Securing Agile: Assessing the Impact of Security Activities on Agile Development

Arpit Thool
Virginia Tech
Blacksburg, Virginia, USA
arpitthool@vt.edu

Chris Brown
Virginia Tech
Blacksburg, Virginia, USA
dcbrown@vt.edu

ABSTRACT

Software systems are expected to be secure and robust. To verify and ensure software security, it is vital to include *security activities*, or development practices to detect and prevent security vulnerabilities, into the software development process. Agile software development is a popular software engineering (SE) process used by many organizations and development teams. However, while Agile aims to be a lightweight and responsive process, security activities are typically more cumbersome and involve more documentation and tools–violating the core principles of Agile. This work investigates the impact of security activities on various aspects of Agile development. To understand how software engineers perceive incorporating security practices into Agile methodologies, we distributed an online survey to collect data from software practitioners with experience working in Agile teams. Our results from 34 survey participants show most software practitioners believe security activities are beneficial to development overall but lack confidence in their impact on the security of software systems. Our findings provide insight into how security activities affect Agile development and provide implications to help SE teams better incorporate security activities into implementing Agile development processes.

CCS CONCEPTS

• **Software and its engineering** → **Agile software development**; • **Security and privacy** → **Social aspects of security and privacy**.

KEYWORDS

Agile, Software Engineering, Security Activities

ACM Reference Format:
Arpit Thool and Chris Brown. 2024. Securing Agile: Assessing the Impact of Security Activities on Agile Development. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3661167.3661280

EASE 2024, June 18–21, 2024, Salerno, Italy

ACM ISBN 979-8-4007-1701-7/24/06
https://doi.org/10.1145/3661167.3661280

1 INTRODUCTION

Software security is crucial for preventing attacks, safeguarding data, and ensuring the quality of software systems. To support software engineers, a wide variety of *security activities* have been introduced to automate and streamline security-related development tasks. For example, security-oriented static analysis tools, such as FindSecurityBugs[1] and Bandit[2], have been implemented to analyze source code and detect vulnerabilities early during the development process [51].

Developing secure computing systems becomes increasingly difficult as software becomes more prevalent and complex. To organize software development efforts, software engineering (SE) teams adopt SE processes to produce quality applications. *Agile* is a software development framework that puts emphasis on flexible and iterative processes [45]. Numerous SE processes apply Agile methodologies, including Scrum, Kanban, and Extreme Programming (XP) [46]. Agile methodologies have become increasingly popular over the last decade and are now widely adopted by software development teams. Previous research has suggested that Agile methods enhance team productivity and product quality [37, 40], foster communication [40] and knowledge sharing [37], but contrasting studies have pointed out potential adverse effects of Agile practices on software security [9, 21]. For example, Agile development emphasizes reduced documentation, while security-based code analysis and scanning tools often produce more documentation to report potential issues and risk analyses. There are efforts to integrate security into Agile in the form of security activities [5, 29]. For example, DevOps— a modern development methodology commonly combined with Agile principles [34]—focuses on automating development and infrastructure tasks to support rapid releases of software to users [26]. *DevSecOps* extends this not just to Development (Dev) and Operations (Ops), but also covers Security (Sec) by incorporating security activities, such as automated static analysis tools, into DevOps pipelines [31, 36]. However, more research is needed to understand how these security activities merge into Agile, affect its various facets, and how Agile practitioners perceive them. Our work aims to understand software practitioners' perspectives and experiences integrating security activities into Agile practices. Our study explores the following research questions (RQs):

> **RQ1** How do software practitioners perceive the effectiveness of adopted and state-of-the-art security practices, and what is their level of willingness to incorporate them into the Agile software development process?

[1] https://find-sec-bugs.github.io/
[2] https://bandit.readthedocs.io/

RQ2 How are the team velocity and productivity, as perceived by the software practitioners, affected by the inclusion of security activities?
RQ3 What is the impact of integrating security activities into Agile development on software practitioners' confidence in their software product and organization?

To answer these research questions, we used an online survey to collect data from 34 software practitioners with experience working in Agile development teams. By answering the research questions, we aim to contribute valuable insights into the security activities adopted by Agile development teams to safeguard their products. Additionally, we seek to understand software practitioners' perceptions of the effectiveness and willingness to adopt these security practices. This includes evaluating the impact of security activities on team productivity and *velocity*, a metric measuring the productivity of Agile teams by the number of tasks they can complete in a sprint [58]. Along with evaluating their effect on software practitioners' confidence in their software products and overall organization. Ultimately, through this work, we aim to contribute insights for improving the adoption of techniques to secure complex software systems.

