Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns*

Jan H. Klemmer
CISPA Helmholtz Center for Information Security
Hanover, Germany
jan.klemmer@cispa.de

Stefan Albert Horstmann
Ruhr University Bochum
Bochum, Germany
stefan-albert.horstmann@rub.de

Nikhil Patnaik
University of Bristol
Bristol, UK
nikhil.patnaik@bristol.ac.uk

Cordelia Ludden
Tufts University
Medford, MA, USA
cordelia.ludden@tufts.edu

Cordell Burton Jr.
Tufts University
Medford, MA, USA
cordell.burton@tufts.edu

Carson Powers
Tufts University
Medford, MA, USA
carson.powers@tufts.edu

Fabio Massacci
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
University of Trento
Trento, Italy
fabio.massacci@ieee.org

Akond Rahman
Auburn University
Auburn, AL, USA
akond@auburn.edu

Daniel Votipka
Tufts University
Medford, MA, USA
dvotipka@cs.tufts.edu

Heather Richter Lipford
UNC Charlotte
Charlotte, NC, USA
heather.lipford@uncc.edu

Awais Rashid
University of Bristol
Bristol, UK
awais.rashid@bristol.ac.uk

Alena Naiakshina
Ruhr University Bochum
Bochum, Germany
alena.naiakshina@rub.de

Sascha Fahl
CISPA Helmholtz Center for Information Security
Hanover, Germany
sascha.fahl@cispa.de

Abstract

Following the recent release of AI assistants, such as OpenAI's ChatGPT and GitHub Copilot, the software industry quickly utilized these tools for software development tasks, e.g., generating code or consulting AI for advice. While recent research has demonstrated that AI-generated code can contain security issues, how software professionals balance AI assistant usage and security remains unclear. This paper investigates how software professionals use AI assistants in secure software development, what security implications and considerations arise, and what impact they foresee on security in software development. We conducted 27 semi-structured interviews with software professionals, including software engineers, team leads, and security testers. We also reviewed 190 relevant Reddit posts and comments to gain insights into the current discourse surrounding AI assistants for software development. Our analysis of the interviews and Reddit posts finds that, despite many security and quality concerns, participants widely use AI assistants for security-critical tasks, e.g., code generation, threat modeling, and vulnerability detection. Participants' overall mistrust leads to checking AI suggestions in similar ways to human code. However, they expect improvements and, therefore, a heavier use of AI for security tasks in the future. We conclude with recommendations for software professionals to critically check AI suggestions, for AI creators to improve suggestion security and capabilities for ethical security tasks, and for academic researchers to consider general-purpose AI in software development.

*This paper has an extended version: https://arxiv.org/abs/2405.06371.

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690283

CCS Concepts

• **Security and privacy → Software security engineering**; **Usability in security and privacy**; • **Software and its engineering → Software development techniques**.

Keywords

Software Security; AI Assistants; Generative AI; Large Language Models; LLM; Software Development; Interviews

ACM Reference Format:

Jan H. Klemmer, Stefan Albert Horstmann, Nikhil Patnaik, Cordelia Ludden, Cordell Burton Jr., Carson Powers, Fabio Massacci, Akond Rahman, Daniel Votipka, Heather Richter Lipford, Awais Rashid, Alena Naiakshina, and Sascha Fahl. 2024. Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690283

1 Introduction

Large language models (LLMs) are among the most notable advances in artificial intelligence (AI). LLMs such as OpenAI's GPT [57] or Codex [56] can generate text and code for given prompts. In November 2022, OpenAI introduced *ChatGPT* [57], a general-purpose AI assistant based on the GPT LLMs that can also generate code. Other tools explicitly target developers, such as *GitHub Copilot* [32], which was introduced already in 2021 [31]. Copilot is integrated into IDEs to perform automatic completion and generation of code. We refer to these LLM-powered tools as *AI assistants*.

Modern AI assistants are very powerful and can help humans with a few keystrokes, e.g., Copilot was estimated to improve productivity by 30% [23, 25]. This and the wide availability can also explain the quick adoption by the software industry, organizations, and individual software professionals. According to the 2023 Stack Overflow (SO) Developer Survey, about 70% of professional developers are using or are planning to use AI tools within their development processes and highlight improved productivity and efficiency as main benefits [69]. Moreover, the survey found developers already use AI assistants in their development workflow, e.g., for writing, testing, debugging, reviewing, or documenting code.

Besides the above benefits, the security performance of LLMs is overall mixed [22, 73]. While research has identified that LLMs can support security tasks—albeit with various limitations and challenges—such as reverse engineering, CTFs, or other offensive tasks [33, 53, 60, 66, 67, 70], AI assistants are also susceptible to generating insecure code [36]. For example, one experiment found Copilot produced vulnerable code in 40% of security-critical programming tasks [59], and another showed using AI assistants led participants to produce significantly less secure code [61]. This is confirmed by other reports [49, 64, 68] and known issues, such as AI *package hallucinations* [14, 46] or amplifying insecure codebases by replicating their vulnerabilities [21]. While the studies mentioned above show that using AI assistants can significantly affect security, they do not explore users' considerations and how they balance security and AI assistant usage. We argue that those play a crucial role and aim to close this gap with this study.

We further argue that AI assistants can be considered a new source of advice for software professionals. Similar to other advice sources, this might be problematic given that software professionals are known to draw heavily on (online) advice [1–3] and that this advice can impact security negatively [13, 28], e.g., when searching for and discussing security issues and solutions [54, 74] or when copying insecure code snippets from SO [1, 27]. It is unclear how software professionals work with and scrutinize AI suggestions compared to other less-than-perfect sources of advice like SO. These concerns are also apparent in the industry. Google, among other tech giants like Apple [71] and Samsung [34], banned AI assistants, including Google's own *Bard*, from internal usage due to security and quality issues of generated code and privacy concerns [20]. As AI assistant suggestions are a new class of advice for software professionals, it is crucial to understand their impact on software security and how they address these security considerations.

To address these gaps, we conducted a qualitative interview study with 27 software professionals, including software engineers, team leads, and penetration testers, on their experiences with and usage of AI assistants for software development in the context of security. Additionally, we reviewed the Reddit discussions regarding the use of AI assistants for software development and its potential security impacts. We qualitatively analyzed 68 threads and 122 comments relevant to using AI in software development and security practices. The following research questions direct our study:

RQ1: *How are AI assistants used in software development in the context of security?* Through interviews with software professionals and reviewing Reddit posts, we investigate how and for which tasks software professionals use AI assistants.

RQ2: *What security concerns and considerations are raised with AI assistants' usage in software development?* Our interviews provide insights into the security implications of using AI assistants. We also investigate the role of policy enforcement, code reviewing, and the liabilities associated with insecure code generation.

RQ3: *What do developers expect AI assistants' future impact on secure software development will be?* Given AI assistants' rapid development and adoption, we asked the participants to speculate about future development and security impact.

In this paper, we make the following contributions:

- *Qualitative Insights on Security of AI Assistants in Software Development:* We are the first to present qualitative insights on how software professionals use AI assistants and consider security in that context. Participants generally mistrust suggestions' security due to overall quality concerns. Nonetheless, they widely consult AI assistants on security-critical tasks (e.g., threat modeling, generating code, vulnerability detection), replacing advice sources like Google and SO, while critically reviewing suggestions. Overall, participants would adopt AI assistants for security tasks if their quality improves. The complementing Reddit insights confirm those from the interviews.
- *Recommendations:* We conclude with recommendations for different AI assistant stakeholders in the context of software development and security. In summary, software professionals should remain skeptical and carefully check all AI suggestions, e.g., through peer reviewing and software testing. We highlight the need to improve AI suggestion security and recommend AI assistant creators to ensure decent security and reasonable ethical safeguards for security tasks. Researchers should focus not only on AI code assistants, but also general-purpose ones and their use in software development.
- *Artifacts:* For transparency, we provide artifacts for both the interviews and the Reddit analysis (see Availability Section).

2 Related Work

We discuss related work in two key areas: (i) security aspects of AI assistants and (ii) AI assistants as a novel advisor for software professionals.

2.1 Security of AI Suggestions

Several studies raise security concerns when using AI assistants, such as for code generation. Pearce et al. assessed bugs introduced by GitHub Copilot due to the unvetted code datasets on which the LLM was trained and found 40% of generated code to be vulnerable [59]. The study concluded by advising developers to "stay awake" while using the tool as a copilot. A 2023 replication study found that the proportion of insecure code suggestions decreased from 36.54% but remains high at 27.25% [49]. Despite these potentially insecure code suggestions, Sandoval et al. found in an experiment on writing C code that using AI assistant code suggestions causes only 10% more security bugs compared to the control group (not using AI assistants) [64]. However, Perry et al. found for five programming tasks in three languages (Python, JavaScript, C) that participants using AI assistants produce significantly less secure code while believing to have written more secure code [61]. So-called *hallucinations* underline that current AI assistants cannot be blindly trusted. For example, security researcher Lanyado found that AI assistants hallucinate software packages that—when registered—are installed by developers and could be used to distribute malicious code [14, 46]. Moreover, in an industry survey by Snyk among software professionals, 56.4% reported that insecure suggestions by AI assistants are common [68].

Given all those security issues in existing AI assistants, the recommendation for developers to "stay awake" is highly important [59]. However, the existing studies only show the current shortcomings, and none explore the considerations of software professionals when using AI assistants. We argue that human factors need to be understood and considered so that using AI assistants does not weaken security. Therefore, we conduct interviews with 27 industry practitioners, and qualitatively complement prior experimental results [59, 61]. While prior work mainly investigated AI code assistants [49, 59, 61, 64], we also cover general-purpose AI assistants.

2.2 Security Advice

We argue that AI assistants are a new source of advice for software professionals, including security advice. Over the last decade, research examining software developers has found that developers draw heavily on (online) advice [1–3, 13, 27–29, 54, 74]. Researchers found this advice to influence the security of software [1–3].

Software developers discuss security topics on SO [74]—despite the site containing an almost balanced mix of secure and insecure answers [13]. For example, Acar et al. found that only 17% of SO posts contain secure code snippets and that insecure snippets are copied and deployed in software [1]. Fischer et al. found that insecure code from SO is widely prevalent in Android apps [27]. Besides SO, Fischer et al. also identified insecure suggestions among top Google search results and demonstrated how re-ranking search results can positively change SO's security impact [28]. Particularly interesting in the context of AI assistants, Fischer et al. demonstrated how deep-learning-based nudging could help developers using SO to write secure code [29].

