Bridging the Gap Between User Interface Security and CI/CD Workflows

Commonwealth Cyber Initiative (CCI) Southwest Virginia
Cybersecurity Research 2025

Chris Brown (PI)
Assistant Professor
Department of Computer Science
Virginia Tech
dcbrown@vt.edu

General Abstract: Software, a set of instructions to execute tasks and manipulate data on devices, impacts nearly every facet of modern life. Humans interact with software-based applications through user interfaces (UIs) which handle the input and output for programs—such as web applications hosted online and accessed through internet browsers. To enhance the delivery of software to users, continuous integration and continuous deployment (CI/CD) techniques have been introduced to automate tasks for more efficient testing and sharing of software.

However, as the complexity and demand for technology increases, user interfaces grow increasingly difficult to design and secure while web-based attacks become more sophisticated. Research findings have posited a wide variety of automated tools to enhance the security of software, such as dynamic application security testing (DAST) tools which can assess the security of web applications by simulating attacks on user interfaces. However, these tools are often avoided in practice—with open source software developers facing difficulties integrating UI test cases into CI/CD workflows and often relying on more ad hoc and manual approaches to test the security of their software.

To this end, the proposed work aims to bridge the gap between UI testing and CI/CD pipelines in the context of securing web applications. We accomplish this by designing, implementing, and evaluating a novel tool that leverages large language models (LLMS) that synthesize DAST tool results to support finding and repairing UI-based security vulnerabilities. This work provides implications and motivates future research to further secure user interfaces and safeguard user experiences with software.

1 Introduction

User interfaces (UIs) define the modes through which humans interact with software-based systems. UIs are notably difficult to design and implement in software development contexts, with UI design reportedly consuming over half of source code and development time [25]. UI functions can also affect the quality of software applications and user experiences [5]. To verify the behavior of software, Graphic User Interface (GUI) testing is crucial to inspect visual elements and ensure that software function as intended. However, UI testing is challenge due to dependence on user input, output relying on the layout of visual elements, and rapid changes made to UIs by developers [24]. Further, many security vulnerabilities can be exploited through UIs [23]–such as cross site scripting, access management issues, and information leakage. To mitigate this, prior work has explored Dynamic Application Security Testing (DAST) tools assess web application security by simulating attacks to its user interface [12].

Rapid release processes are essential in modern software development to efficiently deploy software to users. For example, continuous integration and continuous deployment (CI/CD) platforms provide functionality for developers to stage code changes, build projects, test software, and deploy to users more frequently and reliably [30]. CI/CD is commonly adopted in open source projects [35], and has shown to be effective for automating many different types of testing—including security testing [28]. However, integrating GUI testing in CI/CD presents novel challenges due to the complexity of user interfaces [26]. For instance, developers report writing UI tests is a pain point with CI/CD workflows [38]. Thus, as web-based attacks become more prevalent and complex [3], novel solutions are needed to assess the security of UIs in rapid release environments.

Project Objectives: The primary objective of this project is to enhance user interface security in CI/CD workflows for open source projects. Our ongoing work supported by CCI is collecting insights from software practitioners to understand challenges and solutions to securing user interfaces in rapid release environments. The proposed work will implement and evaluate a tool that leverages large language models (LLMs) to mitigate the reported challenges and support effective UI vulnerability detection and repair in open source software.

Intellectual Merit: This project will advance the state of knowledge in cybersecurity and software engineering (SE) research. The PI will systematically implement and evaluate a novel system that uses generative AI to automatically translate DAST security tool output for web applications into comprehensible reports for developers. The successful completion of this project will strengthen development infrastructure by producing a technical solution to support UI security testing in CI/CD workflows. We also provide implications and motivate future work to further secure UIs in modern software development processes.

Alignment with CCI: The proposed research will advance the state of the art in cybersecurity research by leveraging emerging technologies, in particular machine learning-based LLMs, to innovate the security of web-based software. This work will also catalyze future research regarding the use of machine learning techniques to promote security in society by preventing vulnerabilities and enhancing user experiences in user interfaces.

2 Research Plan

The goal of the proposed work is to develop tooling to support securing user interfaces on CI/CD workflows. In particular, we will implement and evaluate an intelligent system to incorporate user interface security techniques for web applications into modern development workflows.

