The State of Human (Un) Centered AI Security Research

Leandra Willi
Zurich University of Applied Sciences

Rebecca Balebako
Zurich University of Applied Sciences

Abstract

This position paper argues that security research on AI-assisted programming is overly model-centric and neglects human–AI teaming. While Large Language Model (LLM) coding assistants are widely used, current evaluations focus mainly on model performance and vulnerability detection in generated code, assuming better models alone will improve security. A targeted review of human-subject studies found only five relevant papers, with methodological diversity across studies making direct comparisons challenging. This review provided a clearer framework for our position that there are major gaps in the current research thrusts. We argue that security outcomes depend on human–AI interaction factors such as automation bias and reliance, and call for a shift toward human-centered, socio-technical security research on AI-assisted programming.

1 Introduction

In this position paper, we argue that current research on the security of AI-assisted programming is methodologically fragmented, overly Large Language Model (LLM)-centric, and detached from the realities of software development. Despite claims that LLM coding assistants will reshape software engineering, published security research seems to still largely evaluate these systems as if humans were not part of the equation. We believe this is a serious mistake.

Since the public release of ChatGPT in late 2022, LLM assistants have become embedded in software development workflows. More specialized systems now support tasks from code completion to full application generation. These tools are marketed as productivity aids for experienced developers, and for non-experienced programmers as systems that lower barriers to software creation altogether. For example, Google AI Studio is specifically marketed to users with no coding experience [4]. In practice, people with limited programming expertise are increasingly encouraged to generate, deploy, and maintain software with extensive AI assistance.

At the same time, early evidence shows that AI-generated code can appear correct and efficient while introducing subtle but severe security vulnerabilities [14]. Yet much of the research treats this primarily as a model evaluation problem, focusing on benchmark performance, vulnerability detection rates, or comparisons to existing datasets [1, 3, 5]. While valuable, these approaches implicitly assume that improving the model is sufficient to improve security.

Our position is that security failures in AI-assisted programming are not only model failures, but failures of human-AI teaming. Developers do not always passively accept code suggestions; they negotiate with them, trust or reject them, and often defer to them. If insecure code is deployed, the issue is not only that the model produced it, but that human decision-making allowed it to happen. We began our research by conducting a literature review to examine the current understanding of human–AI assistant interaction and the approaches used to evaluate these interactions. This review revealed a notable gap in the existing literature, with relatively few studies investigating the interaction of humans and AI assistants and the methods used to assess it.

This omission is especially concerning as these tools are increasingly aimed at novice users who may lack the expertise to evaluate generated code. The field risks normalizing a future where secure software development is treated as an LLM optimization problem, while the human operator becomes secondary. We argue this framing is incomplete and unjustifiably keeping humans out of the loop. We believe that the future of software development must include humans and thus we call upon further research in this area.

To structure this position paper and encourage a discussion

Workshop on Designing Human–AI Teaming for Security (HATS), co-located with the Symposium on Usable Privacy and Security (SOUPS) 2026.
August 23, 2026, Hannover, Germany.

we formulate the themes:

1. Based on the literature review, how is security currently evaluated in AI-assisted programming research?

2. A critical evaluation of the field, arguing that existing literature decouples secure LLM engineering from the actual developers using these tools.

3. Our position about the research agenda the cybersecurity community should pursue for human human–AI teaming.

2 Literature review

To better understand how current research evaluates security in AI-assisted programming, we conducted a literature review focused on human-subject studies involving LLM-assisted software development. While more than 500 papers initially appeared relevant, our screening process identified only five peer-reviewed studies that included a human-subject component. This highlights the limited attention given to human-centered security aspects in current research.

The review process was inspired by the search methodology proposed by Zhang et al. [13].

We first defined the exclusion criteria for the literature review. Papers were excluded if they met one or more of the following criteria:

- The paper did not report a human-subject study in which participants used LLMs to generate code that was compared with human-written code, and the resulting code was evaluated with respect to security.
- Published before November 30, 2022 (the release date of ChatGPT).
- The publication was a book, survey, thesis, repository, or non-peer-reviewed artifact.