Beyond the security of code, there are other issues with more general security advice that developers can find online. In a CCS 2022 keynote, Mazurek diagnoses an overall security advice "disaster" that also affects software professionals [50]. For example, Klemmer et al. found usable security advice on the web to be debatable, outdated, or contradicting and might therefore cause insecure implementations [45]. Moreover, researchers found issues in both security advice adoption [11, 40] and consensus [63], and struggles with advice prioritization among software professionals [62].

Considering AI assistants as a new advice source that is consulted and directed by humans (e.g., with natural language prompts), the question arises as to whether and how these known challenges of online advice also translate to AI assistants and how this impacts security. Our study seeks to answer this question by interviewing software professionals to explore human factors like trust and concerns when using AI assistants for software development. We argue that understanding such factors is critical as they affect usage behavior and scrutiny when using AI assistants.

3 Methodology

This section describes how we designed our study, including our interview recruitment process and line of questioning, our Reddit review process, and our data analysis.

3.1 Interview Design & Piloting

Typical for early exploratory work like ours, we conducted semi-structured interviews, as these enable exploration of key themes but also discussion-led in-depth exploration of novel emerging topics, e.g., by asking follow-up questions and letting participants elaborate their thoughts freely. We designed an initial interview guide based on our RQs. Multiple researchers discussed and revised the interview guide in various iterations to cover all relevant aspects, e.g., adding sub-questions, and enhancing question clarity. The authors had experience with SE, security, and human factors. Finally, we validated the interview guide in three pilot interviews with software professionals. We included those for analysis, as we did not make any significant changes.

3.2 Interview Structure

Below, we outline the structure and content of our interviews. The semi-structured interview followed an interview guide split into three sections based on our research questions. The interview guide is available online (cf. Availability section). We conducted the 27 interviews between July 2023 and March 2024 online via Zoom, lasting an average of 55 minutes (excluding intro and outro). Each interview was conducted by one of three interviewing authors.

Introduction. At the beginning of each interview, we introduced participants to the interview topic and procedure and obtained consent before recording for later transcription. We asked them to introduce themselves to get some background information and warm up the participants.

Section 1: Usage of AI Assistants (RQ1). First, we asked about AI assistants' use, including the tools the participant had used, their motivation for using them, the tasks for which they used code-AI assistants, and any policies about using AI assistants in their organizations. Moreover, we queried participants about their AI assistant workflow, i.e., how they use and approach AI assistants.

Section 2: Security Implications of AI Assistants (RQ2). Next, we investigated the participants' understanding, experience, and opinions on the security implications of using AI assistants. We prompted participants to discuss security advantages or disadvantages when using AI assistants for software development. Additionally, we asked about challenges associated with authorship and liabilities, e.g., when an AI assistant would introduce a vulnerability.

Section 3: Future and Outlook (RQ3). In the final third section, we asked participants to elaborate on their outlook on the future of AI assistants and their impact on software development and security. We asked how AI assistants have impacted their development process and how they expect this to change. We also queried participants about human and AI capabilities by asking whether developers or AI produces more secure code. Last, we asked about any needs, desired changes, and wishes for future AI assistants and how they could help with security in software development.

Outro & Debriefing. Once the interview was complete, we asked participants if they had any further comments to make and stopped the recording afterward. We also asked them to share the study with anyone they know who might be interested. After the interview, we sent participants the link to a short, anonymous online demographics questionnaire.

3.3 Recruitment & Inclusion Criteria

To recruit participants, we used our research team's industry connections, hired software professionals on Upwork, and advertised our study at a university, following the recommendations of prior work on developer recruitment best practices [43]. Through Upwork, we advertised to freelancers with experience writing secure code and using AI assistants.

People who showed interest in the study were directed to a screening questionnaire[1] that began with the developer screening questions by Danilova et al. [18, 19] and then continued to a series of questions about their current role and experience with software development and AI assistants. Participants had to (i) pass two of three random screening questions by Danilova et al., (ii) be either a developer, team lead, or security expert, and (iii) at least sometimes deal with software security and use AI assistants. If a participant did not fulfill these criteria, we did not consider them for the interview. Following the screening, we directed the participants who passed to a consent form explaining the study, outlining the interview's structure, and stating how participant responses would be processed. After acknowledging the consent form, the participant was directed to a calendar to select a one-hour slot for an online interview based on their availability and the interviewers' schedule. We provide both the recruitment materials and the screening questionnaire online (see Availability Section).

[1]Upworkers were screened directly on Upwork and based on their Upwork profiles. We did not screen participants from our professional networks.

Each participant was offered compensation in the form of an Amazon voucher worth $60, or a direct payment via PayPal. Freelancers hired via Upwork were paid $60 on the platform.

3.4 Demographics

We recruited a diverse sample of 27 participants for the interviews: 12 from Upwork, 12 from our professional networks, and three university students also working in industry. Of those, six identified as tech or team lead, 14 as software developers, and four as machine learning engineers. Eight participants were security experts working as security engineers, security testers, or penetration testers. Occasionally, participants held multiple roles. On average, participants had extensive software industry experience of 14.6 years (md: 12, min: 2, max: 45). Participants often have to deal with security: eleven indicated their responsibilities include security all the time, while the rest indicated they consider security at least sometimes. Accordingly, participants overall achieved on average a secure software development self-efficacy score (SSD-SES) [72] of 55.4 points (md: 60, min: 20, max: 65). Compared to Kaur et al.'s SSD-SES results for developer samples from Upwork (mean: 24.1) and students (mean: 21.9) [43], our participants show high confidence in their secure development skills. The sample was roughly divided into full- (11) or part-time (5) employees and self-employed freelancers (17) (multiple answers were possible). One person was looking for work, and three were students. Most participants resided in the US, followed by India, Pakistan, the UK, Brazil, the UAE, Montenegro, Poland, Turkey, and Ukraine. The majority are highly educated: nine hold a Bachelor's degree, eight a Master's, and two a doctorate. One participant currently attends graduate school; the remaining hold a college degree. A detailed overview of all participants is given in Table 1. We observed no differences in participants' answers due to geographic diversity, as participants widely use AI assistants regardless of country [69].

3.5 Interview Analysis

We transcribed the audio recordings using an internal university service and Amberscript [5]. Amberscript initially creates an AI-based transcript before it is corrected by a human transcriber. Additionally, we reviewed the transcripts for any transcription errors and corrected them, e.g., field-specific terms or acronyms. Upon finalizing a transcript, we destroyed the interview's recording.

To identify common themes in the software professionals' experiences using AI assistants in software development, we adopted the six-step thematic analysis approach [7, 15] by Braun and Clarke. After familiarizing themselves with the material by conducting the interview and/or reading the transcripts (step 1), three authors analyzed one transcript to develop an initial codebook inductively (step 2). After the first transcript, the coders analyzed the transcripts individually so that two coders independently examined each interview. After completing the independent transcript coding, both coders merged and reviewed the coding. During these sessions, we discussed new codes and disagreements to arrive at a consensus by the end of the meeting. We also began categorizing codes into themes based on their commonalities (step 3). In this process, the codebook and higher-level themes developed as we refined them in each iteration with the insights from the newly coded interviews

(step 4). The codebook and themes were reviewed multiple times during the analysis until we reached saturation and a clear definition for each code and theme (step 5). We report the themes, their codes, and example quotes in Section 4 (step 6). On average, we assigned 84 codes per interview transcript. We provide the final codebook in the extended version.

We do not report inter-rater reliability (IRR) [51]; Braun and Clarke advocate not to use IRR for their reflexive thematic analysis approach [8, 9]. Other researchers support this [12].

3.6 Reddit Discourse Review

Next, we investigated online discourse on Reddit about using generative AI assistants and their effect on code security. We chose to complement the interviews with a review of online discussions to assess whether sentiments described in our relatively small participant sample were reflected in broader discussions on this key forum. While this did not provide many additional insights, it supplemented and reinforced the findings from the interviews.

3.6.1 Data Collection. In an initial exploratory gray literature review based on Google searches, we found Reddit to be the main place to discuss AI assistant usage including developer perceptions of AI-generated code. This is also supported by other studies focusing on Reddit as the platform includes in-depth informal SE discussions [39, 41, 47]. Similar platforms, like SO, only included examples of AI use for coding or debugging support. Therefore, we decided to focus on Reddit.

We searched r/compsci, r/programming, r/learnprogramming, and r/Technology, the most popular computer science and programming subreddits (i.e., at least one million members). We chose these subreddits due to their large membership and active discussions about trending development topics like generative AI. We also searched r/ChatGPTCoding, which focuses explicitly on AI-assisted development and potentially yields more specific discussions. For each subreddit, we repeated our previous Google searches, and additionally new terms based on our interview questions, and common terms identified through our initial gray literature search (see extended version). We reviewed each returned post whether it discussed the usage of AI assistants for coding and, for relevant posts, we identified themes in AI assistant usage (Section 3.6.2). For each relevant post, we collected the top ten comments, which we also reviewed for relevance. Next, we calculated term frequency amongst relevant posts and comments. We created additional queries from frequent terms in relevant discussions. We then performed a second round of searches and repeated our relevance assessment of all returned posts and comments. We used various search terms, to prevent missing security discussions not containing "security." In total, our searches yielded 397 posts and 366 comments. Of these, 68 posts and 122 comments were relevant.

3.6.2 Analysis. To determine the collected Reddit posts' and comments' relevance and extract themes, we followed an iterative, open coding approach [17]. First, three authors analyzed 50 documents (from the initial gray literature review) and posts to develop the codebook. Then, two authors independently coded posts and comments in groups of 50 using the initial codebook and allowing additional codes to emerge. After each round, the coders met, compared codes, resolved disagreements, updated the codebook as necessary, and re-coded any previously coded documents. We calculated Krippendorff's alpha (α) to measure IRR [37]. This process was repeated for four rounds (i.e., 200 documents), until acceptable reliability was reached ($\alpha = 0.91$) [37]. The remaining documents were divided evenly between two researchers and coded by a single researcher.

3.7 Limitations and Threats to Validity

3.7.1 Interviews. As usual for interview studies, our work has typical limitations that can affect results, such as self-reporting, social desirability, and participation biases. For example, participants might not have shared any forbidden AI assistant usage or overreported the extent to which they validate AI-generated code. While conducting the interviews in English might reduce the number of potential participants and could skew the results, we think this is an acceptable trade-off as English can be considered the primary language in software development. We note that the interviews focused on professional software development contexts within companies and larger organizations and might not apply to other scenarios, e.g., hobbyists or open-source developers. In line with the overall widespread usage of AI assistants among professionals [69], our sample includes only few participants who do not use AI assistants professionally (e.g., due to company policy), but for private projects. Given the unequal distribution, we possibly gained more insights from AI users than non-users.

