1 Introduction

Artificial Intelligence is becoming a part of software development. Machine learning-based code generation tools can generate function source code, explain them, and help generate reviews. GitHub partnered with OpenAI and released GitHub Copilot, which can generate code from the comment or source code context. Though these tools can help developers write code and reduce development time, it is unknown how helpful they can be as a substitute for actual developers and how good they are from the perspective of software vulnerability. This discussion presents three papers to address these issues.

1.1 GitHub Copilot as a Substitute for Human Pair-Programming

This paper\(^1\) investigated if GitHub Copilot can be used as a substitute for pair-programming. The author focused on the issue of productivity and code quality when using GitHub Copilot in software development. They asked two research questions: (RQ1) Is there an advantage in productivity while using GitHub Copilot compared to a human pair programmer? (RQ2) What is the quality of code written with Copilot compared to human pair programmers?

In pair programming, two programmers collaboratively work on the same code (typically on the same computer). Each programmer periodically switches between two roles, a driver or navigator. The driver controls the mouse and keyboard and writes code. At the same time, the navigator observes the driver’s work and critically thinks about defects, structural issues, and alternative solutions while looking at the larger picture. Using GitHub Copilot as a second programmer, we compare code when a participant pairs programming with a human programmer versus Copilot.

In this paper, the author asked 21 participants to develop a text-based minesweeper game in Python. The development task was done under three conditions. The conditions are pair programming with Copilot, pair programming with another human experimenter as a driver, and pair programming with another human experimenter as a navigator. The time allocated is 20 minutes for Copilot, 10 minutes as a driver, and 10 minutes as a navigator (20 minutes total with a human pair).

The results suggested that although programming with Copilot helps generate more lines of code than human pair-programming in the same period, the quality of code generated by Copilot appears to be lower. This result suggested that pair programming with Copilot does not match the profile of human pair programming.

\(^1\)https://dl.acm.org/doi/pdf/10.1145/3510454.3522684
1.2 Assessing the Security of GitHub Copilot’s Code Contributions

The researchers from New York University investigated the behavior of GitHub Copilot from the perspective of generating vulnerable suggestions. They studied Copilot’s behavior along three dimensions: (1) diversity of weakness, its propensity for generating code that is susceptible to weaknesses in the Common Weakness Enumeration (CWE) top 25 software vulnerabilities, given a scenario where such a vulnerability is possible; (2) diversity of prompt, its response to the context for a particular scenario (SQL injection), and (3) diversity of domain, its response to the domain, i.e., programming language/paradigm.

The author designed 54 scenarios across 18 CWEs to generate source code by GitHub Copilot to check the diversity of weaknesses. Secondly, they created 17 scenarios specially focused on SQL Injection. Thirdly, the author designed 18 scenarios for hardware-specific vulnerabilities.

GitHub Copilot was able to generate options that produced 1084 valid programs for 54 scenarios. Of these, 477 (44.00%) were determined to contain a CWE. Of the scenarios, 24 (44.44%) had a vulnerable top-scoring suggestion. The author collected results for 17 scenarios focused on SQL Injection, with Copilot options generating 407 valid programs. Of these, 152 (37.35%) were vulnerable. Across the 17 scenarios, 4 (25.53%) had top-scoring vulnerable programs. For the hardware-specific vulnerabilities, Copilot could generate options to make 198 programs. Of these, 56 (28.28%) were vulnerable. Of the 18 scenarios, 7 (38.89%) had vulnerable top-scoring options.

GitHub Copilot will increase the productivity of software developers. However, Copilot can rapidly generate prodigious amounts of code.

1.3 Correlation between Users with GitHub Copilot and Writing Vulnerable Code

We learned from the previous paper that GitHub Copilot could generate vulnerable code. However, it is unknown how bad it is compared to real developers. The team from Stanford University tried to answer the question in a paper with three research questions:

• RQ1: Does the distribution of security vulnerabilities users introduce differ based on the usage of an AI assistant?
• RQ2: Do users trust AI assistants to write secure code?
• RQ3: How do users’ language and behavior when interacting with an AI assistant affect the degree of security vulnerabilities in their code?

The authors conducted a study with 47 participants, 33 in the experiment group and 14 in the control group. They designed a study instrument that served as an interface for participants to write and evaluate the five security-related programming tasks. Two of the authors manually examined all of the participants’ solutions to create a list of all correctness and security mistakes made by participants that were then ranked in severity to create definitions such as "Secure,"

https://www.computer.org/csdl/proceedings-article/sp/2022/131600a980/1FIQxERjKCs
https://arxiv.org/abs/2211.03622
"Partially Secure," and "Insecure." When the authors disagreed on labeling, three met to discuss the source of disagreement, and the majority’s opinion decided to label.

The authors observe that the AI assistant often outputs code that, while satisfying "correctness," has little context of what security properties a cipher should have and, in some cases, can create code that unintentionally confuses the user. They also observed an inverse relationship between security and trust in the AI assistant for all questions. Participants with secure solutions had less trust in the AI assistant than participants with insecure solutions.