2.1 Tool Development

Based on the identified challenges from our preliminary work, our first research activity will produce a system to incorporate GUI security testing in CI/CD project workflows for web applications. The goal of the system will be to increase awareness of vulnerabilities in user interfaces. The following design guidelines derived from our ongoing work will be used to implement our system.

1) LLM-Powered. To automatically notify open source developers of user interface issues in CI/CD pipelines, our tool will leverage large language models (LLMs) fine-tuned on output from GUI security testing tools. Software engineers frequently seek help from LLMs (*i.e.,* ChatGPT[1]) for various software development and security tasks [6, 16, 37]. LLMs have also been shown to be effective for summarizing natural language [4]. Moreover, our prior work shows that LLMs are preferred for summarizing the output from DAST tools, with developers finding summarized versions of security reports significantly more understandable and clear than traditional reports [32]. This project will further explore the capabilities of LLMs for providing insight into web application security testing vulnerabilities.

Note: Despite the aforementioned benefits of LLMs, we forsee two main limitations to this approach. First, LLMs have been shown to incorporate security vulnerabilities—for instance, in generating insecure code [20]. Second, LLMs have been shown to "hallucinate", or provide incorrect responses to input prompts portrayed as correct by the model [29]. To mitigate these issues, we will only use generative AI models to summarize the output from existing secure and trusted web application security testing tools (*i.e.,* OWASP ZAP [2] and Burp Suite [1]). This will limit the possibility of insecure or inaccurate output from LLMs by summarizing existing information without generating new knowledge or code. Future work can explore further leveraging the capabilities of LLMs for detecting and repairing GUI security vulnerabilities. For instance, training custom models on DAST reports and source code for software systems.

2) Concise. The second design goal of our system will be to have concise output. Prior work shows that developers prefer concise information in communications with teammates [7], code review feedback [11], and development tool recommendations [10]. Alternatively, research shows developers often fail to adopt static analysis tools [19] and SAST tools [31] due to incomprehensible output—especially in agile development processes [33]. Our preliminary work suggests developers prefer summarized DAST reports over traditional ones. To this end, we aim to provide concise information to provide developers with clearer summaries of reported vulnerabilities. However, to support devel-

[1] `https://chat.openai.com`

opers with more expertise or who desire more information, we will also provide features for users to access the full report generated by DAST tools.

3) Workflow Integration. Our final design guideline is to implement a tool that integrates into the existing workflow of open source developers. Prior research shows that developers are frustrated with tools that interrupt or conflict with their existing processes [9, 19], which can lead to decreased productivity and lower code quality [35] as well as cognitive overload for developers [18]. Our prior work on general user interface testing in CI/CD processes also shows that inconsistent environments and increased setup are challenges with testing user interfaces in CI/CD workflows [15]. Thus, to improve developers' awareness of UI security vulnerabilities in CI/CD processes for open source software, we aim to incorporate our tool into the existing project workflows.

These guidelines will motivate the design of our system. The specific features and implementation of our system will be based on our ongoing work engaging with open source developers to gain insights on the challenges and workaround solutions used to secure user interfaces in rapid release environments. Here, we provide specific examples of potential systems that could be implemented based on the design guidelines: (1) an automated bot that utilizes pull request review comments to notify developers of potential UI vulnerabilities on the line(s) of code in question; (2) a CI/CD plugin to automatically generate a GitHub issue[2] with summarized details of detected vulnerabilities; or (3) an extension to an existing open source CI/CD platform that allows users to visualize web application security testing results alongside other test results for project builds (*i.e.*, unit and integration testing).

2.2 Tool Evaluation

To evaluate our system, we will conduct a user study evaluation to understand its usability and effectiveness for enhancing user interface security in CI/CD projects. This preliminary study will answer the following research questions: **RQ1: How accurate are LLMs for summarizing DAST tool reports?** and **RQ2: How effective are LLMs for summarizing DAST tool reports in CI/CD workflows?**. Our study will receive IRB approval and follow the empirical SE guidelines for controlled experiments with human subjects [21].