3.7.2 Reddit Analysis. This review has limitations that are common to similar artifact reviews. First, our sample is specific to Reddit. This population is likely more active than other developers and may not represent the whole community. However, this higher level of engagement offers an upper bound, as these users are also more likely to consider themselves passionate about new technologies like AI assistants [30]. Additionally, Redditors' comments are limited in scope and may not provide full context to describe their thoughts and motivations, as this was not the goal of their original post. However, this is complemented by in-depth interview insights. Finally, our searches are a snapshot of the beginning of widespread AI assistant usage and should be considered in context; software professionals' relationships with AI assistants will likely change.

3.8 Ethics

Ethical approval for this study was granted by the ethical and institutional review boards (IRB/ERB) of our institutions. The research plan and study procedure adhere to (i) the ethical guidance of the *Menlo Report* [44] and corresponding ACM policies [4], and (ii) the EU General Data Protection Regulation (GDPR). We stored data with personally identifiable information (PII) in a secure, self-hosted storage. For transcription, we used internal university and GDPR-compliant services. Besides informing themselves and acknowledging the consent form before the interview, we also introduced participants to our data handling practices, clarified any open questions, and let them know their participation was entirely voluntary. They could skip questions or leave the interview at any time.

4 Results

Below, we detail the results from our qualitative analysis and complement it with additional insights from the Reddit analysis, where

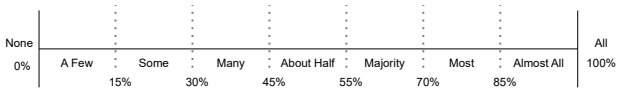

Figure 1: Qualifiers and their respective percentages as used to report our qualitative results. Graphic from Amft et al. [6].

appropriate. For our qualitative interview insights, we do not report exact numbers but rough qualifiers (see Figure 1). Exact numbers on individual codes' occurrence can be found in our codebook (extended version). For the Reddit analysis, we report descriptive statistics, as the sample was sufficiently large and as we achieved an appropriate IRR [51] (cf. Section 3.6.2).

4.1 AI Assistant Overview

To set the general context for the following subsections, we provide an overview of participants' AI assistants.

4.1.1 AI Assistants. We found participants widely use AI assistants in their professional work. Almost all reported using some AI assistant and doing so very often, most even daily, for various tasks (cf. Section 4.2.1). Participants mainly use ChatGPT and GitHub Copilot, which aligns with the results of SO's 2023 developer survey [69]. Participants also reported using other general-purpose chatbots by Google (Bard/Gemini) and Microsoft (Bing Chat), but more rarely. A few participants mentioned other models for coding, such as UniXcoder, Amazon CodeWhisperer, and Llama. However, they were used less often for various reasons, including secondary use of Llama when dealing with sensitive information or proprietary code that should not be shared with ChatGPT or when the primary AI assistant does not yield the anticipated results. Other participants tried assistants (e.g., Amazon CodeWhisperer) for a while, but then abandoned them in favor of ChatGPT or Copilot. An overview of participants' AI assistants is given in Table 1.

4.1.2 Experience with AI Assistants. All participants reported having used AI assistants previously. Most participants started using AI assistants after ChatGPT emerged in late 2022. Participants became aware of AI assistants through the widespread news and social media coverage around LLMs; peers, friends, and colleagues using AI assistants; or sometimes, via clients requesting AI features.

4.1.3 Motivations for Using AI Assistants. Participants reported several motivations for using AI assistants. While some reported security-related motivations, these were rare. Some participants mentioned that AI assistants could support them as a security expert in their work:

> *"For the security point, there are a lot of checks that maybe, as a developer, I couldn't be aware of. Security is exactly one of those points that are not for humans because I believe a lot in machine solutions."* — P1

Those participants anticipate AI assistants conducting comprehensive security checks or advising them with more security expertise than they have themselves.

However, most participants were motivated by the increased productivity and time savings when using AI assistants. One participant explained this by saying: *"It can absolutely support us, it can make us more efficient at our jobs, and [...] if I become more efficient, I need less headcount to do the same amount of work."* (P17). Along those lines, a few participants reported that using AI assistants can save money—for security, P4 stated that AI assistants save money compared to expensive security scanners. Given the productivity improvements, some participants used AI assistants to stay competitive, learn new or enhance their skills in software development. About half also mentioned a general curiosity in AI and new technology. Participants also reported certain tasks (discussed in Section 4.2.1) as use-cases motivating their AI assistant use, e.g., generating code or retrieving information.

4.2 Usage of AI Assistants

We asked participants how they used AI assistants for software engineering and security (RQ1). Below, we report the tasks they perform with AI assistants, participants' associated concerns, how their professional context constrains usage, and how they validate AI-generated code.

4.2.1 Tasks. Similar to the motivations above, we find—while some participants use AI assistants for security-specific tasks in software development (e.g., threat modeling, identifying vulnerabilities)—participants used AI assistants for many tasks throughout the software development life cycle (SDLC), e.g., generating code, writing documentation or requirements. Although the latter are not primarily security-focused, they have security implications.

Security Tasks: Many participants reported using AI assistants for security tasks. Identifying vulnerabilities and fixing security bugs in code were mentioned the most by some participants: *"If you just grab some pieces of code that are exploitable and just paste them in ChatGPT [...] Probably the AI is going to find out some changes for you to make the code more secure."* (P15). Some participants used AI assistants earlier in the SDLC. P4 mentioned using ChatGPT for threat modeling: *"What we are doing is using all this prompt engineering and giving as much information as possible to the tool and help us in defining the threat models rather than doing manual work."* (P4). While P4 mentioned AI assistants are not perfect, they at least provide a solid starting point for manual refinement, and sometimes, they would have forgotten the AI-suggested attack vectors otherwise. P15 also mentioned using AI assistants to create an exploit, and P12 mentioned AI assistants helped them explain results from static analysis tools. We found our software engineers used AI assistants slightly more for security tasks, like checking for vulnerabilities, than those primarily focused on security, e.g., security testers. One explanation is that participants with a strong security background assume AI performs worse on security tasks.

Notably, only two Reddit posts specifically targeted security and no comments. While rarely discussing security explicitly, commenters pointed out that AI-generated code is often of low quality and should not be relied on (P=3, C=24)[2], which could include security issues. One commenter explained *"It [AI] will lead to tech debt and shabbily maintained and written code."* This AI skepticism was the most common response to posts indicating a use or interest in using AI assistants for code generation. On average, posts about code generation received 0.62 comments indicating AI-generated code should be thoroughly scrutinized.

[2] C denotes the number of comments about a topic, P the number of posts.

Table 1: Overview of the 27 interviews, participants, and AI assistants they use.

ID	Duration	#Codes [1]	Recruitment	Country	Industry Exp. [2]	Role	ChatGPT	GitHub Copilot	Bard/Gemini	Self-hosted AI	Bing Chat	Llama	Anthropic Claude	Microsoft Copilot	CodeWhisperer	Tabnine	CodeT5	UniXcoder	Code Bird	Colab AI	Jarvis	Jurassic-1	JupiterOne J1 AI
P01	00:46:54	52	Network	Italy	>25	SW Eng.	●	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○
P02	00:50:34	69	Network	France	0–5	SW Sec. Eng.	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P03	00:49:30	69	Network	Viet Nam	11–15	Director	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P04	00:43:45	97	Upwork	India	16–20	Director	●	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	●
P05	01:04:12	107	Upwork	UK	6–10	SW Eng.	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P06	00:39:37	75	Student	USA	0–5	SW Eng.	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P07	00:53:18	97	Student	India	6–10	SW Eng.	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P08	00:54:44	81	Network	UK	11–15	Sec. Expert	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P09	00:51:34	103	Student	USA	6–10	SW Eng.	●	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○
P10	00:57:31	58	Network	Italy	6–10	SW Eng.	●	●	○	○	○	●	○	○	●	○	○	○	○	○	○	○	○
P11	00:58:53	68	Network	USA	>25	Sec. Expert	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P12	00:58:18	127	Network	Canada	6–10	Tech Lead	●	○	○	○	●	○	○	○	○	○	●	●	●	○	○	○	○
P13	00:43:06	67	Upwork	India	0–5	Pen. Tester	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P14	01:00:56	80	Upwork	USA	0–5	SW Eng.	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P15	00:45:36	76	Network	Brazil	21–25	SW Eng.	●	●	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○
P16	01:06:12	129	Network	UK	6–10	SW Eng.	●	●	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○
P17	00:45:15	104	Network	UK	16–20	Director	●	●	●	●	○	○	●	○	○	○	○	○	○	○	○	●	○
P18	01:10:28	46	Network	USA	21–25	SW Eng.	●	●	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○
P19	01:02:45	129	Upwork	India	21–25	Tech Lead	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P20	01:11:30	74	Network	USA	11–15	SW Eng.	●	●	●	●	○	●	○	○	○	○	○	○	○	○	○	○	○
P21	01:11:02	83	Upwork	UAE	>25	Tech Lead	●	●	●	●	●	○	○	●	○	○	○	○	●	●	○	○	○
P22	01:00:54	117	Upwork	USA	16–20	Tech Lead	●	○	○	●	○	●	●	○	○	○	○	○	○	○	○	○	○
P23	00:45:57	58	Upwork	Montenegro	0–5	ML Eng.	●	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○
P24	01:07:18	85	Upwork	Pakistan	6–10	Sec. Expert	●	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○
P25	00:50:05	77	Upwork	Poland	11–15	Sec. Expert	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P26	01:12:57	102	Upwork	Turkey	16–20	Pen. Tester	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P27	00:25:53	51	Upwork	Ukraine	0–5	SW Sec. Eng.	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sum		2,281					27	13	9	6	5	4	2	2	1	1	1	1	1	1	1	1	1

● Used the respective AI assistant. ○ Did not use it. [1] Number of codes assigned to the interview transcript. [2] Software industry experience in years.

