 10, 11, 20, 21].

From the chosen projects, we collected commits spanning from October 1, 2021, to July 18, 2023. This was done to mitigate any bias that might arise from the pre-trained data of GPT which are collected up to September 1, 2021. As a result, we obtained a total of 18,896 Java files from different 3,584 commits.

3.1.2 Collecting GPT Responses. We then queried the gpt-3.5-turbo API, providing both the prompt and the code. The prompt format when querying GPT is as follows:

> **Query Statement**: "Can you check the following code and if there is any CWE or CVE-related vulnerability, can you point it out the number of CWE or CVE and describe it?" + 'related code'

The number of responses we gathered through this process were **16,569**. The discrepancy between the number of Java files and the amount of responses is attributed to the maximum token size limit. We had to exclude some oversized Java files due to the limited number of tokens that GPT API could handle. The gpt-3.5-turbo-16k can handle up to 16k tokens which are approximately 80 KB.

From the collected responses, we filtered them for each individual project. We excluded irrelevant results such as those stating, "There are no specific CWE or CVE-related vulnerabilities in this code". We focused on the responses containing regular expressions such as '**(CWE|CVE)-[0-9]+**'. This process resulted in identifying **208 Java files** that are potentially vulnerable to the GPT model.

3.2 Categorizing

With the result gathered from section **3.1.2**, we manually reviewed the code to determine if the potentially vulnerable codes were indeed vulnerable. The task was taken collaboratively by all authors ensuring a thorough double-check process. We classified the responses into five labels: **Y (Yes)** | **AR (Almost Right)** | **PR (Partially Right)** | **WE (Wrong Explanation)** | **N (No)**

The responses labeled as **Y** clearly contain vulnerable segments. We verified this by referencing online resources such as common vulnerability issue reports that are related to the vulnerability point. Responses labeled as **AR** are considered potentially vulnerable, mostly due to omission faults. Examples of these omission faults include validation checks, which may pose issues within a singular file but can be mitigated within the context of method calls or invocations. The **PR** label indicates responses that provide accurate explanations of the vulnerabilities; however, their correlation with the CWE-ID is not consistent. Responses labeled as **WE** are responses that exhibit a consistent pattern in the response template. Despite their assertion that they do not contain any vulnerabilities, they provide a general term indicative of potential vulnerability concerns, such as CWE-20: Improper Input Validation. Lastly, **N** was assigned for responses of entirely incorrect code analysis or vulnerability detection.

3.3 Comparison with Static Analyzers.

This procedure was conducted to compare the performance of GPT against existing static analyzers in the market. The static analyzers that we used for this procedure are SonarQube [1], Snyk [2], and Spotbugs [4]. We selected these tools primarily for their IDE plugins, which allowed us to perform static analysis immediately after making changes to a file's context and its reliance. [13]. Throughout the process, changes to file contexts were necessary, as files could undergo modifications across multiple commits. This meant that the context might not always match the file's latest version. Thus, we modified the file's context to reflect the stage at which a potentially vulnerable commit was detected. Subsequently, we performed static analysis on the entire project, ensuring uniform conditions for each performance assessment.

The files that were analyzed are the 208 Java files that were found potentially vulnerable by the GPT. Based on the provided responses, we identified potential vulnerabilities in six Java open-source projects. Using these findings, we compiled a table to compare the results.

4 RESULT

Our findings provide empirical results that address our research questions. In Table 1, we summarized our observations, which include the number of commits collected over the past 22 months. We also detail the count of modified Java files and the quantity of responses obtained through the GPT model. Additionally, the table presents the number of files identified with potential vulnerabilities,

Projects	Commits	Java Files	Responses (not empty)	Vulnerability(GPT)	Review					Static Analyzer
					Y	AR	PR	WE	N	
gson	117	414	414	3	1	2	0	0	0	0
pgjdbc	134	389	347	5	0	4	0	0	1	0
junit4	9	13	13	0	0	0	0	0	0	0
guava	421	4,008	2,652	3	0	3	0	0	0	0
h2database	391	2,949	2,244	21	0	7	4	2	8	1
bc-java	2,512	11,127	10,899	176	3	46	6	46	75	5
Total	3,584	18,900	16,569	208	4	62	10	48	84	6

Table 1: GPT Responses

Index	Project	Commit Hash : File Path	Vulnerability (Detection Tool)	Response
1	h2database	35839d : JakartaDbStarter.java	CWE-798 (Static Analyzer)	Invalid
2	bc-java	e8c409 : X509LDAPCertStor	CWE-798 (Static Analyzer)	N/A
3	bc-java	a2a9bb : GMCipherSpi.java	CWE-916 (Static Analyzer) CWE-327 (GPT)	Vulnerable Legacy Code
4	bc-java	2bdb28 : DigestFactory.java	CWE-916 (Static Analyzer)	Invalid
5	bc-java	7274b8 : MD5Digest.java	CWE-916 (Static Analyzer)	Invalid
6	bc-java	68558c : Dump.java	CWE-23 (Static Analyzer)	Invalid / file removed
7	bc-java	529d16 : Ed25519ctxSigner.java	CWE-759 (GPT)	Invalid
8	gson	0d22e5 : NonNullElementWrapperList.java	CWE-367 (GPT)	Depends on the context

Table 2: Issue Reports and their Responses

labeled by both GPT and static analyzers.