Recruitment. We will recruit open source developers working on various projects to participate in our research study. Potential subjects will be recruited via purposive sampling based on contributors to GitHub repositories that use popular CI/CD platforms and DAST tools. If this does not provide a sufficient sample, we will broaden our reach via recruitment on social media. A pre-survey will be used in the recruitment phase to collect demographic information and ensure participants have adequate background and experience to participate. We aim to recruit at least 25 participants, and will compensate subjects $40 each for completing the study. We anticipate all study sessions will no more than one hour and will be conducted virtually over Zoom.

Task Design. We will ask participants to complete tasks to find and fix a UI vulnerability in two settings. The baseline condition will have participants use an out-of-the-box DAST

[2] `https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues`

tool (*i.e.,* in an IDE) to find and fix an issue. The treatment condition will incorporate our tool. The task will be given in a mock repository with a vulnerability introduced and known by the research team. The study settings will be switched across participants to avoid ordering bias. We will employ a think aloud protocol for participants to verbalize their thoughts when completing both tasks.

Debriefing. We will conduct a semi-structured interview immediately following the completion of the study tasks to gain more insight on participants' experiences in both settings. This will be to gain more insight on the advantages, disadvantages, and improvements to be made to our system. We will further assess the usability of our system with a post-survey based on the System Usability Scale (SUS) [8].

Data Analysis. All study sessions will be recorded and transcribed by the research team to be analyzed retroactively. We will inspect recordings to investigate the correctness of the LLM-generated summary of reports found by our tool. We will also use thematic analysis techniques to derive themes from qualitative responses regarding our tool and its effectiveness for finding and fixing security vulnerabilities in UIs.

3 Milestones and Deliverables

The project milestones are shown in Figure 1. Specific deliverables are described below.

Dissemination. Our findings will be submitted to peer-reviewed conferences and workshops related to SE, HCI, and Cybersecurity. Potential venues include Foundations of Software Engineering (FSE), Automated Software Engineering (ASE), Human Factors in Computing Systems (CHI), Visual Languages and Human-Centric Computing (VL/HCC), the USENIX Security Symposium, the Symposium on the Science of Security (HotSoS), and the Secure Development Conference (SecDev). Beyond academic venues, we will also target disseminating our work in industry-focused conferences, such as All Things Open[3] and BSides Roanoke[4], and engage with colleagues in the CCI.

We will also provide additional deliverables to broaden the impacts of this work. The completion of this project will contribute to the degree requirements (*i.e.,* thesis) and workforce development (*i.e.,* CCI Symposium attendance) of student researchers. We will publish relevant findings via publicly available resources (*i.e.,* blogs) to share our results with practitioners and the general public.

Tool Availability and Extension. The output from this project will be made open source and publicly available in a GitHub repository. The research team will also participate in an initial meeting with a representative from LINK+LICENSE+LAUNCH—an initiative at Virginia Tech focused on corporate partnerships, commercialization, and start-ups—to discuss the viability and interest of commercializing the research deliverables from this proposed work. Based on the results of our preliminary study, we aim to make improvements to our tool and conduct a larger scale study. We also anticipate extending this work to further improve additional types of testing in further modern software development workflows (*i.e.,* DevOps) for UIs in other domains (*i.e.,* virtual reality). To support these

[3] `https://allthingsopen.org/`

[4] `https://bsidesroa.org/`

efforts, we will target external proposals, such as calls for programs within the NSF Computer and Information Science and Engineering (CISE) or a CAREER award submission.

Research Workshop. We will also host a topical workshop during the project period to convene a diverse community of researchers and practitioners. The one-day workshop will feature an overview of our findings as well as invited speakers and panel discussions related to security, user interface testing, and CI/CD workflows. The workshop will be shared broadly to researchers in the CCI, students and faculty at local institutions, and regional practitioner-focused communities (*i.e.*, NRV Dev[5]).