Coding-Related Tasks: Most often, almost all participants reported using AI assistants for coding-related tasks. Almost all participants used AI assistants to generate code, as in this example:

> *"In most cases, I found the code written by ChatGPT to be a very good starting point. It's never perfect, but a very good starting point that we just need to add a few if-else statements to catch some edge cases or to fill in our credentials for certain databases."* — P12

This sentiment was common on Reddit, as 33 posts described using AI assistants to generate code or wanting to learn how to use AI assistants. Similarly, this was the second-most common topic of AI assistant-related Reddit comments (C=30). The only more common comment topic was whether AI assistants would replace developers altogether (P=9, C=48). Regarding code generation, a few participants stated that they used AI assistants to translate code into other programming languages. At the same time, participants said they might not understand code for unfamiliar programming languages: *"I have close to no experience with Rust and I guess I can ask ChatGPT to produce Rust code for me, even though I would not be able to actually understand whether it's correct or not."* (P2).

Some participants reported using ChatGPT for debugging code, fixing bugs, or explaining code: *"The thing I love about things like ChatGPT is it doesn't just give you the answer, it explains why, especially if you're asking it for code it will tell you: […] here's what it does."* (P17). Minor other use cases were related to code quality, such as refactoring, optimizing, and reviewing code. These uses were reflected in the Reddit discourse, with several posters and commenters describing using or wanting to use AI assistants for debugging (P=7, C=0) or explanations (P=7, C=4). Few discussed refactoring (P=2, C=0) and optimizing (P=4, C=10). One Redditor preferred using ChatGPT to do more straightforward coding tasks, saying they *"[Use] ChatGPT for automating the boring stuff like code refactoring, unit tests, and code documentation"* (redditor).

Information & Advice Source: The majority of participants reported consulting AI assistants as general sources for researching information, asking questions, and obtaining advice. About half of the participants said they use AI assistants as replacements for search engines (like Google) and online communities (like SO). Two participants fittingly describe it: *"Wherever I would formerly use StackOverflow, I now use OpenAI [ChatGPT]."* (P17) and *"Previously […] you would say, 'Have you Googled it?' Nowadays we'll say, 'Did you ask ChatGPT?'"* (P12).

Documentation & Requirement Analysis: Lastly, we found the majority of participants used AI assistants to perform tasks supporting application design and development in the SDLC—beyond coding. This includes requirement analysis, creating documentation and reports, and writing Jira stories (e.g., for bugs/issues). A few participants who conducted security tests said they write their security reports with AI assistance. Facing those tasks likely explains the higher popularity of ChatGPT compared to GitHub Copilot.

Tasks AI Assistants are not Used for: Some participants explicitly said not to use AI assistants for the above-mentioned tasks. For example, some participants stated not to use it for discovering vulnerabilities: *"Vulnerability wise, I don't think it does that good, just to identify that source code wise. […] Other premium scanners or some things would be doing a better job, I guess."* (P13). The concerns about AI assistant performance and capabilities were also prevalent

among other participants and prevented them from AI-assisted bug fixing, code reviews, or threat modeling: *"I even tried to use for the threat modeling, but it was so bad that it was […] just a nightmare. I just […][did] it from scratch by myself."* (P25).

Security Relevance of Non-Security Tasks: Considering code generation, the primary task is not security, but using insecure AI-suggested code might have severe security consequences. However, some participants explicitly mentioned that generated code has almost no security impact, as they would only create smaller snippets or not use them in production. P15 explained: *"I don't generate one page of code. It's just a few lines of code […]. Those are small functions; they don't have security concerns at all."* (P15). Participants predominantly expressed that security issues would be easier to spot when generating smaller code chunks, which is their typical use case. We cannot assess this hypothesis without future research—the participants' experience might be correct. Still, small snippets might be dangerous, e.g., when AI hallucinates packages [14, 46]. Besides code generation, searching and looking up information could be security-relevant; if the AI output is incorrect, software professionals might make decisions that undermine security.

4.2.2 Organizational Context & Privacy Constrain AI Assistant Usage. While most participants use AI assistants daily for various tasks, they reported that the professional context in their organization can constrain how AI assistants are used. When asked about security, participants did not mention constraints due to the security of the software they create but mainly privacy, legal, and indirect security concerns when using third-party AI assistants.

The primary concern among most participants was leaking sensitive data when using third-party AI assistants—either that the AI provider is breached or their inputs are used for training LLMs and might be reproduced by future models. A few participants stated that they feared their code or internal knowledge might be leaked and used by attackers, e.g., to find and exploit vulnerabilities in their code. Due to these concerns, many participants reported using AI assistants only in one direction: using the AI-generated code, but never supplying their code to the model.

However, participants were mainly concerned about leaking proprietary information and code, confidential company data, violating non-disclosure agreements, license agreements, or other contracts, or leaking otherwise protected data or PII. Consequently, participants police themselves on what they supply as inputs to AI assistants. P09 describes this fittingly:

> *"I cannot simply copy code to a ChatGPT or other AI assistant because they're going to put it on a larger pool of data and supply it everywhere. The security measures in our company or any company in general wouldn't permit us to do those things. We'll have to break down the problem statement and essentially ask only the context, as if asking another person who is not in our company."* — P09

Due to leakage concerns, some organizations set AI usage policies (Section 4.3.3) or desire self-hosting models or getting privacy guarantees (Section 4.4.3). While this sentiment was not common on Reddit, one commenter, in response to a post about using AI assistants, warned *"[it] returns entire snippets of copyrighted code without any attribution."* We expect the discrepancy between interviews and Reddit comes from the focus on internal organization policies.

Participants reported other minor constraints that were largely unrelated to security. This includes copyright infringement and intellectual property violations when using AI assistants that reproduce training data and high AI assistant costs, e.g., for subscriptions or operation costs if self-hosting. For most, the costs were outweighed by productivity improvements.

4.2.3 Quality Assurance of AI-Generated Code. Generally, participants reported checking AI-generated outputs, especially code, before using them. This is grounded in a general mistrust due to correctness and reliability issues that participants experienced, which they also translate to security (Section 4.3.1). One participant said: *"I just think there are vulnerabilities and there are things that it doesn't know. Currently, and at least for the next five years, I think all code written by AI will need to be gone over by a professional."* (P14). Overall, we found participants to commonly follow a three-step process to check AI-generated code, as depicted in Figure 2:

(1) Manual Inspection: First, almost all participants said to inspect the generated code and check for anomalies or issues: *"I don't completely trust them, but I at least read over their code."* (P19). While some participants mentioned specifically checking for security issues, the majority was more concerned about functional correctness—one even said not to check security at all.

(2) Copy, Execute, and Fix Suggested Code: Next, many participants reported copying and executing the suggested code to check whether it works. If not, they fix it manually or with the AI assistant's help. However, a few participants indicated not adopting the generated code but using it as a blueprint, re-implementing the final code entirely on their own: *"[I] reuse it without copying, but just reading, understanding how it works, and doing it by myself."* (P27).

(3) Peer Review & Software Testing: Third, about half of the participants reported that they complement their checks with peer reviews before merging code. The respective reviewers varied depending on the organizational structure and resources: Participants reported other team members, team leads, quality assurance teams, or dedicated security experts/teams. We suspect a higher prevalence of peer reviewing, as it is a common practice not specific to AI-generated code that participants might not report. Some participants said they did not distinguish human and AI-generated code and apply the same reviewing and testing procedures:

> *"This isn't something that we introduced because of generative AI […]. We've always had a full SDLC. […] the same rules apply as every other piece of code you write. Code that's generated by the AI goes to the exact same review process […]"* — P17

Similarly, many participants used various forms of software testing to validate AI-generated code. This included classical forms of software testing like unit tests, static analysis tools, and fuzzing. Some participants asked the AI assistant who generated the code to check it or cross-check it with another AI assistant.

Key Findings: Usage of AI Assistants (RQ1).

- AI assistants are used for various security tasks, such as threat modeling or vulnerability detection, and security-relevant tasks (e.g., code generation) in the SDLC. Moreover, participants consult AI assistants with general questions and for advice, replacing SO and Google.
- In the corporate context, the main concern is privacy—not security of software created with its help—which constrains AI assistant usage.

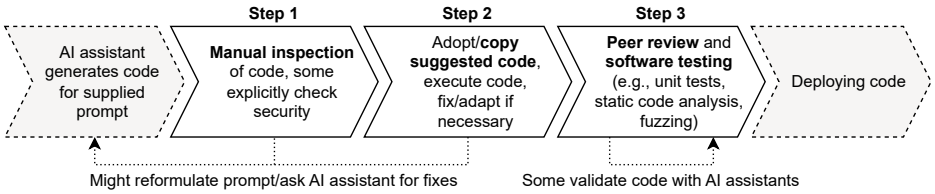

Figure 2: Our participants described three steps to inspect AI-generated code.

- While AI-generated code is often directly copied to codebases, it undergoes quality assurance similar to human-written code, including peer review and software testing.

4.3 Security Concerns & Considerations

Below, we cover participants' security concerns and considerations (RQ2). As participants largely showed mistrust in AI suggestions, we report on reasons for their mistrust. Moreover, we report on AI assistant usage policies in the participants' corporate context.

4.3.1 Security Concerns and AI Assistant Challenges. Overall, participants generally mistrusted AI assistants for security due to several challenges they experienced.

Mistrust and Blind Trust: Most participants mistrusted AI assistants and their generated suggestions, as they doubt the security of its suggestions or more indirectly the correctness of suggestions:

> *"Right now, I will not be able to trust the code produced by ChatGPT in security-sensitive scenarios […] It's mostly like I cannot really trust the correctness of this code, I'm not sure I should trust the security properties of it either."* — P02

Despite this general mistrust, many participants feared negative security impacts as they expect some software professionals might trust AI blindly, not questioning security: *"I worry that people will lean on or depend too much on the things generated by AI and forget about the security."* (P03). Relatedly, some participants complained that AI assistants are always confident, even if the suggestions are wrong or insecure. Instead, participants desired AI assistants to indicate their confidence, similar to how humans would express uncertainty about a solution. Many participants also mentioned that wrong AI suggestions are hard to recognize.

Poor AI Suggestion Quality: Participants expressed the above mistrust concerning poor AI suggestion quality—for general suggestions and AI-generated code. Most participants reported overall quality issues, such as inaccurate, outdated results—due to older datasets on which LLMs were trained—or hallucinations. A few also expressed that quality degraded over time.