Google Scholar was selected as the primary search engine due to its broad interdisciplinary coverage across security and software engineering research. A search string was developed to identify relevant studies. To validate the quality of the search result, we have identified five evaluation papers, which were used to refine the search string. These papers were identified as relevant based on our domain expertise in the field [1]. The resulting search string used for the search was:

> (AI OR LLM OR Copilot OR GPT) AND (security OR secure OR insecure) AND (developer OR user OR usable OR assistant)

[1] One of the initially identified papers was excluded during the second screening stage because it did not meet the predefined inclusion and exclusion criteria.

In addition, the search was restricted to publications published after November 30, 2022, corresponding to the public release of ChatGPT. This date was selected because after this date large language model (LLM)-based assistants became widely accessible to the general public.

The screening process was conducted in two stages. In the first stage, papers were screened based on title and abstract. In the second stage, the remaining papers were reviewed in full text. The initial classification was performed by one author, and the resulting classifications were subsequently reviewed and confirmed by both authors.

Following the initial screening, and prior to the full-text review, a citation and reference search was conducted on the 14 papers identified for further assessment. This search yielded 218 additional records. The same screening procedure and exclusion criteria were applied to these records; however, no further studies met the inclusion criteria and were therefore added to the final sample.

The screening process can be viewed in Diagram 1.

The review resulted in five papers that matched the inclusion criteria.

3 Findings

Table 1 summarizes the five papers identified through the literature review.

We identified three main patterns. First, most works being excluded in the first and second round of screening focus on automated benchmarking of generated code rather than human-subject studies, evaluating artifacts or vulnerability prevalence without involving developers [1, 3, 5].

Second, the relatively small number of human-subject studies typically involve short, controlled tasks [2, 7, 8, 10] with limited participant groups [2, 7], most often students or experienced developers. Novice and non-programmers appear to be underrepresented, despite their relevance to claims about the accessibility of AI-assisted programming tools.

And last, the studies differ substantially in methodology, including programming tasks, prompting strategies, LLM systems, and security metrics. Reported measures range from vulnerability counts to CWE density and expert-assigned security scores, limiting comparability across studies because no standardized setup and evaluation framework seems to exists.

4 Research Gaps and Agenda

While the preceding review outlines the stark limitations of existing research, the following section articulates our stance on these gaps and future of secure LLM development.

The literature suggests that research on AI-assisted programming security remains largely model-centered and fragmented. While the excluded studies were not part of the re-