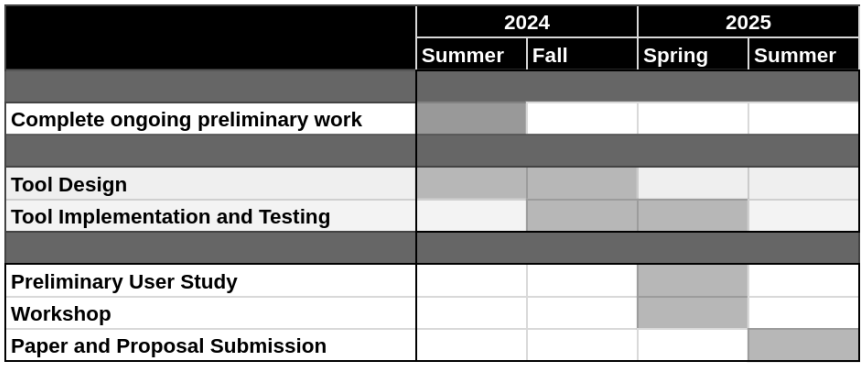

Figure 1: Timeline for the proposed research milestones

4 PI Qualifications

The PI is well-positioned to conduct the proposed research. The PI is experienced in investigating current practices and interventions to improve software development and CI/CD processes, including examining code reviews [11, 27], debugging [22], and security [33]. In addition, he has investigated the impact of machine learning and large language models to automatically repair programming bugs [17], provide automated feedback on technical interview practice [34], and solve computing problems in educational contexts [36]. Related to this proposal, preliminary work shows LLMs are effective for summarizing DAST tool reports [32] and open source developers face many challenges integrating UI testing tools, including Selenium,[6] Cypress,[7] and Playwright,[8] into CI/CD workflows [15]. PI Brown received funding from CCI to investigate the impact of data privacy policies in open source software [13, 14] and gain insights from open source developers on challenges and techniques to secure user interfaces in rapid release environments.[9] He has received a Google Award for Inclusion Research to improve technical interview preparation for under-resourced job seekers and funding from the Sloan Foundation to improve development practices among researchers from non-SE backgrounds.

[5] `https://nrv.dev/`
[6] `https://www.selenium.dev/`
[7] `https://www.cypress.io`
[8] `https://playwright.dev/`
[9] Data collection in progress

Bridging the Gap Between User Interface Security and CI/CD Workflows

Commonwealth Cyber Initiative (CCI) Southwest Virginia
Cybersecurity Research 2025

Chris Brown (PI)
Assistant Professor
Department of Computer Science
Virginia Tech
dcbrown@vt.edu

General Abstract: Software, a set of instructions to execute tasks and manipulate data on devices, impacts nearly every facet of modern life. Humans interact with software-based applications through user interfaces (UIs) which handle the input and output for programs—such as web applications hosted online and accessed through internet browsers. To enhance the delivery of software to users, continuous integration and continuous deployment (CI/CD) techniques have been introduced to automate tasks for more efficient testing and sharing of software.

However, as the complexity and demand for technology increases, user interfaces grow increasingly difficult to design and secure while web-based attacks become more sophisticated. Research findings have posited a wide variety of automated tools to enhance the security of software, such as dynamic application security testing (DAST) tools which can assess the security of web applications by simulating attacks on user interfaces. However, these tools are often avoided in practice—with open source software developers facing difficulties integrating UI test cases into CI/CD workflows and often relying on more ad hoc and manual approaches to test the security of their software.

To this end, the proposed work aims to bridge the gap between UI testing and CI/CD pipelines in the context of securing web applications. We accomplish this by designing, implementing, and evaluating a novel tool that leverages large language models (LLMS) that synthesize DAST tool results to support finding and repairing UI-based security vulnerabilities. This work provides implications and motivates future research to further secure user interfaces and safeguard user experiences with software.

5 Introduction

User interfaces (UIs) define the modes through which humans interact with software-based systems. UIs are notably difficult to design and implement in software development contexts, with UI design reportedly consuming over half of source code and development time [25]. UI functions can also affect the quality of software applications and user experiences [5]. To verify the behavior of software, Graphic User Interface (GUI) testing is crucial to inspect visual elements and ensure that software function as intended. However, UI testing is challenge due to dependence on user input, output relying on the layout of visual elements, and rapid changes made to UIs by developers [24]. Further, many security vulnerabilities can be exploited through UIs [23]–such as cross site scripting, access management issues, and information leakage. To mitigate this, prior work has explored Dynamic Application Security Testing (DAST) tools assess web application security by simulating attacks to its user interface [12].