Almost all participants expressed quality problems when generating code with AI assistants. This mainly concerned the necessity to review and rework code (Section 4.2.3) as the majority experienced that code did not work as intended or were even concerned that AI might otherwise introduce bugs: *"I don't think we've ever taken anything directly from the AI […] straight into code. Even after all the checks, I think everything's had to go through and be subtly changed."* (P17). Especially for more complex problems and when generating larger amounts of code, about half the participants reported lower quality. A few participants found AI-generated code challenging to refactor. This aligns with some participants who said AI-generated code was hard to understand and fix.

These issues likely relate to the interaction challenges the participants mentioned with AI assistants. For most participants, that concerned prompting and prompt engineering, i.e., steering the AI assistants to generate what the user desires. Many participants reported needing to change their prompts in multiple iterations until they were satisfied with the AI-generated answers. One participant said: *"You end up just having to both tweak the prompts and then tweak the code […] to actually fit your purpose."* (P08). Fittingly, many participants struggled to provide the LLM with enough context to create high quality results, mentioning the limited context windows and numbers of tokens that LLMs can process, or request limits in AI assistants. P06 summarizes all this well: *"Code generated by AI [might not] be safe because it does not know the entire code, it just gives relation to my question of what I asked. I don't feel it's completely safe. I do check back."* (P06).

Actual Security Issues through AI Assistants: Despite the widespread security and quality concerns, participants rarely reported facing security issues using AI assistants. Instead, experiences were mixed. While a few were unsure, some participants did not perceive any change in their projects' security since adopting AI assistants. Only a few mentioned security improvements, like identifying a new attack vector in their software. A few others, however, reported security shortcomings in the generated code:

> *"For example, it doesn't hash the password, it doesn't add salt to the password unless you specifically tell it to do that. […] Generally, it's not very secure. We will generally find mistakes in the majority of code snippets that it creates."* — P08

Nonetheless, others did not experience security issues despite all other quality shortcomings: *"It definitely writes bad or suboptimal code in some places, but I don't frequently see glaringly obvious security vulnerabilities being created by the AI."* (P17).

Security Concerns: Most participants expressed security concerns when using AI assistants. Many doubted its ability for security tasks, as one said: *"I would never rely on ChatGPT for security."* (P26).

First, about half the participants found that AI assistants will likely introduce security issues through generated code: *"I think the more code will be produced by the current existing AI tooling, the more we'll see security bugs in them, and we'll probably see new patterns of security bugs."* (P02).

Second, some participants were concerned about LLM poisoning and facing models trained to create insecure suggestions. A few outlined that they would not be able to recognize this: *"What if it is developed using a data set that is inherently vulnerable? […] Those challenges are there."* (P04). Similarly, a few were concerned about

AI assistants acting as a malicious dependency: *"Actually, anything that's produced also has that vulnerability."* (P08).

Third, some participants questioned AI assistants' suitability for security at all or partly: *"ChatGPT is much more limited in cyber security, but it's very good in code generation and programming."* (P24). For example, participants found it performs worse than static application security testing (SAST) tools in finding vulnerabilities; P13 and P26 argued that current AI models, like GPT and Codex, cannot handle more complex security tasks. A few participants experienced that ChatGPT needs to be actively directed into a *"security mindset"* in their prompts and questioned why this is not a default: *"Honestly, I'm not so satisfied with the security level of the code because ChatGPT doesn't include it by default. If you don't ask it, it doesn't include the security steps in the code."* (P27).

4.3.2 AI Assistants' Ethics Safeguards on Security Tasks: Some participants reported that the *ethical safeguards* built into AI assistants (also called *guardrails* or *constraints*) rejected their prompts for security tasks, e.g., finding vulnerabilities. However, several participants explained that they had reframed their prompts to circumvent the AI assistants' safeguards and have it assist with a vulnerability:

> *"In ChatGPT, for example, if I'm asking how to do an SQL injection, it will say 'SQL injection is an unethical thing, sorry, I can't help you with that.' If I'm asking in some […] indirect way, it will explain to me all the details."* — P07

Therefore, participants perceived it more like a circumventable usability obstacle, but not an actual constraint. However, this might not be enough to leverage AI assistants for more offensive security tasks, even when circumventing ethics constraints. For example, P25 explained that AI assistants can suggest possible attack vectors but will not perform the attack for them:

> *"It gives you some tricks. […][and] possible attack vectors, but it will not make the attack instead of you. If you don't […] understand how the basic attack works, just using one command that ChatGPT gives you, will not make you a successful attacker or hacker."* — P25

4.3.3 Policy & Regulations on AI Usage. As large tech companies, such as Apple or Google, have banned the use of AI assistants for security and quality reasons, we asked participants about policies on using AI in their work.

About half of the participants stated their company did not have a policy regulating the use of AI assistants. Participants who work in companies that recently started using AI assistants argued that policies were not considered as they needed to test the boundaries and use cases of AI assistants. One participant declines policies and explicitly said: *"No, absolutely no. No policy interference!"* (P10). Self-employed participants did not require a policy, since they work for themselves. Some participants advocate for self-policing and argue that a policy is not required when following common sense, e.g., not sharing sensitive data. A few participants expressed that the need for policies when using AI assistants is security-relevant, especially when using it to generate code that, therefore, needs to be tested (see Section 4.2.3). These participants desire a standardized verification process to evaluate and verify the security of the generated code before merging it to the main code base.

Besides policy absence, other participants reported having policies. A few even reported that their companies banned AI assistants by policy, but mainly for privacy reasons, as outlined in Section 4.2.2. One explained how their company developed a custom AI assistant based on public LLM APIs, which serves as a proxy that filters requests based on the company policy before sending the prompt to the third-party LLM. Another participant explained that AI assistants remain forbidden by default but can be used when clients authorize their data to be shared with AI assistants.

Participants rarely reported shadow practices, while a few assumed AI assistants usage anyway, even if forbidden. P16 justified their use of ChatGPT, even though their company requires Bing Enterprise because they perceived the former to perform better.

4.3.4 Responsibility for AI-Generated Code. For a fictive scenario, we asked about responsibility when AI-generated code is used in production, only to find it vulnerable and exploited later. Almost all participants agreed the human who uses the AI assistant remains responsible, while a few said their company is also responsible. Participants mainly argued that an AI assistant is a tool and does not replace the developers' agency, but developers must check suggestions before using them. P01 compared it to copying and pasting code from SO, stating that it is the human's responsibility that the code works as intended. P08, however, argued that their company would be responsible as it should have ensured a code review process, specifically for AI-generated code. Only a few other participants said they consider the AI assistant creators responsible.

4.3.5 Security Performance: Human vs. AI. We asked participants who wrote the more secure code, comparing humans and AI. Overall, participants' opinions differed. About half of them argued that humans would create more secure software. On the contrary, some expected AI to perform better. A few participants said that AI assistants currently perform at the level of junior human developers but expect AI to become better than humans. While P02 currently expected neither to perform well, some participants perceived AI assistants and humans to complement each other, therefore achieving the best security when AI assists humans:

> *"I think that the most secure code would be a combination of the two. I think that both I and the model alone would generate code with insecurity. I think that the combination of both me and the model would write the most secure code."* — P14

Key Findings: Security Concerns & Considerations (RQ2).

- When using AI assistants, participants consider security, indicated through many security concerns, but only a few faced actual security issues. The overall (security) mistrust in AI assistants is primarily due to code quality concerns.
- Participants mainly assume humans to perform better security-wise than AI assistants, and perceive humans to remain responsible, as AI is just an assisting tool.
- AI assistant usage policies are rare and mainly motivated by privacy concerns, but some participants desire policies to ensure secure usage.

4.4 Expected Future Impact of AI Assistants on Security and Development Practices

Lastly, our participants shared their views on the future impact of AI assistants on software development, the changes they expect to software development, their influence on software security, and their wishes for future usage of AI tools (RQ3).

4.4.1 Future Impact on Security. Participants were undecided regarding the future impact of AI assistants on security. Many expected AI tools to help with security, as they believed AI tools would be able to support developers with security during software development. For example, they imagined AI could find common vulnerabilities and take over static code analysis. Some stated, however, that the quality and accuracy of the tools would need to improve for use in security tasks: *"If we can make sure AI spits out perfectly secure code, which it will be capable of doing along the line, then AI will be improving the overall cybersecurity posture."* (P26).

In contrast, about half of the developers expected using AI assistants to impact software security negatively. For example, some noted that the quality of AI tools was not yet high enough to ensure security, but they expected this to be the case in the future:

> *"For the first time, using AI tools for writing applications, we will have more vulnerabilities, and we will need to fix them. However, with time, the AI model will learn some details, and I think it will be fixed."* — P27

A few others suspected developers of blindly trusting the AI output during software development and about software security, resulting in vulnerabilities being introduced into the software: *"If you're just blindly […][using AI and check] the code in without proper testing or without proper reviewing, I think there is a very high chance that there can be a security flaw in that."* (P9).

Further, some had concerns that AI assistants could effectively be used by potential attackers, with AI tools being able to find and exploit common vulnerabilities, making it possible for attackers with little technical knowledge to perform attacks:

> *"I think decades ago, script kids referred to kids who get access to some dangerous piece of code. They do not necessarily know what the code does, but when they run it, it's very damaging. Nowadays, the kids just need to express what they want to achieve, and then ChatGPT will write some bad code for them."* — P12

4.4.2 Expected Changes to the Software Development Process. Overall, participants expected AI assistants to become more involved in the software development process, taking over more mundane tasks, thus shifting the developers' responsibilities toward complex tasks that AI assistants cannot solve.

Participants mentioned a variety of tasks for which they want to use AI in the future. The majority expected to be able to use AI on security-relevant tasks. This includes code generation, security reviews, vulnerability detection, software testing, and malware analysis. However, many participants agreed that the suggestion quality needs to improve for such tasks. Some participants expect this to happen in the future: *"I think over the course of time, in the next three, two, or four years, definitely AI will write better code than developers. That is something more secure and better."* (P4). Participants expect AI assistance for other tasks, like maintaining code, installing and updating libraries, or software design.

Some participants speculated that AI tools might improve through highly task-specific training, e.g., tools specifically trained for software security instead of models for general use. They expect these tools would perform better and would trust them more:

> *"It will be really hard to trust and rely on the [general] models when there are security properties required. However, I guess that if a model was developed with a security-first principle, focused on security, the story might be different […]"* — P2

Many participants expected significant shifts in software developers' responsibilities, such as developers becoming prompt engineers or becoming an AI supervisor who primarily evaluates AI suggestions. About half of the participants did not expect AI assistants to replace humans in software development. Instead, they suspect AI assistants will speed up tedious and time-consuming tasks, shifting software developers' tasks with more time for high-level tasks. Thus, the majority shared a generally positive outlook on AI's influence on its future integration into the SDLC: *"I don't think it will ever replace a developer. […] I think it's more likely that we, as a software engineering industry, would do less and less of those mundane tasks and more of the interesting stuff."* (P16). However, some were worried about their job, as AI might perform many current software developer tasks if they further improve. This fear was mostly caused by AI assistants' ability to solve many tasks very quickly compared to human developers.