This article is organized as follows: In Section 2, we provide a glossary and background information on Security and Agile Methodology. Section 3 reviews related work. Section 4 explains the methodology we used to address our research questions. Section 5 presents the data analysis results. In Section 6, we discuss our results. Section 7 outlines the study's limitations and suggests potential future work, while Section 8 concludes.

2 BACKGROUND

2.1 Security

Secure software systems safeguard users' data and prevent malicious attacks. Compromised software security can lead to detrimental consequences for software users and companies. For example, the Equifax data breach in 2017–involving one of the largest credit reporting agencies in the U.S.–saw hackers exploit a vulnerability to access and expose the sensitive personal identifiable information (PII) of approximately 147 million users [32]. Further, this led to billions of dollars lost in market value and numerous lawsuits and fines, including a $700 million settlement. This motivates organizations to invest in security and development teams to integrate security-related tasks into their software development processes.

Securing software is an increasingly difficult task for software practitioners [23]. Various tools and methods have been introduced to automate and support security-related tasks to reduce the burden of securing software. In addition to our primary research efforts, we conducted a comprehensive review of existing literature focusing on security challenges encountered in Agile software development methodologies and the recommended security practices to address these challenges. From this extensive collection of security practices, we selected eight activities that we deemed to have the potential to provide the highest value in enhancing the security posture of Agile software development processes. These security activities are presented in Table 1. We incorporate these state-of-the-art security activities in our evaluation to understand software engineers' perceptions of their impact on Agile development processes.

2.2 Agile

Software engineering processes are useful for organizing development activities within the Software Development Life-cycle (SDLC). A typical SDLC begins with analyzing requirements for a project and goes through the system's design, implementation, testing, maintenance, and deployment. Agile is a framework for SE processes that focuses on iterative and incremental development, continuous delivery, and customer satisfaction through collaboration [38]. The primary goal of Agile methodologies is to be responsive to change to incorporate flexibility and avoid increased development costs and unnecessary rework typical in traditional SE processes [2].

Agile, known for being lightweight and adaptable [27], was officially introduced with the "Manifesto for Agile Software Development"[3] released in 2001–outlining the values of Agile software development [14]. However, these principles can positively or negatively impact software security in software development, as summarized below:

Individuals and interactions over processes and tools: Agile emphasizes communication and interactions between individuals over processes and tools. These interactions between software practitioners can lead to increased knowledge about security vulnerabilities and methods to mitigate them. For instance, software engineers primarily discover security-related tools through recommendations from coworkers [57], and peer code reviews are effective for detecting possible vulnerabilities [16]. However, many security activities necessitate adopting tools and libraries–which are challenging for developers to use to fix vulnerabilities [13, 51].

Working software over comprehensive documentation: Agile processes prioritize functional software over extensive documentation. Incorporating security activities into working software can increase confidence in the system's security. Yet, many security activities increase documentation–primarily related to potential vulnerabilities. For example, security tools generate reports outlining detected issues, increasing documentation for development teams [48]. Moreover, minimizing this documentation can lead to inadequate knowledge of the system's security. However, the incomprehensible output generated by security tools is one of the main barriers to adoption in DevOps workflows [42] and development teams [51].

Customer collaboration over contract negotiation: Engaging with clients is emphasized over formal contract agreements in Agile development processes. Involving customers in security-related discussions early in the development process has also been shown to address security needs and concerns [50] effectively. However, clients may not possess the knowledge needed to implement security practices or understand risks in software. Thus, security activities can involve formal contracts for beneficial practices, such as hiring external security experts and teams to validate software security. For instance, red teams are frequently employed by large software companies such as Microsoft to emulate hackers and try to compromise production software and systems [52].