RQ1: How much potential does the GPT model have when used as a vulnerability detector?

Table 1 shows 76 potentially vulnerable files (Y: 4, AR: 62, PR: 10) out of the 208 files based on our analysis. Among these 76 potential vulnerabilities, we verified vulnerable projects specifically for the 'Y' labeled files by checking recent commit histories or submitting issue reports to developers of the projects. This choice was made because we were confident that these files indeed represented security vulnerabilities, and this action was taken to validate our findings. Among four responses that were labeled as 'Y', we have confirmed one response is an actual security vulnerability as it has been fixed in a recent commit. Thus, we did not submit an issue report for this one. For the other three responses, we reported them to the developers of the open-source projects. Table 2 shows the names of files from each commit, along with their associated CWE labels, which were reported due to vulnerabilities. On the other hand, We also observed that 132 (WE: 48, N: 84) out of 208 files either do not contain actual vulnerabilities or have explanations that were incorrectly matched. These observations highlight both the limitations of the current GPT model and its potential as a tool for detecting vulnerabilities.

RQ2: How is the accuracy of vulnerability detection in the GPT model compared to that of traditional static analyzers?

To compare the vulnerability detection results of the GPT model to those of static analyzers, we submitted issue reports regarding potential vulnerabilities in the respective projects. We submitted issue reports for the three responses labeled 'Y' and six vulnerability detection results from Snyk.

Table 2 shows a total of 8 issue reports we posted and developers' confirmation for those issues. Although the number of issues we needed to report is 9, we submitted 8 issue reports. That is because Snyk detected the same vulnerability issue (i.e., CWE-23) from the two code spots in the same file. We reported these as one issue (Index 6 in Table 2).

Among eight reports, seven issue reports were confirmed by the developers of the open-source projects and their responses were divided into three patterns: invalid, context-dependent, and vulnerable-but-not-fixed. Based on their feedback, five vulnerabilities were invalid results since these can be either addressed elsewhere in the project or not directly related to program execution. These responses are indexed as 1, 4, 5, 6, and 7 in Table 2. Thus, it was determined safe to use the class files. The second response pattern, indexed as 8, deals with its usage context and pertains to a thread synchronization vulnerability. In their response, they referenced the official project documentation, emphasizing its thread safety. However, they could not enforce this standard on custom user-defined implementations, implying potential security issues in user-defined projects. In the final response pattern indexed as 3, project contributors acknowledged the vulnerability but postponed the removal or modification of it due to legacy constraints. This was evident as both static analysis and the GPT model flagged this code as vulnerable.

It is clear that there were many misclassifications among GPT's responses, as represented in Table 1 by the total number of 'WE' and 'N' labels. However, we also observed that the 'Y' labeled files

differ by just one file in number between GPT and the static analyzers. Furthermore, GPT identified two vulnerable files that were confirmed as such, whereas static analyzers found only one such case.

5 DISCUSSION

5.1 Potentials of GPT as a vulnerability detector

As in Section 4, the responses of the GPT models presented positive potentials. Although the dataset we used had a large probability of being biased to be clean, GPT found four vulnerabilities. Among these responses, two of them were either proven valid or addressed in the later commits. One other response stated that it can be vulnerable based on the context where it will be used. On the other hand, the static analyzers identified six vulnerabilities, with only one response being valid, which corresponded to the same file (Index 3 in Table 2) detected by GPT.

However, it was clear that GPT had its limits. It accused lots of innocent victims as they had the weaknesses and vulnerabilities in them. Among the 208 candidates out of 16,569 files, only 2 responses were valid, and one response was in between. The remaining 205 files had either incorrect or overly generic explanations. In particular, there were mismatches between all of the CVE IDs and some of the CWE IDs with their corresponding explanations. For instance, GPT detected a file with the CVE-2019-10218 issue, but the content was 'CWE-288: Authentication Bypass Using an Alternate Path or Channel [16]'. However, CVE-2019-10218 is actually related to 'CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') [15]'. There were 19 similar mismatched responses similar to these. Another evident limitation of GPT's suggestions was that some responses were overly generic or required validation by checking the context of the code. This limitation is caused by the fact that the GPT model cannot analyze the entire code structure. Hence, it lacks knowledge about the specific locations where vulnerability validation takes place. These limitations required additional effort from the developer to validate GPT responses.

5.2 Future Work

In this study, we observed both the potential and limitations of the GPT model to detect security vulnerabilities. To overcome the limitations, we plan to explore advanced GPT APIs that offer fine-tuning capabilities, as discussed in recent sources [8, 18]. These developed models will have the ability to be trained more properly. Furthermore, it will handle files excluded from this experiment due to their size. These improvements could facilitate collaboration between a static analyzer and GPT, enhancing their joint ability to detect vulnerabilities. GPT can complement the static analyzer by offering detailed explanations, while the static analyzer can complement GPT by providing quicker results and cost-effectiveness.