4.4.3 Wishes. Participants hoped that the usability of AI assistants would improve along with the quality of the generated output. For example, some developers hoped AI tools would be better integrated into IDEs. A few others wanted easier methods to prompt the AI, e.g., voice commands or the ability to pass drawings and diagrams to the AI to explain program structures easily. This was combined with the wish for technical improvements of the AI systems, for example, a larger context window:

> *"Generative AI will get more powerful and can process more context and broader context to generate better codes, and maybe in the future, a whole project, which will need very minor modifications from the human user or developer, probably."* — P20

However, many participants mentioned that the output quality needs to improve before AI assistants could be of greater help to them, as they had issues with the quality and correctness of the AI output in the past (Section 4.3.1). About half were confident that the quality would improve in the future, with a few unsure how fast these tools could improve: *"I think they still have a long way to go. There is a lot more training that they should undergo. They have to improve."* (P10). A few participants mentioned observing a drop in the quality of the AI output over time, claiming the AI needs to be trained with higher-quality data.

As most participants had concerns regarding leaking sensitive data and IP through third-party AI assistants (Section 4.2.2), some desire and expect increased use of self-hosted and specialized AI assistants within companies:

> *"The thing is, if you are having your local model, which is, again, coming to the security and the privacy, that's the only way that your data is not accessed. […][If] this model on your server, it's not going outside your network."* — P21

Key Findings: Expected Future Impact (RQ3).

- Participants expect their role to shift from writing code to more creative and complex tasks, leaving the mundane for AI assistants under their supervision.
- Participants desire improvements in AI assistant quality, correctness, and security abilities.
- Some participants envision AI assistants to improve in security tasks, while others argue for a negative impact.

5 Discussion

Below, we discuss our results by setting them in context and deriving recommendations on usage and future AI assistants.

5.1 (Mis)Trust in AI Assistants

While our participants largely maintain a critical mistrust towards AI assistants, they widely use them in software development (e.g., to generate code) at the same time. Although this mistrust applies to security (e.g., generating vulnerable code), participants reported security issues rarely. They used other aspects, like functionality and correctness of AI suggestions, as a proxy to assess AI assistants' security performance. That said, mistrusting AI assistant security is likely due to overall quality issues that participants widely experience. Currently, this skepticism leads participants to use AI assistants with care and scrutinize AI suggestions. Changes to these proxy indicators might affect developer behavior. For example, future quality improvements might lead to blindly trusting AI assistants and not checking code suggestions' security before using them. Further, getting more used to AI assistants might have similar effects and could be expected for such a novel technology. However, this hypothesis needs to be investigated in future research. The Reddit discourse analysis broadly aligns with the interview findings, as Redditors and our participants use AI assistants for various software development tasks. We found similar mistrust in AI assistants, as interviewees and Redditors expressed the need to scrutinize AI suggestions.

5.2 Comparison with Related Work

5.2.1 Mismatch with Prior Experimental Results. As this study contributes qualitative insights that complement prior experiments on the security impact of AI assistants, a comparison finds a major mismatch: While our participants reported to critically scrutinize AI suggestions (Section 4.2.3) due to general mistrust and did not perceive negative security impact from AI usage, a negative security impact is evident in practice [49, 59, 61, 64, 68]. This mismatch reveals a skewed self-perception, so that software professionals overestimate their capabilities in scrutinizing AI suggestion for security. Nonetheless, we conclude that software professionals are aware of potential security issues due to AI assistant usage and try to "stay awake" [59], but seem to lack methods and support to effectively validate AI suggestions.

5.2.2 Security Capabilities of AI Assistants. Our participants feel that AI capabilities are still limited and unreliable, while being a supportive tool at the same time. However, they generally believe LLMs will become more helpful in assisting with security in the future (Section 4.4). Currently, research found mixed capabilities in AI assistants, ranging from poor quality and insecure suggestions [49, 59, 61, 64] to autonomously outperforming CTF players [33, 53, 60, 66, 67, 70]. As our participants did not perceive such immense benefits, this supports that studies might overestimate AI security performance [22, 73] when used in practice. Based on our interviews, we hypothesize that challenges when using AI assistants, e.g., providing enough context, engineering prompts, or ethics safeguards, currently prevent leveraging the full potential that might be achievable in theory and under ideal lab conditions.

5.3 AI Assistants as a (Novel) Source of Advice

We found our assumption confirmed that AI assistants are a new advice source. Participants reported to have primarily replaced classical online advice sources, like Google and SO, by using AI assistants (Section 4.2.1). We see similar usage patterns, such as copying, pasting, and adapting code (Section 4.2.3)—that are known to cause security issues, e.g., when copying from SO [27]. Comparably, the research community also found software professionals to achieve worse security when using AI assistants compared to not using them [49, 59, 61, 64]. While recent research found ChatGPT not to entirely replace SO, 35% preferred the former due to its language characteristics and comprehensiveness—even with a large portion of incorrect answers [42].

Considering AI assistants a (partial) replacement for other online advice sources, it remains an open question how AI assistants impact online knowledge communities in which they have been trained (partly). Recently, Burtch et al. found AI assistants degrade online communities and reduce the number of users on SO [10]. This could cause a "vicious cycle" of feedback when it drains the online communities on which it is trained. Following that argument and given the often poor quality and insecure suggestions on SO, creators of AI assistants need to be aware of this problem and prevent reinforcing insecure suggestions [21].

5.4 Recommendations

Below, we give recommendations for AI assistant users and creators:

5.4.1 Critically Scrutinizing AI Suggestions. We advocate, similar to other work that demonstrated the security shortcomings of AI code assistants [59], to critically validate all AI suggestions. Despite the found mismatch that questions its feasibility for software professionals (Section 5.2.1), we argue that awareness of AI unreliability and potential security issues is important nonetheless—and required for critical scrutiny. While our participants often already showed this awareness, we underline the need to educate software professionals and companies about potential security issues arising from AI usage, e.g., package hallucinations [14, 46].

How to scrutinize AI suggestions (and generally ensuring code security) remains an open question. As a rule of thumb, we recommend treating AI-generated code like human code and applying the same quality assurance measures, e.g., code reviews, software testing, static analysis, or pentesting. Many software professionals and companies already had such structured processes (cf. Figure 2).

Given potential security decreases, we argue that AI assistant usage should be considered depending on the security guarantees needed in a software project on a case-by-case basis. Currently, our participants are concerned that AI assistants do not outperform humans with expert security knowledge, raising the question of why AI is used for security-critical tasks. Given that participants (need to) check the AI suggestions, humans with sufficient knowledge and skills are still required to do these checks anyway.

5.4.2 Improving Model Quality and Security. Given the current widespread usage of AI assistants, which can be expected to become even more ubiquitous, reducing security issues at the model level is likely to have the highest impact. Our participant's quality and security concerns were reflected in the demand for improved future

models (Section 4.4.3). As our participants, we anticipate further AI assistant improvements. Along with the general improvements in AI assistant quality and performance, AI creators should also ensure suggestions are reliable and secure to avoid risk to downstream users of software built with AI suggestions.

We argue that models with security capabilities are needed if AI assistants are used in software engineering. For example, coding models must be constrained to generate secure code suggestions (at least at a high rate). As this can largely depend on the LLMs' training data, we advocate rethinking what data is used for training. While current models are trained on large corpora of online content, e.g., from GitHub or SO, it is no surprise that the resulting AI suggestions might be insecure given many insecure code snippets online [27–29]. Using datasets with better quality and security could also make AI suggestions more secure. For existing models, security hardening techniques should be considered [38].

We hypothesize that using task-specific models, e.g., for vulnerability detection, threat modelling, or secure code generation, instead of general-purpose models might result in better quality and security capabilities. For example, HackerOne recently launched *Hai* beta, an AI assistant specifically tailored to vulnerability intelligence tasks, e.g., to assist with vulnerability remediation [35].

5.4.3 Leveraging Prompt Engineering. To improve and get the best possible AI assistant suggestions, we recommend software professionals to leverage prompt engineering. Also, the AI assistant creators suggest prompt engineering, e.g., OpenAI [58], indicating this is necessary to circumvent low-quality suggestions like our participants reported. When not satisfied with the first suggestion, software professionals should try to iteratively refine their prompts, e.g., starting with simple queries, then providing more context, being more specific, or splitting a problem in smaller sub-problems. Many participants already apply known prompt engineering techniques [26] by adapting their prompts to obtain the desired AI suggestions (Section 4.3.1). We recommend learning about and exploring prompt engineering practice guides [26, 58, 65]. Still, prompt engineering remains a significant usability obstacle, limiting AI assistant usefulness [48].

5.4.4 Shifting from Compliance-Driven to Security-Driven Policies. Interestingly, the companies participants work for seem less concerned about the security of AI-suggested code. Instead, they view data usage and privacy aspects (Section 4.3.3) as the main motivation behind AI assistant usage policies—although Google did ban AI assistants due to security concerns [20]. Given many participants shared AI assistant security concerns, one explanation for compliance-driven policies is that management or legal teams create them, but not software professionals, as our participants were rarely involved. Another explanation is that AI suggestions are evaluated like human code (Section 4.2.3), not needing a dedicated policy. Nonetheless, we think data and privacy leakage concerns are important aspects and need to be considered by companies; recently, Niu et al. uncovered that about 8% of prompts to the GitHub Copilot models result in privacy leaks [55]. Overall, we acknowledge that using AI assistants is a trade-off between security, privacy, cost, efficiency, and liability. We call companies to remember to consider security in this trade-off.

5.4.5 Prioritizing Usage of Privacy-Friendly AI Assistants. A significant participant concern was leaking data and sensitive information when using AI assistants (Section 4.2.2), which also results in policies regulating usage (Section 4.3.3). Other researchers also found these privacy concerns among general users of AI assistants, and we can confirm trade-offs between privacy and utility [75] for software professionals. Consequently, participants desire self-hosted AI assistants, i.e., not involving a third party, or private ones, i.e., hosted by a third party but with privacy guarantees (e.g., no training on prompts). The latter might be interesting if the hardware is unavailable, e.g., to host models like Llama [52]. We recommend software professionals and companies to consider these more privacy-friendly variants of AI assistants. The creators of AI assistants should offer their models either for self-hosting or in a private subscription. The industry recognizes this need already; for example, GitHub recently launched *Copilot Enterprise* [16, 24]. When also fine-tuning such models (e.g., on a company's code base), this might improve quality and security of AI suggestions.