Responding to change over following a plan: Adaptability is one of the main benefits of Agile development [2], allowing development

[3] https://agilemanifesto.org/

Table 1: Security Activities for Agile Software Development

Studies	Security Activity	Definition
[8, 12, 30, 47]	*Addressing security in early iterations with requirements and testing*	This security activity emphasizes the importance of development teams addressing security issues and concerns early in the project before deploying the software.
[18, 30, 47]	*Stating security requirements that are expected in the production software*	This requires incorporating security expectations in project requirements when describing the responsibilities and behavior of the software.
[8, 18, 19]	*Adding a security specialist to your team*	Security specialists, such as a Security Master, are members of a development team that focus on security aspects of the project to address concerns and ensure the security of the system.
[5, 35, 41, 53]	*Additional points or weights to issues with an impact on security*	This activity involves increasing the weights, such as story points in an Agile development environment, of issues that will have a higher impact the security of the product to prioritize security-related tasks and encourage more secure development and testing.
[5, 15, 30]	*Iterative and incremental vulnerability and penetration testing*	This security activity suggests incorporating recurring security scanning, such as Dynamic Application Security Testing (DAST), to test for security flaws in the working software automatically.
[5, 8, 15]	*Iterative and incremental security static analysis*	Similar to DAST, Static Application Security Testing (SAST) involves using security-related static analysis tools to detect potential security vulnerabilities by scanning the source code.
[6, 15, 20]	*Iterative and incremental risk analysis, countermeasure graphs*	This security activity consists of using tools to monitor networks, applications, and infrastructure and perform risk analysis to identify vulnerabilities. These tools can evaluate the system's security and suggest methods to prevent attacks.
[3, 15, 30]	*Automatic testing*	This security activity involves incorporating secure coding practices, such as vulnerabilities analysis and risk assessment, into the deployment pipeline for software projects. This allows security checks to be automatically triggered with code changes and issues to be addressed before the software is deployed to users.

teams to react to modified requirements instead of strictly adhering to a specification. This can benefit software security by encouraging development teams to respond quickly to security incidents to reduce the time between identifying security issues and deploying a solution [56]. Integrating security activities into development processes often involves following a defined plan. For example, prior work suggests traditional security practices, such as Microsoft's Security Development Lifecycle [24], fail to scale in Agile conditions because of high costs and time constraints due to iterative development on projects [5]. Further, incorporating security standards, such as the Federal Information Processing Standard (FIPS), requires documented system security planning to discuss security-related concepts, such as how information is inventoried [55].

The success of incorporating these values with security practices depends on the implementation of individual development teams. Thus, this work aims to explore the perceptions and experiences of software practitioners incorporating various security activities into Agile development processes.

3 RELATED WORK

Prior work has investigated software development methods in Agile development and concluded that the myth of Agile methodologies decreasing software security is false and security activities are increasing in popularity [44]. Jabangwe and colleagues [25] also found similar results in a literature review investigating research exploring ways to integrate security practices into Agile development processes. While these studies are based on theoretical evidence, this work aims to analyze how specific security activities are perceived by software engineers with experience working on Agile development teams.

Beznosov et al. [9] categorize security assurance methods into four groups based on their adaptability to Agile methodology. Their results saw that almost half of these security practices clash directly with Agile development, primarily due to their dependence on heavy documentation. Similarly, Bartsch and colleagues[7] conducted a literature review to investigate security challenges in Agile development, concluding clients and developers should explicitly discuss non-functional security requirements earlier in the project and risk awareness incorporated into Agile team retrospective meetings. This work builds on these studies by surveying software practitioners to analyze their experiences and perspectives on integrating security activities into Agile development processes.

Hammad et al. [22] explored the neglect of risk management in Agile software development. By conducting an online survey, the authors revealed that while risk management strategies are used, they are often applied non-systemically, with project deadlines and varying requirements being the most commonly faced risks by practitioners. We add to this work by shifting the focus from risk management practices to incorporating security activities in Agile. This extension is significant because it addresses the unique challenges posed by security activities, which often involve more documentation and tools, potentially conflicting with Agile's lightweight and responsive nature. Our research focuses on integrating security practices within Agile and understanding practitioners' perceptions of them, providing insights into software development practices beyond risk management. Another study by Agrawal et al. [1] examined the current limitations and advantages of Agile software development, conducting an online survey among Agile practitioners highlighting issues such as lack of upfront planning, budget constraint, insufficient documentation, and predictability challenges. We study the impact of security activities on Agile software development rather than the limitations of Agile methods. We aim to examine the integration of security practices into Agile, acknowledging the potential conflicts between Agile principles and the more traditional and sequential nature of security activities.

Assal et al. [3] investigated real-life software security practices during various SDLC stages through developers' interviews. With the popularity of Agile, our research aims to narrow the focus from the broader SDLCs to specifically Agile software development. Similar to study [3], we aim to explore real-life software security practices but specifically in Agile contexts. Additionally, we investigate how these security activities impact the Agile development process. The study [3] discusses discrepancies between real-life security practices and best practices for securing software identified in the literature. Our research aims to extend this discussion by examining how software practitioners perceive the effectiveness of security practices within Agile and their willingness to incorporate them.

