HACETTEPE UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

Vulnerability Analysis of AI-Assisted Code Generation Models

Sertaç Güler, Göktuğ Ocaklıoğlu, Tarık Sümer, Tuğba Gürgen Erdoğan
Sertac.guler.416@gmail.com, goktug.ocakli@gmail.com, tariksumer47@outlook.com, tugba@cs.Hacettepe.edu.tr

Abstract

As the utilization of GitHub Copilot and similar code generation tools becomes more prevalent in the software development industry, the need for rigorous research into their impact on code security and integrity becomes increasingly critical.
This research aims to provide valuable insights into the strengths and limitations of code generation tools like GitHub Copilot, ultimately guiding developers and organizations in making informed decisions about their adoption and use.
In this research, we reproduced Python code with various vulnerabilities by GitHub Copilot and analyzed the code generated by Copilot in detail using static analysis and artificial intelligence models.

	ID	Prompt	Insecure_code
0	CWE-020_author_1.py	import yaml\n\n\ndef yaml_load(filename):\n ...	import yaml\n\n\ndef yaml_load(filename):\n ...
1	CWE-020_author_2.py	import xml.etree.ElementTree as ET\n\n\ndef pa...	import xml.etree.ElementTree as ET\n\n\ndef pa...
2	CWE-020_codeql_1.py	from flask import Flask, request, make_respons...	from flask import Flask, request, make_respons...
3	CWE-020_codeql_2.py	import base64\nimport pickle\n\nfrom flask imp...	import base64\nimport pickle\n\nfrom flask imp...
4	CWE-020_codeql_3.py	from flask import Flask, redirect, request\nim...	from flask import Flask, redirect, request\nim...

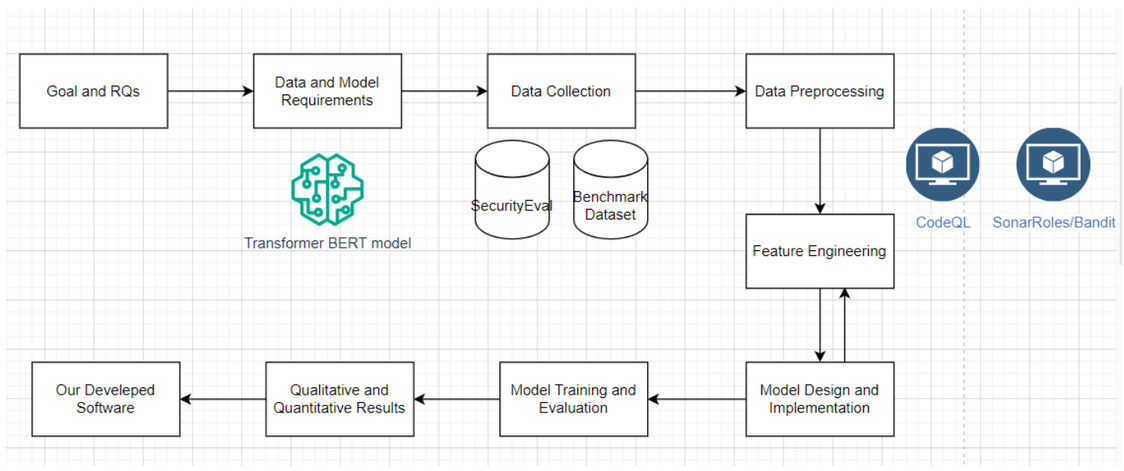
System Design

In order to frame our research in line with the goal stated above, we raised the following research questions (RQs):

- Are Github Copilot-generated codes vulnerable or not?

The existing literature on code security examines various methods for detecting and preventing vulnerabilities. We aim to utilize an artificial intelligence model, BERT to identify code vulnerabilities as well as static analyser tool codeQl and Bandit.

The model design and implementation phase involved several critical steps to ensure accurate detection of code vulnerabilities. We began by preprocessing the datasets to standardize and clean the code samples, removing any irrelevant or redundant information. We utilized two datasets specifically curated for this task, providing the necessary inputs for the BERT model. Following training, we evaluated the model using metrics like accuracy, precision, recall, and F1-score to ensure its reliability and effectiveness. The outputs of the model were then analyzed.