5.4.6 Balancing Ethical Concerns and Using AI Assistants for Security. As participants reported, one limitation to using AI assistants for security tasks are the implemented ethics safeguards. AI assistants might refuse prompts they deem unethical, e.g., more offensive security tasks like identifying a vulnerability or creating an exploit (Section 4.3.2). While these ethical considerations are important, they create a dilemma, as vulnerabilities must be found and fixed to improve security. For ethical usage, e.g., security evaluations of one's software, this can limit AI assistants' usefulness—for valid use cases that would be done by human security experts otherwise. Further, participants reported circumventing safeguards with prompt engineering. While we advocate the creation of AI assistants for specific security tasks, we believe it is necessary to discuss the ethics first and to implement the respective ethical constraints that cannot be easily circumvented. This is also necessary as specific security AI assistants like *Hai* [35] emerge.

5.5 Outlook & Future Work

5.5.1 General-Purpose & Code AI Assistants. Table 1 confirms the wide usage of ChatGPT and other general-purpose AI assistants among software professionals [69], even more than coding assistants like GitHub Copilot. However, we perceived a strong focus on AI code assistants like Copilot in security research [49, 59, 61, 64]. Future research should close this gap and consider both coding-related and general-purpose AI assistants (which are also used for coding tasks), e.g., comparing the security impact of using both kinds. Similarly, the creators of general-purpose models should also consider the security impact when their models are capable of and used for software development tasks.

5.5.2 AI Assistants vs. Other Advice Sources. Considering AI assistants as novel advice sources (Section 5.3), the question of whether software professionals deal differently with AI assistants' suggestions and other advice sources arises. Hence, we advocate experiments to compare the security impact of AI assistants to other advice sources (e.g., Google, SO), similar to earlier related work [1].

6 Conclusion

We investigated software professionals' usage of AI assistants in software development, focusing on security and their security considerations in 27 interviews, complemented by the analysis of Reddit posts. Besides being used often by almost all participants, we found that both coding AI assistants, like Copilot, and general-purpose AI assistants, like ChatGPT, are widely used for security-critical software development tasks (e.g., code generation, threat modeling, code reviews, and vulnerability detection). Despite ubiquitous usage, we found that our participants mistrust and check AI suggestions. While security is a primary concern, only a few participants reported negative experiences with AI assistants in the past. As our results qualitatively complement prior experiments [49, 59, 61, 64], a comparison reveals a mismatch between our participants' reported scrutiny and actual code security when using AI assistance. This indicates that software professionals overestimate how well they can scrutinize AI suggestions. A contributing factor is likely that participants reasoned about AI assistant security capabilities based on proxies such as functionality. Overall, we conclude that AI assistants change software development by being a novel source of security and security-relevant advice for software professionals.

Acknowledgments

We thank our anonymous reviewers and shepherd for their valuable feedback and for helping us to improve this paper. We also acknowledge *Dagstuhl Seminar 23181* in which most authors participated and where this project started. This research was funded by VolkswagenStiftung Niedersächsisches Vorab – ZN3695. This research was also partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CaSa – 390781972 and supported by the U.S. National Science Foundation (NSF) Award # 2247141 and Award # 2312321. The research was also partly supported by European Union Horizon Europe program - Cybersecurity *Sec4AI4Sec* Award # 101120393 and by NWO, Dutch Research Organization - Kennisen Innovatieconvenant (KIC) *HEWSTI* Award # KICH1.VE01.20.004. This work was also supported by EPSRC Grants *REPHRAIN: National Research Centre on Privacy, Harm Reduction and Adversarial Influence Online* (EPSRC Grant EP/V011189/1) and *Equitable Privacy* (EPSRC Grant EP/W025361/1).

Availability

To support transparency, replication, and meta-studies, we provide the following research artifacts: (1) our recruitment materials, (2) the screening questionnaire, (3) the interview guide, (4) the demographics questionnaire, and (5) the list of posts from the Reddit analysis. We do not provide interview transcripts to protect our participants' privacy. The replication package is available as supplementary material and at https://doi.org/10.17605/OSF.IO/XZ72H.

References

[1] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. 2016. You Get Where You're Looking For: The Impact of Information Sources on Code Security. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 289–305.

[2] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. 2017. How Internet Resources Might Be Helping You Develop Faster but Less Securely. *IEEE Security & Privacy* 15, 2 (2017), 50–60.

[3] Yasemin Acar, Christian Stransky, Dominik Wermke, Charles Weir, Michelle L Mazurek, and Sascha Fahl. 2017. Developers Need Support Too: A Survey of Security Advice for Software Developers. In *2017 IEEE Cybersecurity Development (SecDev)*. IEEE, 22–26.

[4] ACM Publications Board. 2021. *ACM Publications Policy on Research Involving Human Participants and Subjects*. https://www.acm.org/publications/policies/research-involving-human-participants-and-subjects

[5] Amberscript Global B.V. 2024. *Amberscript*. https://www.amberscript.com

[6] Sabrina Amft, Sandra Höltervennhoff, Rebecca Panskus, Karola Marky, and Sascha Fahl. 2024. Everyone for Themselves? A Qualitative Study about Individual Security Setups of Open Source Software Contributors. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 249–249.

[7] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[8] Virginia Braun and Victoria Clarke. 2024. *Got questions about TA? We have prepared some answers to some of the common ones we receive.* https://www.thematicanalysis.net/faqs/

[9] Virginia Braun, Victoria Clarke, and Nikki Hayfield. 2022. 'A starting point for your journey, not a map': Nikki Hayfield in conversation with Virginia Braun and Victoria Clarke about thematic analysis. *Qualitative Research in Psychology* 19, 2 (2022), 424–445.

[10] Gordon Burtch, Dokyun Lee, and Zhichen Chen. 2024. Generative AI Degrades Online Communities. *Commun. ACM* 67, 3 (Feb. 2024), 40–42.

[11] Karoline Busse, Julia Schäfer, and Matthew Smith. 2019. Replication: No One Can Hack My Mind Revisiting a Study on Expert and Non-Expert Security Practices and Advice. In *Proc. 15th Symposium on Usable Privacy and Security (SOUPS'19)*. USENIX, 117–136.

[12] David Byrne. 2021. A worked example of Braun and Clarke's approach to reflexive thematic analysis. *Quality & Quantity* 56, 3 (June 2021), 1391–1412.

[13] Mengsu Chen, Felix Fischer, Na Meng, Xiaoyin Wang, and Jens Grossklags. 2019. How Reliable is the Crowdsourced Knowledge of Security Implementation?. In *Proc. IEEE/ACM 41st International Conference on Software Engineering (ICSE'19)*. IEEE, 536–547.

[14] Thomas Claburn. 2024. AI hallucinates software packages and devs download them – even if potentially poisoned with malware. *The Register* (2024). https://www.theregister.com/2024/03/28/ai_bots_hallucinate_software_packages/ Accessed: 2024-04-04.

[15] Victoria Clarke, Virginia Braun, and Nikki Hayfield. 2015. Thematic analysis. *Qualitative psychology: A practical guide to research methods* 3 (2015), 222–248.

[16] Copilot. 2024. *About GitHub Copilot Enterprise*. https://docs.github.com/en/copilot/github-copilot-enterprise/overview/about-github-copilot-enterprise

[17] Juliet Corbin and Anselm Strauss. 2014. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage Publications.

[18] Anastasia Danilova, Stefan Horstmann, Matthew Smith, and Alena Naiakshina. 2022. Testing time limits in screener questions for online surveys with programmers. In *Proc. 44th International Conference on Software Engineering (ICSE '22)*. ACM, 2080–2090.

[19] Anastasia Danilova, Alena Naiakshina, Stefan Horstmann, and Matthew Smith. 2021. Do you really code? Designing and Evaluating Screening Questions for Online Surveys with Programmers. In *Proc. 43rd International Conference on Software Engineering (ICSE '21)*. IEEE, 537–548.

[20] Jeffrey Dastin and Anna Tong. 2023. Focus: Google, one of AI's biggest backers, warns own staff about chatbots. *Reuters* (2023). https://www.reuters.com/technology/google-one-ais-biggest-backers-warns-own-staff-about-chatbots-2023-06-15/ Accessed: 2024-03-01.

[21] Randall Degges. 2024. *Copilot amplifies insecure codebases by replicating vulnerabilities in your projects*. Technical Report. snyk. https://snyk.io/blog/copilot-amplifies-insecure-codebases-by-replicating-vulnerabilities/

[22] Yangruibo Ding, Yanjun Fu, Omniyyah Ibrahim, Chawin Sitawarin, Xinyun Chen, Basel Alomair, David Wagner, Baishakhi Ray, and Yizheng Chen. 2024. Vulnerability Detection with Code Language Models: How Far Are We? arXiv:2403.18624 [cs.SE]

[23] Thomas Dohmke. 2023. The economic impact of the AI-powered developer lifecycle and lessons from GitHub Copilot. https://github.blog/2023-06-27-the-economic-impact-of-the-ai-powered-developer-lifecycle-and-lessons-from-github-copilot/ Accessed: 2024-03-01.

[24] Thomas Dohmke. 2024. *GitHub Copilot Enterprise is now generally available.* https://github.blog/2024-02-27-github-copilot-enterprise-is-now-generally-available/ Accessed: 2024-04-17.