Ayalew et al. [4] evaluated and compared existing high-profile waterfall security-engineering processes with Agile methodologies, identifying the need for specific security practices tailored for Agile projects and emphasizing the importance of balancing security efforts with integration costs. Our research complements these findings by focusing on the impact of existing security activities within Agile software development. While this study suggests existing waterfall security-engineering processes may not align well with Agile projects, we investigate how software practitioners perceive the effectiveness of security practices deemed compatible with Agile. By exploring these aspects, our study addresses the practical implications of securing Agile development, offering insights beyond identifying and evaluating security activities presented by Ayalew and colleagues [4].

4 METHODOLOGY

4.1 Data Collection

Using QuestionPro,[4] we created and distributed an online survey. The survey was conducted to help us answer our research questions investigating the impact of security activities on various aspects of Agile development. It contained nine questions with multiple parts. The Institutional Review Board (IRB) provided approval for both the survey and its protocol before the commencement of data collection.

4.2 Participant Recruitment

We aimed to recruit diverse participants to ensure a comprehensive perspective. Our recruitment strategy involved reaching out to potential participants through multiple channels. We used personalized invites and posts on LinkedIn[5] to promote our survey, inviting professionals in the software industry. Through Slack [6], we sent the survey to our network of contacts actively engaged in IT teams within different organizations. We also reached out to Virginia Tech graduate students with relevant technical work experience developing software in a professional industry setting.

4.3 Survey Structure

Our survey collected demographic information and background details on participants' experiences with Agile development and security practices, specifically asking respondents to provide insight into the security activities adopted by their development teams and to provide feedback on the recommended practices in Table 1. As this research focuses specifically on security activities in Agile software development, we discarded participants without Agile experience.

We asked participants to provide their perspectives on adopting security practices into Agile, their take on these security practices and how these practices affect them, their team and their overall organization. To answer RQ1, we inquired about their perceived effectiveness of specific security practices to increase the overall software security individually. Subsequently, we also asked the participants how willing they were to include each state-of-the-art security practice in their Agile software development process. To answer RQ2, we asked how the inclusion of these security practices affected the sprint velocity. Lastly, we asked about their confidence level in the overall security of the software product to answer RQ3. Also, it is important to note that some questions were kept as optional in our research survey, and consequently, only 41% ($n = 14$) of the respondents chose to provide answers to them. These questions delved into various aspects of security practices within Agile processes, seeking insights into their impact on teams, productivity, software products, organizations, and individuals' day-to-day activities. Despite their optional nature, the responses to these questions offer valuable perspectives on integrating security practices into Agile development. Kindly access the survey questions and their corresponding format through the hyperlink [7].

4.4 Data Analysis

Our survey contained closed-ended, Likert scale, and open-ended questions asking respondents about their opinion on software security, incorporating security practices in Agile, and their impact. We used a mixed methods approach to analyze the 5-point Likert scale questions. The percentage value for an option in a question is calculated by using this formula:

$$\left(\frac{n}{N}\right) \times 100\%$$

where (n) is the number of responses for that option and (N) is the total number of responses to the question. We also used a one-way (Analysis of Variance) ANOVA test to analyze the variance of responses across the activity. We consider a result to be statistically significant if $p < 0.05$. Qualitative analysis of the free response data was done using an iterative-inductive thematic open coding analysis. We worked in a team of two, with each individual performing the open coding analysis on the responses separately; then, the results were compared and discussed to reach a conclusion agreed on by both researchers. For example, one survey question asked about the different security practices employed in the respondents' current workplace. We reviewed the responses to this question using the open coding method and highlighted parts of sentences that indicate one or more adopted security practices. In one of the responses for the above question, *"Dual-authentication, least privilege so only certain users could access certain stage environments just as testing."*, "Dual-authentication" indicated the *Multi-Factor Authentication* practice. Another response, *"Only certain users could*

[4]https://www.questionpro.com/
[5]https://www.linkedin.com/
[6]https://www.slack.com
[7]https://doi.org/10.6084/m9.figshare.25655838

access certain stage environments" indicated the *Identity Access Management* security practice. In this manner, as shown Table 2, each response was parsed and categorized into one or more security practices. By utilizing open coding, we could discern recurrent themes and trends in the data and, as a result, gain a more profound comprehension of the respondents' opinions and experiences incorporating security activities into Agile development processes. The analysis revealed various security practices employed in Agile software development for the above survey question.