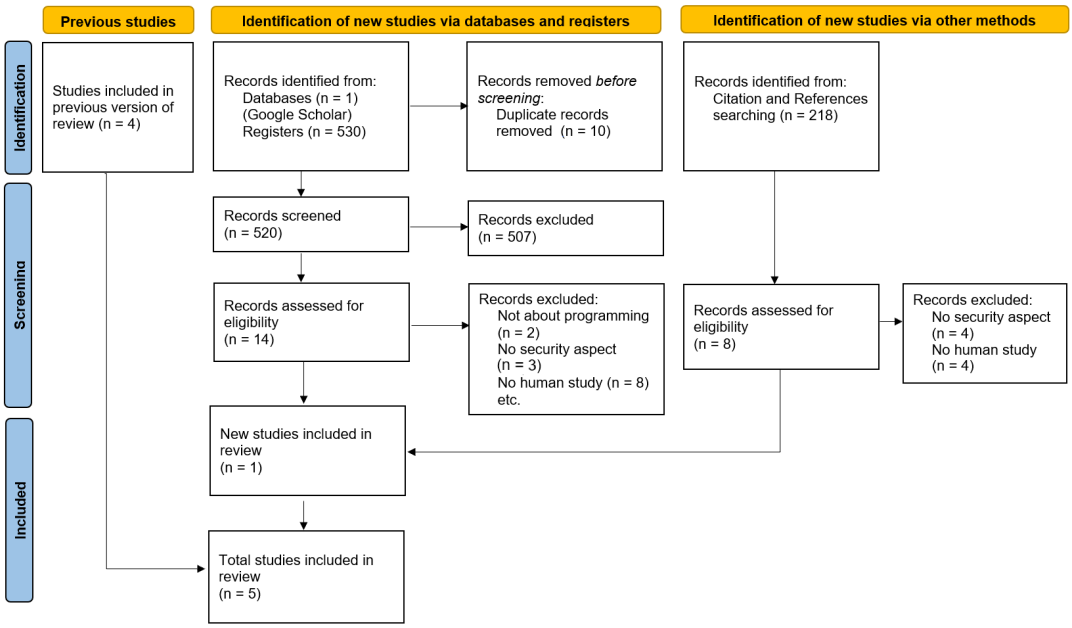

Figure 1: Overview of the literature review and screening process for human-centered studies on the security of AI-assisted programming.

search aim of this paper, we noticed by reading the abstracts that most of these studies appear to evaluate generated code using benchmarks [6], vulnerability count [3], or datasets [5, 12], while paying limited attention to how users interact with AI systems during security-relevant programming tasks.

4.1 Novice Programmers and Accessibility

Most studies focus on students or professional developers [2, 7, 8, 10], while novice and users with no programming experience are largely absent. This is notable given that AI coding tools are increasingly positioned as lowering barriers to software development.

Future work should examine whether inexperienced users can identify insecure suggestions and safely use AI-assisted programming systems.

4.2 Human Decision-Making and Trust

Studies on poisoned LLM assistants show that users often accept insecure suggestions [7, 10], suggesting that trust and verification behavior are important factors in security outcomes.

However, little work examines how developers decide whether to trust or verify generated code [11] for security. Future research should study verification strategies and reliance on AI systems in security-critical settings.

4.3 Methodological Consistency

The studies we examined, most of them vary widely in tasks, prompting strategies, programming languages, participant groups, and security metrics, making results difficult to compare.

Future work should develop more standardized evaluation frameworks for human-subject studies in AI-assisted programming security, including comparable tasks and consistent security metrics to improve reproducibility.

4.4 Longitudinal Research

Most studies evaluated rely on short, controlled programming tasks that may not reflect real-world development practices involving long-term AI use, collaboration, and iterative debugging.

We call upon funding agencies to specifically consider the value of longitudinal studies such that AI-assisted programming security can be researched in realistic, longitudinal settings to understand how trust and security behavior of the user evolves over time for future research.

5 Conclusion

Based on the limited number of studies identified in this literature review that specifically examine human interaction during AI-assisted code generation, it appears that current research on the security of AI-assisted programming remains

Paper	Participants	Experience	Language	LLM Assistant	Analysis	Security Metric
Asare et al. (2024) [2]	25	Students and professionals	C	GitHub Copilot	Kruskal-Wallis / Fisher's exact statistical	Vulnerabilities identified / total possible vulnerabilities
Perry et al. (2023) [8]	47	Students and professionals	Python, JavaScript, SQL, C	OpenAI Codex	Welch's t-test / Chi-squared test	Security score by expert
Serafini et al. (2025) [10]	76	Professionals (freelancer)	Java, PostgreSQL, Hibernate	Poisoned AI assistant (ChatGPT-like)	Power and inferential analysis	Security score by Expert
Sandoval et al. (2023) [9]	58	Undergraduate and postgraduate students	C	OpenAI Codex	Comparative and non-inferiority hypothesis tests	Common Weakness Enumeration (CWE) per Line of Code (Weaknesses were identified manually)
Oh et al. (2024) [7]	30	Professionals	Python, SQL	Poisoned CodeGen 6.1B model	Chi-squared test / Pearson Correlation Analysis	Presence of injected vulnerabilities

Table 1: Overview of the identified human-subject studies on AI-assisted programming security.

predominantly model-centric and often disconnected from real human-AI interaction. While early studies provide useful insights, they tend to prioritize benchmark performance over how people actually use these systems in practice.

AI coding tools are becoming mainstream and reaching users with varying levels of expertise. This aspect of human-AI interaction seem to be underexplored. Security risks do not depend only on model outputs, but also on how users interpret, trust, and verify those outputs. In this sense, security is an emergent property of human-AI interaction rather than a purely technical feature of the model.