[25] Thomas Dohmke, Marco Iansiti, and Greg Richards. 2023. Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle. arXiv:2306.15033 [econ.GN]

[26] Oluwole Fagbohun, Rachel M. Harrison, and Anton Dereventsov. 2024. An Empirical Categorization of Prompting Techniques for Large Language Models: A Practitioner's Guide. arXiv:2402.14837 [cs.CL]

[27] Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. 2017. Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security. In *2017 IEEE*

Symposium on Security and Privacy (SP). IEEE, 121–136.
[28] Felix Fischer, Yannick Stachelscheid, and Jens Grossklags. 2021. The Effect of Google Search on Software Security: Unobtrusive Security Interventions via Content Re-Ranking. In *Proc. 28th ACM Conference on Computer and Communication Security (CCS'21)*. ACM, 3070–3084.
[29] Felix Fischer, Huang Xiao, Ching-Yu Kao, Yannick Stachelscheid, Benjamin Johnson, Danial Razar, Paul Fawkesley, Nat Buckley, Konstantin Böttinger, Paul Muntean, and Jens Grossklags. 2019. Stack Overflow Considered Helpful! Deep Learning Security Nudges Towards Stronger Cryptography. In *Proc. 28th Usenix Security Symposium (SEC'19)*. USENIX, 339–356.
[30] Foundation Inc. 2024. Reddit Statistics for 2024: Eye-Opening Usage & Traffic Data. https://foundationinc.co/lab/reddit-statistics/ Accessed: 2024-07-24.
[31] Nat Friedman. 2021. Introducing GitHub Copilot: your AI pair programmer. https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/ Accessed: 2024-03-01.
[32] GitHub. 2024. GitHub Copilot · Your AI pair programmer. https://github.com/features/copilot Accessed: 2024-03-01.
[33] Sergei Glazunov and Mark Brand. 2024. *Project Naptime: Evaluating Offensive Security Capabilities of Large Language Models*. https://googleprojectzero.blogspot.com/2024/06/project-naptime.html
[34] Mark Gurman. 2023. Samsung Bans Staff's AI Use After Spotting ChatGPT Data Leak. *Bloomberg* (2023). https://www.bloomberg.com/news/articles/2023-05-02/samsung-bans-chatgpt-and-other-generative-ai-use-by-staff-after-leak Accessed: 2024-03-01.
[35] HackerOne. 2024. *Hai: The AI Assistant for Vulnerability Intelligence*. https://www.hackerone.com/ai/hai-ai-assistant-vulnerability-intelligence
[36] H. Hajipour, K. Hassler, T. Holz, L. Schonherr, and M. Fritz. 2024. CodeLMSec Benchmark: Systematically Evaluating and Finding Security Vulnerabilities in Black-Box Code Language Models. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE, 684–709.
[37] Andrew F Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication methods and measures* 1, 1 (2007), 77–89.
[38] Jingxuan He and Martin Vechev. 2023. Large Language Models for Code: Security Hardening and Adversarial Testing. In *Proc. 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, 1865–1879.
[39] Sameera Horawalavithana, Abhishek Bhattacharjee, Renhao Liu, Nazim Choudhury, Lawrence O. Hall, and Adriana Iamnitchi. 2019. Mentions of Security Vulnerabilities on Reddit, Twitter and GitHub. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI '19)*. ACM, 200–207.
[40] Iulia Ion, Rob Reeder, and Sunny Consolvo. 2015. "...No one Can Hack My Mind": Comparing Expert and Non-Expert Security Practices. In *Proc. 11th Symposium On Usable Privacy and Security (SOUPS'15)*. USENIX, 327–346.
[41] Tahira Iqbal, Moniba Khan, Kuldar Taveter, and Norbert Seyff. 2021. Mining Reddit as a New Source for Software Requirements. In *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE, 128–138.
[42] Samia Kabir, David N. Udo-Imeh, Bonan Kou, and Tianyi Zhang. 2024. Is Stack Overflow Obsolete? An Empirical Study of the Characteristics of ChatGPT Answers to Stack Overflow Questions. In *Proc. CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, Article 935, 17 pages.
[43] Harjot Kaur, Sabrina Amft, Daniel Votipka, Yasemin Acar, and Sascha Fahl. 2022. Where to Recruit for Security Development Studies: Comparing Six Software Developer Samples. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX, 4041–4058.
[44] Erin Kenneally and David Dittrich. 2012. *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research*. Technical Report. U.S. Department of Homeland Security. https://www.dhs.gov/sites/default/files/publications/CSD-MenloPrinciplesCORE-20120803_1.pdf
[45] Jan H. Klemmer, Marco Gutfleisch, Christian Stransky, Yasemin Acar, M. Angela Sasse, and Sascha Fahl. 2023. "Make Them Change it Every Week!": A Qualitative Exploration of Online Developer Advice on Usable and Secure Authentication. In *Proc. 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, 2740–2754.
[46] Bar Lanyado. 2024. Diving Deeper into AI Package Hallucinations. https://www.lasso.security/blog/ai-package-hallucinations#heads-up-hallucinated-packages-in-the-wild Accessed: 2024-04-04.
[47] Tianshi Li, Elizabeth Louie, Laura Dabbish, and Jason I. Hong. 2021. How Developers Talk About Personal Data and What It Means for User Privacy: A Case Study of a Developer Forum on Reddit. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3, Article 220 (Jan. 2021), 28 pages.
[48] Jenny T. Liang, Chenyang Yang, and Brad A. Myers. 2024. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. In *Proc. 46th IEEE/ACM International Conference on Software Engineering (ICSE '24)*. ACM, Article 52, 13 pages.
[49] Vahid Majdinasab, Michael Joshua Bishop, Shawn Rasheed, Arghavan Moradidakhel, Amjed Tahir, and Foutse Khomh. 2023. Assessing the Security of GitHub Copilot Generated Code – A Targeted Replication Study. arXiv:2311.11177 [cs.SE]
[50] Michelle Mazurek. 2022. We Are the Experts, and We Are the Problem: The Security Advice Fiasco. In *Proc. 29th ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*. ACM, 7. Keynote.
[51] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *ACM on Human-Computer Interaction* 3, CSCW, Article 72 (2019), 23 pages.
[52] Meta. 2024. *Meta Llama*. https://llama.meta.com/
[53] Stephen Moskal, Sam Laney, Erik Hemberg, and Una-May O'Reilly. 2023. LLMs Killed the Script Kiddie: How Agents Supported by Large Language Models Change the Landscape of Network Threat Testing. arXiv:2310.06936 [cs.CR]
[54] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. 2017. Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study. In *Proc. 24th ACM Conference on Computer and Communication Security (CCS'17)*. ACM, 311–328.
[55] Liang Niu, Shujaat Mirza, Zayd Maradni, and Christina Pöpper. 2023. CodexLeaks: Privacy Leaks from Code Generation Language Models in GitHub Copilot. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX, 2133–2150.
[56] OpenAI. 2021. OpenAI Codex. https://openai.com/index/openai-codex/
[57] OpenAI. 2022. Introducing ChatGPT. https://openai.com/blog/chatgpt Accessed: 2024-03-01.
[58] OpenAI. 2024. *Prompt engineering*. https://platform.openai.com/docs/guides/prompt-engineering Accessed: 2024-04-17.
[59] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2022. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 754–768.
[60] Hammond Pearce, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, and Brendan Dolan-Gavitt. 2022. Pop Quiz! Can a Large Language Model Help With Reverse Engineering? arXiv:2202.01142 [cs.SE]
[61] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants?. In *Proc. 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, 2785–2799.
[62] Elissa M. Redmiles, Noel Warford, Amritha Jayanti, Aravind Koneru, Sean Kross, Miraida Morales, Rock Stevens, and Michelle L. Mazurek. 2020. A Comprehensive Quality Evaluation of Security and Privacy Advice on the Web. In *Proc. 29th USENIX Security Symposium (SEC'20)*. USENIX, 89–108.
[63] Robert W. Reeder, Iulia Ion, and Sunny Consolvo. 2017. 152 Simple Steps to Stay Safe Online: Security Advice for Non-Tech-Savvy Users. *IEEE Security and Privacy* 15, 5 (2017), 55–64.
[64] Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. 2023. Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX, 2205–2222.
[65] Shubhra Kanti Karmaker Santu and Dongji Feng. 2023. TELeR: A General Taxonomy of LLM Prompts for Benchmarking Complex Tasks. arXiv:2305.11430 [cs.AI]
[66] Minghao Shao, Boyuan Chen, Sofija Jancheska, Brendan Dolan-Gavitt, Siddharth Garg, Ramesh Karri, and Muhammad Shafique. 2024. An Empirical Evaluation of LLMs for Solving Offensive Security Challenges. (2024). arXiv:2402.11814 [cs.CR]
[67] Minghao Shao, Sofija Jancheska, Meet Udeshi, Brendan Dolan-Gavitt, Haoran Xi, Kimberly Milner, Boyuan Chen, Max Yin, Siddharth Garg, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, and Muhammad Shafique. 2024. NYU CTF Dataset: A Scalable Open-Source Benchmark Dataset for Evaluating LLMs in Offensive Security. (2024). arXiv:2406.05590 [cs.CR]
[68] snyk. 2023. *AI Code, Security, and Trust: Organizations Must Change Their Approach*. Technical Report. snyk. https://snyk.io/reports/ai-code-security/
[69] Stack Overflow. 2023. Stack Overflow Developer Survey 2023. https://survey.stackoverflow.co/2023/ Accessed: 2024-03-01.
[70] Wesley Tann, Yuancheng Liu, Jun Heng Sim, Choon Meng Seah, and Ee-Chien Chang. 2023. Using Large Language Models for Cybersecurity Capture-The-Flag Challenges and Certification Questions. arXiv:2308.10443 [cs.AI]
[71] Brandon Vigliarolo. 2023. Apple becomes the latest company to ban ChatGPT for internal use. *The Register* (2023). https://www.theregister.com/2023/05/19/apple_chatgpt/ Accessed: 2024-03-01.
[72] Daniel Votipka, Desiree Abrokwa, and Michelle L. Mazurek. 2020. Building and Validating a Scale for Secure Software Development Self-Efficacy. In *Proc. 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. ACM, 1–20.
[73] Fangzhou Wu, Qingzhao Zhang, Ati Priya Bajaj, Tiffany Bao, Ning Zhang, Ruoyu "Fish" Wang, and Chaowei Xiao. 2023. Exploring the Limits of ChatGPT in Software Security Applications. arXiv:2312.05275 [cs.CR]
[74] Xin-Li Yang, David Lo, Xin Xia, Zhi-Yuan Wan, and Jian-Ling Sun. 2016. What security questions do developers ask? a large-scale study of stack overflow posts. *Journal of Computer Science and Technology* 31, 5 (2016), 910–924.
[75] Zhiping Zhang, Michelle Jia, Hao-Ping Lee, Bingsheng Yao, Sauvik Das, Ada Lerner, Dakuo Wang, and Tianshi Li. 2024. "It's a Fair Game", or Is It? Examining How Users Navigate Disclosure Risks and Benefits When Using LLM-Based Conversational Agents. In *Proc. 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, 1–26.