Rapid release processes are essential in modern software development to efficiently deploy software to users. For example, continuous integration and continuous deployment (CI/CD) platforms provide functionality for developers to stage code changes, build projects, test software, and deploy to users more frequently and reliably [30]. CI/CD is commonly adopted in open source projects [35], and has shown to be effective for automating many different types of testing—including security testing [28]. However, integrating GUI testing in CI/CD presents novel challenges due to the complexity of user interfaces [26]. For instance, developers report writing UI tests is a pain point with CI/CD workflows [38]. Thus, as web-based attacks become more prevalent and complex [3], novel solutions are needed to assess the security of UIs in rapid release environments.

Reduced Scope: The original proposal sought to: a) implement an automated tool to support UI security in CI/CD environments; and b) conduct a user study to evaluate our tool. To reduce the scope of this project, funding from CCI will only support the tool development efforts. Our tool will leverage large language models to provide concise recommendations to developers in modern workflows regarding potential UI vulnerabilities. We will submit this work to Tool Demonstration tracks at software engineering-related research venues, and augment this effort with a future user study to gain insights from developers on the effectiveness of our system (expected Summer/Fall 2025).

Intellectual Merit: This project will advance knowledge in cybersecurity and software engineering (SE). The PI will systematically implement and evaluate a novel system that uses generative AI to automatically translate DAST security tool output for web applications into comprehensible reports for developers. The successful completion of this project will strengthen development infrastructure by producing a technical solution to support UI security testing in CI/CD workflows. We also provide implications and motivate future work to further secure UIs in modern software development processes.

Alignment with CCI: The proposed research will advance the state of the art in cybersecurity research by leveraging emerging technologies, in particular machine learning-based LLMs, to innovate the security of web-based software. This work will also catalyze future research regarding the use of machine learning techniques to promote security in society by preventing vulnerabilities and enhancing user experiences in user interfaces.

References

[1] Burp suite. https://portswigger.net/burp/pro.

[2] Owasp zap. https://www.zaproxy.org/.

[3] Browser-based phishing attacks increased 198% in h2 2023. *Security Magazine*, 2024. https://www.securitymagazine.com/articles/100343-browser-based-phishing-attacks-increased-198-in-h2-2023.

[4] T. Ahmed and P. Devanbu. Few-shot training llms for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–5, 2022.

[5] I. Banerjee, B. Nguyen, V. Garousi, and A. Memon. Graphical user interface (gui) testing: Systematic mapping and repository. *Information and Software Technology*, 55(10):1679–1694, 2013.

[6] L. Belzner, T. Gabor, and M. Wirsing. Large language model assisted software engineering: prospects, challenges, and a case study. In *International Conference on Bridging the Gap between AI and Reality*, pages 355–374. Springer, 2023.

[7] M. Blatt and A. Norman. Email, communication and more: How software engineers use and reflect upon email at the workplace. Master's thesis, 2013.

[8] J. Brooke. Sus: a retrospective. *Journal of usability studies*, 8(2):29–40, 2013.

[9] C. Brown and C. Parnin. Sorry to bother you: designing bots for effective recommendations. In *Proceedings of the 1st International Workshop on Bots in Software Engineering*, pages 54–58. IEEE Press, 2019.

[10] C. Brown and C. Parnin. Comparing different developer behavior recommendation styles. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 78–85, New York, NY, USA, 2020. Association for Computing Machinery.

[11] C. Brown and C. Parnin. Understanding the impact of github suggested changes on recommendations between developers. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 1065–1076, New York, NY, USA, 2020. Association for Computing Machinery.

[12] L. Dencheva. *Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools*. PhD thesis, Dublin, National College of Ireland, 2022.

[13] L. Franke, H. Liang, A. Brantly, J. C. Davis, and C. Brown. A first look at the general data protection regulation (gdpr) in open-source software. *arXiv preprint arXiv:2401.14629*, 2024.

[14] L. Franke, H. Liang, S. Farzanehpour, A. Brantly, J. C. Davis, and C. Brown. An exploratory mixed-methods study on general data protection regulation (gdpr) compliance in open-source software. In *In Submission to: International Symposium on Empirical Software Engineering and Measurement*, 2024.

[15] X. Gan, H. Liang, and C. Brown. Challenges, benefits, and strategies: A qualitative study on the integration of ui testing in ci/cd processes. In *In Submission to: International Symposium on Empirical Software Engineering and Measurement*, 2024.