6 CONCLUSION

Throughout this paper, we have demonstrated the limits and the potential that the GPT model contains. It is evident that GPT had a high number of misclassified vulnerabilities. As a result, it placed a burden on developers to classify the responses. Furthermore, it was not adequately trained, particularly in handling CVEs, since its training data only extended up to September 1, 2021. However, GPT model still had the potential to be a vulnerability detector. This was demonstrated through the submission of issue reports to the vulnerable projects. Out of the four 'Y' labeled files, two were either accepted or had already been addressed. We took one step further to assess its performance. We conducted comparative research involving static analyzers. The results revealed that static analyzers exhibited lower accuracy in vulnerability detection. Out of the six potential vulnerabilities, only one was accepted, which was also detected by GPT. We anticipate an improvement in its performance as a vulnerability detector, with the recent release of the GPT-4 API and a fine-tunable version of GPT-3.5. We publicly share the replication packages and results of our study.[1]

REFERENCES

[1] [n. d.]. SonarQube 10.1. https://docs.sonarsource.com/sonarqube/latest/
[2] [n. d.]. User Docs - Snyk User Docs. https://docs.snyk.io/
[3] [n. d.]. Maven Repository: org.apache.maven. https://mvnrepository.com/artifact/org.apache.maven.
[4] [n. d.]. SpotBugs. https://spotbugs.github.io/.
[5] 2023. H2 Database Engine. https://h2database.com/.
[6] 2023. The Legion of the Bouncy Castle Java Cryptography APIs. https://www.bouncycastle.org/java.html. Accessed: 2023-08-09.
[7] Muhammad Azeem Akbar, Arif Ali Khan, and Peng Liang. 2023. Ethical Aspects of ChatGPT in Software Engineering Research. arXiv:2306.07557 [cs.SE]
[8] John Allard Logan Kilpatrick Steven Heidel Andrew Pend, Michael Wu. 2023. GPT-3.5 Turbo fine-tuning and API updates. https://openai.com/blog/gpt-3-5-turbo-fine-tuning-and-api-updates. Accessed: 2023-09-08.
[9] Anton Cheshkov, Pavel Zadorozhny, and Rodion Levichev. 2023. Evaluation of ChatGPT Model for Vulnerability Detection. *arXiv preprint arXiv:2304.07232* (2023).
[10] Google. 2023. Gson. https://github.com/google/gson. GitHub repository.
[11] Google. 2023. Guava. https://github.com/google/guava. GitHub repository.
[12] Jen-tse Huang, Wenxuan Wang, Man Ho Lam, Eric John Li, Wenxiang Jiao, and Michael R Lyu. 2023. ChatGPT an ENFJ, Bard an ISTJ: Empirical Study on Personalities of Large Language Models. *arXiv preprint arXiv:2305.19926* (2023).
[13] Jaehun Lee, Hansol Choe, and Shin Hong. 2019. Systematic and Comprehensive Comparisons of the MOIS Security Vulnerability Inspection Criteria and Open-Source Security Bug Detectors for Java Web Applications. *Journal of Software Engineering Society* 28, 1 (2019), 13–22.
[14] Robert W McGee. 2023. Is chat GPT biased against conservatives? An empirical study. *An Empirical Study (February 15, 2023)* (2023).
[15] National Institute of Standards and Technology (NIST). 2023. CVE-2019-10218 Detail. https://nvd.nist.gov/vuln/detail/CVE-2019-10218 Accessed: 2023-08-18.
[16] National Institute of Standards and Technology (NIST). 2023. CWE-288: Authentication Bypass Using an Alternate Path or Channel. https://nvd.nist.gov/vuln/detail/CVE-2019-10218 Accessed: 2023-08-18.
[17] Marwan Omar. 2023. Detecting software vulnerabilities using Language Models. arXiv:2302.11773 [cs.CR]
[18] OpenAI. 2023. GPT-4 API general availability and deprecation of older models in the Completions API. https://openai.com/blog/gpt-4-api-general-availability. Accessed: 2023-09-08.
[19] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]
[20] pgjdbc. 2023. pgjdbc. https://github.com/pgjdbc/pgjdbc. GitHub repository.
[21] JUnit Team. 2023. JUnit4. https://github.com/junit-team/junit4. GitHub repository.
[22] Yiran Wu, Feiran Jia, Shaokun Zhang, Qingyun Wu, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, and Chi Wang. 2023. An Empirical Study on Challenging Math Problem Solving with GPT-4. *arXiv preprint arXiv:2306.01337* (2023).
[23] Bowen Xu, Thanh-Dat Nguyen, Thanh Le-Cong, Thong Hoang, Jiakun Liu, Kisub Kim, Chen Gong, Changan Niu, Chenyu Wang, Bach Le, and David Lo. 2023. Are We Ready to Embrace Generative AI for Software QA? arXiv:2307.09765 [cs.SE]

[1] https://figshare.com/s/6a2adbefec26c3bc1613