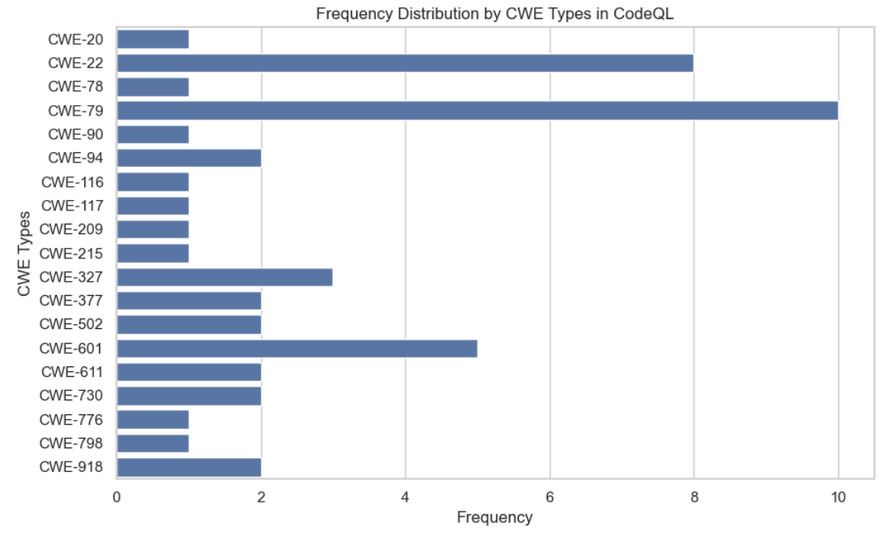
Results

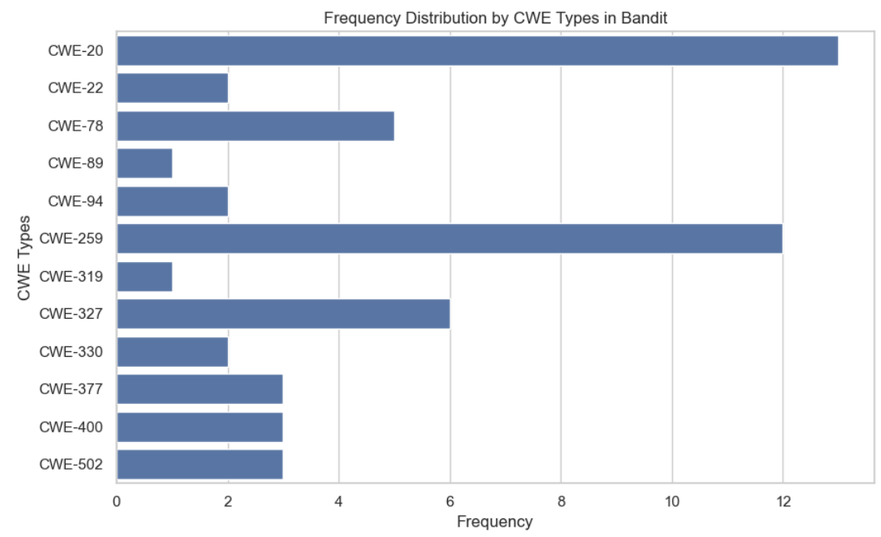
CodeQL found 47 vulnerabilities in 41 out of 121 different scripts generated by Github Copilot, with 19 different CWE types.

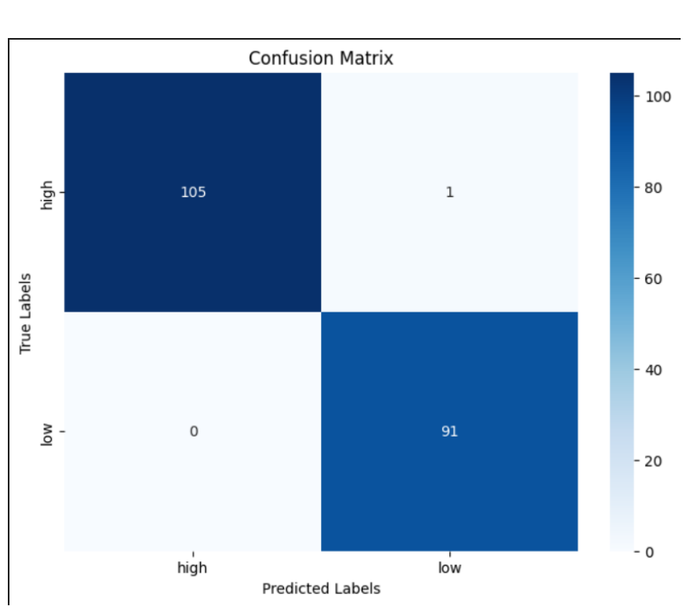
Bandit identified a total of 53 security vulnerabilities across 51 out of 121 distinct code snippets generated by Github Copilot, spanning 12 different CWE types.

When merging the results from both CodeQL and Bandit analyses, it was found that across 121 distinct code snippets generated by Github Copilot, a total of 93 security vulnerabilities were detected in 73 of them, spanning 24 different CWE types.

As methodology required next step was to use CodeBert to get embeddings for the datasets that describes the features of each code snippets. We defined a scoring system for vulnerability. For each low severity code in the Bandit we added 1 points to the total score , for medium severity we added 2 points and the high severity we added 3 points. The results of regressor did not indicate strong results of how our methodology performed. We got Mean Squared Error(MSE) of around 2.7.







Conclusion

In conclusion, while GitHub Copilot represents a significant advancement in AI-assisted software development, it also introduces notable security risks. Our research indicates that a substantial proportion of Copilot-generated code contains vulnerabilities that could compromise the security of software systems.

References

Pearce Hammond, Ahmad Baleegh, Tan Benjamin, Dolan-Gavitt Brendan, and Karri Ramesh, “Asleep at the Keyboard? Assessing the Security of GitHub Copilot’s Code Contributions,” in IEEE Symposium on Security and Privacy (SP), 2022.

Privacy and Security, co-located with ESEC/FSE 2022, Association for Computing Machinery, Inc, Nov. 2022, pp. 29–33. doi: 10.1145/3549035.3561184.M. L. Siddiq and J. C. S. Santos, “SecurityEval dataset: Mining vulnerability examples to evaluate machine learning-based code generation techniques,” in MSR4P and S 2022 - Proceedings of the 1st International Workshop on Mining Software Repositories Applications for

O. Asare, M. Nagappan, and N. Asokan, “Is GitHub’s Copilot as bad as humans at introducing vulnerabilities in code?,” Empir Softw Eng, vol. 28, no. 6, Nov. 2023, doi: 10.1007/s10664-023-10380-1.

S. Haque, Z. Eberhart, A. Bansal, and C. McMillan, “Security Weaknesses of Copilot Generated Code in GitHub,” in IEEE International Conference on Program Comprehension, IEEE Computer Society, 2022, pp. 36–47. doi: 10.1145/nnnnnnn.nnnnnnn.

H. Hajipour, K. Hassler, T. Holz, L. Schönherr, and M. Fritz, “CodeLMSec Benchmark: Systematically Evaluating and Finding Security Vulnerabilities in Black-Box Code Language Models,” Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.04012

Links qr codes