[16] B. Grewal, W. Lu, S. Nadi, and C.-P. Bezemer. Analyzing developer use of chatgpt generated code in open source github projects. 2024.

[17] M. M. A. Haque, W. U. Ahmad, I. Lourentzou, and C. Brown. Fixeval: Execution-based evaluation of program fixes for programming problems. In *2023 IEEE/ACM International Workshop on Automated Program Repair (APR)*, pages 11–18. IEEE, 2023.

[18] J. Hyysalo, J. Lehto, S. Aaramaa, and M. Kelanti. Supporting cognitive work in software development workflows. In *Product-Focused Software Process Improvement: 14th International Conference, PROFES 2013, Paphos, Cyprus, June 12-14, 2013. Proceedings 14*, pages 20–34. Springer, 2013.

[19] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge. Why don't software developers use static analysis tools to find bugs? In *2013 35th International Conference on Software Engineering (ICSE)*, pages 672–681. IEEE, 2013.

[20] R. Khoury, A. R. Avila, J. Brunelle, and B. M. Camara. How secure is code generated by chatgpt? In *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2445–2451. IEEE, 2023.

[21] A. J. Ko, T. D. LaToza, and M. M. Burnett. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering*, 20(1):110–141, 2015.

[22] M. Ko, D. B. Bose, H. A. Chowdhury, M. Seyam, and C. Brown. Exploring the barriers and factors that influence debugger usage for students. In *Visual Languages and Human-Centric Computing*, 2023.

[23] M. Luo, O. Starov, N. Honarmand, and N. Nikiforakis. Hindsight: Understanding the evolution of ui vulnerabilities in mobile browsers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 149–162, 2017.

[24] A. M. Memon. Gui testing: Pitfalls and process. *Computer*, 35(08):87–88, 2002.

[25] B. A. Myers. State of the art in user interface software tools. *Readings in Human–Computer Interaction*, pages 323–343, 1995.

[26] M. Nass, E. Alégroth, and R. Feldt. Why many challenges with gui test automation (will) remain. *Information and Software Technology*, 138:106625, 2021.

[27] N. Palvannan and C. Brown. "Suggestion bot: Analyzing the impact of automated suggested changes on code reviews ". In *Proceedings of the 5th International Workshop on Bots in Software Engineering*. IEEE Press, 2023.

[28] T. Rangnau, R. v. Buijtenen, F. Fransen, and F. Turkmen. Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 145–154. IEEE, 2020.

[29] V. Rawte, S. Chakraborty, A. Pathak, A. Sarkar, S. Tonmoy, A. Chadha, A. P. Sheth, and A. Das. The troubling emergence of hallucination in large language models–an extensive definition, quantification, and prescriptive remediations. *arXiv preprint arXiv:2310.04988*, 2023.

[30] M. Shahin, M. Ali Babar, and L. Zhu. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943, 2017.

[31] J. Smith, L. N. Q. Do, and E. Murphy-Hill. Why can't johnny fix vulnerabilities: A usability evaluation of static analysis tools for security. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, pages 221–238, 2020.

[32] A. Thool and C. Brown. Harnessing the power of llms to simplify security: Llm summarization for human-centric dast reports. In *In Submission to: Visual Languages and Human-Centric Computing*, 2024.

[33] A. Thool and C. Brown. Securing agile: Assessing the impact of security activities on agile development. In *International Workshop on Secure Software: Challenges, Opportunities, and Lessons Learned*, 2024.

[34] S. Vaishampayan and C. Brown. Will you trust me more than chatgpt? evaluating user perceptions of llm-generated feedback for technical interviews. In *In Submission to: Visual Languages and Human-Centric Computing*, 2024.

[35] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov. Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pages 805–816, 2015.

[36] T. Wang, D. V. Díaz, C. Brown, and Y. Chen. Exploring the role of ai assistants in computer science education: Methods, implications, and instructor perspectives. In *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 92–102. IEEE, 2023.

[37] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. *arXiv preprint arXiv:2303.07839*, 2023.

[38] D. Widder, M. Hilton, C. Kästner, and B. Vasilescu. A conceptual replication of continuous integration pain points in the context of Travis CI. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE, pages 647–658. ACM, 2019.