4.5 Participants

Our survey was completed by 34 individuals averaging approximately seven years of software engineering experience. Most participants were software engineers with years of experience working in the software industry. Of 34 participants, 67% (n = 23) were working professionals with an average of eight years of technical work experience, and 33% (n = 11) were university students pursuing graduate studies. Participants who identified as industry professionals reported working for organizations such as Acquia, Flexcar, GlobalLogic, Decisions.com, Cvent, Lutron Electronics, Palo Alto Networks, and Microsoft in various roles such as software engineer, infrastructure engineer, senior product manager, technical consultant, systems architect, and database administrator. The remaining participants were graduate students with an average of two years of work experience in the software engineering domain. Most participants viewed security as extremely (n = 25, 76%) or very important (n = 7, 21%) for their software team. Details about our survey respondents are presented in Table 3.

5 RESULTS

5.1 RQ1: Security Activities in Agile, Perceived Effectiveness & Willingness to Adopt

5.1.1 Adoption. While 97% of participants (n = 33) reported using Agile software development methodologies, only 72% (n = 23) reported having security-related activities in their Agile processes. When excluding participants who reported being students, we found 77% (n = 27) of software professionals adopt security activities in their development process. The participants were working in Agile development and not specifically in Agile security. Example security activities adopted by Agile teams include security training, security scanning & monitoring systems, security static analysis tools, code reviews, integrating standards such as Open Worldwide Application Security Project (OWASP) [39], Identity Access Management, Multi-Factor Authentication, and zero trust policies [49], and separate security teams.

5.1.2 Perception. Then, we asked a follow-up question to understand participants' perspectives regarding the security practices employed within their respective teams. The responses encompassed a range of sentiments, which indicated diverse opinions. A majority of the respondents expressed these practices as "good" (n = 8), "informative" (n = 2), "necessary" (n = 2) and something that needs to be complied with (n = 1). On the other hand, some participants expressed that these activities were time-consuming (n = 1) and disliked (n = 1). The respondents also acknowledged the need for improvement (n = 4), emphasizing the importance of enhancing security measures. Overall, the data suggests that while certain individuals recognize the value of security practices, there is room for progress and optimization within software development teams.

5.1.3 Agile Security. We also asked respondents how much they agree with the statement: "Software developed through Agile methods is relatively less secure when compared to software developed through sequential SDLC processes, like Waterfall". As shown in Figure1, the responses from the survey show a diverse range of opinions. Most (43.7%) of the participants disagreed with the statement, indicating that they perceive Agile software development processes as relatively secure. Furthermore, a considerable proportion (31%) remained neutral, neither fully agreeing nor disagreeing. This suggests that many respondents may be uncertain about the security implications of Agile practices. On the other hand, a minority of respondents (15.6%) strongly agreed with the statement, while an even smaller percentage (9.38%) agreed. These responses indicate no consensus among the majority, emphasizing the need for further exploration into the impact of security practices on Agile software development processes.

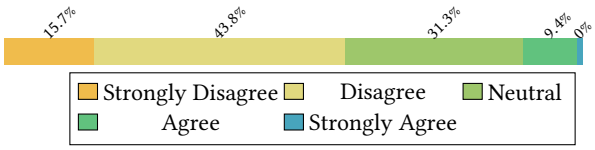

Figure 1: Participants' response on, "Agile software is less secure compared to the ones developed using Waterfall".

5.1.4 Effectiveness of software security practices and willingness to include them in Agile. The survey also listed the eight state-of-the-art security practices mentioned in Table 1. We asked participants how effective these activities would be in increasing the security and robustness of software if their team included them in the Agile software development process. A total of 30 participants responded to the question. The collected data, displayed as a heat map in Table 4, revealed notable scores assigned to each practice, indicating the degree of perceived effectiveness. "Iterative vulnerability testing" garnered the second highest score of 60% (Extremely effective), suggesting its efficacy in identifying and mitigating potential vulnerabilities. Similarly, "Automatic testing" obtained the highest score of 66.67% (Extremely effective), underscoring the potential of automation to bolster security throughout the SDLC. Other practices, such as "Iterative security static analysis" and assigning "Additional weight based on security impact," also demonstrated strong effectiveness ratings. However, certain practices received slightly lower scores, indicating potential areas for improvement. For instance, "Iterative risk analysis, countermeasure graphs" obtained the lowest scores. However, an ANOVA test showed no significant difference in participants' perceived effectiveness across the aforementioned security activities ($F = 1.806, p = 0.09035$). These findings still provide valuable insights into the perceived effectiveness of various security practices and motivate the need to incorporate automation in security activities to bolster software security and robustness in an Agile setting.