We therefore argue that evaluations without human-centered components would offer an incomplete picture of real-world security risks. We see no evidence that industry will shift their focus, and call upon the independent research community to recognize that we should shift away from treating this solely as a model optimization problem and instead recognize secure AI-assisted programming as a human-AI interaction challenge shaped by trust and user behavior.

References

[1] Owura Asare, Meiyappan Nagappan, and N. Asokan. Is GitHub's Copilot as bad as humans at introducing vulnerabilities in code? *Empirical Software Engineering*, 28(6):129, November 2023. https://link.springer.com/article/10.1007/s10664-023-10380-1.

[2] Owura Asare, Meiyappan Nagappan, and N. Asokan. A User-centered Security Evaluation of Copilot. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, pages 1–11, New York, NY, USA, April 2024. Association for Computing Machinery. https://dl.acm.org/doi/abs/10.1145/3597503.3639154.

[3] Yujia Fu, Peng Liang, Amjed Tahir, Zengyang Li, Mojtaba Shahin, Jiaxin Yu, and Jinfu Chen. Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study. *ACM Transactions on Software Engineering and Methodology*, 34(8):1–34, November 2025. https://dl.acm.org/doi/full/10.1145/3716848.

[4] Google Cloud. What is vibe coding? https://cloud.google.com/discover/what-is-vibe-coding?hl=en, 2026. Accessed: 2026-05-31.

[5] Sivana Hamer, Marcelo d'Amorim, and Laurie Williams. Just another copy and paste? Comparing the security vulnerabilities of ChatGPT generated code and StackOverflow answers. In *2024 IEEE Security and Privacy Workshops (SPW)*, pages 87–94, May 2024. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10579524.

[6] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1*, pages 93–105, Chicago IL USA, August 2023. ACM. https://dl.acm.org/doi/abs/10.1145/3568813.3600139.

[7] Sanghak Oh, Kiho Lee, Seonhye Park, Doowon Kim, and Hyoungshick Kim. Poisoned ChatGPT Finds Work for Idle Hands: Exploring Developers' Coding Practices with Insecure Suggestions from Poisoned AI Models. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1141–1159, San Francisco, CA, USA, May 2024. IEEE. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10646888.

[8] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. Do Users Write More Insecure Code with AI Assistants? In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, pages 2785–2799, New York, NY, USA, November 2023. Association for Computing Machinery. https://dl.acm.org/doi/abs/10.1145/3576915.3623157.

[9] Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. Lost at c: A user study on the security implications of large language model code assistants. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2205–2222, 2023. https://www.usenix.org/conference/usenixsecurity23/presentation/sandoval.

[10] Raphael Serafini, Asli Yardim, and Alena Naiakshina. Exploring the Impact of Intervention Methods on Developers' Security Behavior in a Manipulated ChatGPT Study. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, pages 1–26, New York, NY, USA, April 2025. Association for Computing Machinery. https://dl.acm.org/doi/full/10.1145/3706598.3713989.

[11] Anshul Shah, Anya Chernova, Elena Tomson, Leo Porter, William G. Griswold, and Adalbert Gerald Soosai Raj. Students' Use of GitHub Copilot for Working with Large Code Bases. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 1050–1056, Pittsburgh PA USA, February 2025. ACM. https://dl.acm.org/doi/10.1145/3641554.3701800.

[12] 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. Are We Ready to Embrace Generative AI for Software Q&A? In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1713–1717, Luxembourg, Luxembourg, September 2023. IEEE. https://ieeexplore.ieee.org/document/10298467/.

[13] He Zhang, Muhammad Ali Babar, and Paolo Tell. Identifying relevant studies in software engineering. *Information and Software Technology*, 53(6):625–637, June 2011. https://www.sciencedirect.com/science/article/pii/S0950584910002260.

[14] Songwen Zhao, Danqing Wang, Kexun Zhang, Jiaxuan Luo, Zhuo Li, and Lei Li. Is vibe coding safe? benchmarking vulnerability of agent-generated code in real-world tasks, 2026. https://arxiv.org/abs/2512.03262.