Table 2: Open coding examples for "What are the security practices used in your Agile process?"

Participant	Response	Categories Identified
P1	"Security training which takes place each quarter. And quiz based on that training."	Security Training
P3	"Use of continuous Integration, Add multiple checks for static analysis and code scan Proper IAM policy."	Continuous Integration (CI), Static Application Security Testing (SAST), Security scanning, Identity Access Management (IAM)
P4	"Compliance best practices and checklist"	Compliance best practices
P8	"Differential Access to resources"	Identity Access Management (IAM)
P10	"We follow Owasp standards. Each enhancement goes through the tests according to OWASP standards."	OWASP standards
P12	"In my previous company, there was a separate team that worked on the security aspects of the product, like login, user management, permissions, etc. To prevent unauthorised access to certain parts of software. There were security expectations in each user story, and those aspects used to get tested by QA before deploying the code."	Separate Security team, Identity Access Management (IAM), Agile Stories, Separate security team, Agile stories
P18	"Regular security scans using commercial tools, as well as requested scans of servers and applications by the IT security office. Monitoring of known security outlets for zero-day exploits."	Continuous Integration (CI), Security scanning, Automation tools, Monitoring
P19	"Periodic reviews"	Periodic reviews
P27	"Before any story is picked up, there's a Security Review of the entire epic with the Security Team."	Agile stories, Separate security team, Periodic reviews
P30	"Dual-authentication, least privilege so only certain users could access certain stage environments just as testing."	Multi-factor Authentication (MFA), Identity Access Management (IAM)
P31	"Code review, multiple release plan, security checks by software development engineering team and security team."	Code reviews, Separate security team, Continuous Integration (CI)
P32	"There are security experts to ensure secure practices and regular security testing and analysis is performed."	Security expert, Security practice insurance, Continuous Integration (CI)
P33	"Secure Clouds and systems - MFA - Including a round of security test with every PR or Features - Zero Trust Policies."	Multi-factor Authentication (MFA), Zero trust policy, Secure cloud services & systems, Code reviews

Table 3: Survey Participants

Participant	Role	Industry Exp. (years)	Agile?	Security?	Participant	Role	Industry Exp. (years)	Agile?	Security?
P1	Associate Software Engineer	1	Yes	Yes	P18	Chief Test Monkey	41	Yes	Yes
P2	Software Engineer	2.2	Yes	Yes	P19	Cloud Engineer	7	Yes	Yes
P3	Software Engineer	2	Yes	Yes	P20	Systems Architect	8	Yes	No
P4	Engineering Manager	11	Yes	Yes	P21	Department Head	23	Yes	Yes
P5	Software Engineer	6	Yes	Yes	P22	Associate Director of Systems Development	11	Yes	Yes
P6	Student	0	Yes	Yes	P23	Director, DBAA	22	Yes	Yes
P7	Quality Engineer	1.5	Yes	No	P24	Software Developer	20	No	No
P8	Graduate Teaching Assistant	1	Yes	Yes	P25	Software Engineering Co-Op	1	Yes	Yes
P9	Student	4	Yes	No	P26	Senior Product Manager	10	Yes	No
P10	Consultant	15	Yes	Yes	P27	Software Engineer	2.5	Yes	Yes
P11	Senior Software Engineer	5	Yes	Yes	P28	Student	3	Yes	No
P12	Student	3	Yes	Yes	P29	Student	1	Yes	No
P13	Student	3	Yes	No	P30	Technical Consultant	1	Yes	Yes
P14	Automation Test Engineer	4.2	Yes	Yes	P31	Software Engineer	2	Yes	Yes
P15	Graduate Student	2.8	Yes	No	P32	Security Co-Op	0-1	Yes	Yes
P16	Student	0	Yes	No	P33	Senior Staff Machine Learning Engineer	4	Yes	Yes
P17	Senior Software Engineer	6	Yes	No	P34	Infrastructure Engineer	1.5	Yes	Yes

Table 4: Security Activities and Practitioners' Perceived Effectiveness

Security Activity	Not at all	Slightly	Moderately	Very	Extremely
Addressing security in early iterations with requirements and testing	0%	0%	13.33%	73.33%	13.33%
Stating security requirements that are expected in the production software	0%	3.33%	20%	46.67%	30%
Adding a security specialist to your team	0%	6.67%	20%	40%	33.33%
Additional points or weights to issues with an impact on security	0%	0%	20%	46.67%	33.33%
Iterative and incremental vulnerability and penetration testing	0%	0%	10%	30%	60%
Iterative and incremental security static analysis	0%	3.33%	6.67%	53.33%	36.67%
Iterative and incremental risk analysis, countermeasure graphs	0%	6.67%	30%	43.33%	20%
Automatic testing	0%	3.33%	0%	30%	66.67%

Table 5: Security Activities and Practitioners' Willingness to Include Them in Agile Processes

Security Activity	Not at all	Slightly	Moderately	Very	Extremely
Addressing security in early iterations with requirements and testing	0%	0%	29.03%	45.16%	25.81%
Stating security requirements that are expected in the production software	0%	0%	29.03%	41.94%	29.03%
Adding a security specialist to your team	0%	6.45%	19.35%	48.39%	25.81%
Additional points or weights to issues with an impact on security	0%	0%	12.9%	38.71%	48.39%
Iterative and incremental vulnerability and penetration testing	0%	0%	16.13%	19.35%	64.52%
Iterative and incremental security static analysis	0%	0%	3.23%	45.16%	51.61%
Iterative and incremental risk analysis, countermeasure graphs	3.23%	0%	22.58%	48.39%	25.81%
Automatic testing	0%	0%	3.23%	29.03%	67.74%

Lastly, we inquired about the willingness of the respondents to include specific security practices into their Agile process for the same security practices listed in Table 1. In total, 31 participants responded to this question. As shown in Table 5, several practices received notable scores–indicating a high willingness to include them. For example, both iterative vulnerability testing and iterative security static analysis obtained strong scores. The automatic testing activity received the highest score of 67.74% (Extremely willing), emphasizing a strong willingness to incorporate this particular practice. Other practices, such as additional weights based on security impact and stating security requirements, also received favorable scores. However, certain practices, including addressing security in early iterations and iterative risk analysis, obtained the lowest scores. Using an ANOVA test, we found no statistically significant difference in participants' willingness to adopt certain practices ($F = 1.3191, p = 0.2457$). However, understanding these inclinations can assist organizations in effectively promoting and implementing security measures in Agile software development teams.

5.2 RQ2: Impact on Productivity

5.2.1 Team Velocity. Most of the participants who responded to the optional question about their opinion on the impacts of security practices on team velocity ($n = 14$) reported that adopting security practices did not affect their team's overall output. Some of this was dependent on specific team policies, such as one participant who noted the activities were planned *"before the sprint starts, [and] the developers go in the spring knowing what to expect"* (P26). Some participants noted minor changes to their productivity, adding it affected their productivity *"10-20%"* (P17) or one to two days (P16, P28). However, one participant responded that incorporating new initiatives for security engineering *"will always affect the sprint velocity drastically"* leading to a slower rollout of features, but concluded *"such changes are fruitful"* (P14).

5.2.2 Day-to-day Activities. Participants also noted that security activities have less effect on their day-to-day work. Many respondents ($n = 16$) noted that they had little to no effect on daily development tasks. Specific activities mentioned as not interrupting development processes include integrating tools to do *"periodic security checking"* (P20) and having an external team. However, the most popular disruption reported ($n = 2$) was increased time in the development process. For instance, P27 mentioned security activities lead to *"more time spent authenticating to access different environments and projects"*. Thus, we found that integrating security activities into Agile development processes does not have a major impact on the sprint velocity of Agile development teams.

5.3 RQ3: Impact of Software Activities

5.3.1 Software Products. We were also interested in exploring how incorporating security activities into Agile practices impacted the software practitioners. We asked participants how the involvement of their reported security practices affected their software products. Many participants ($n = 10$) reported that the involvement of security practices increased overall security. This suggests that implementing robust security measures positively influenced software products, making them more secure and resilient to potential threats. Additionally, a few participants mentioned that the involvement of security practices contributed to increased customer trust ($n = 1$), highlighting the importance of instilling confidence in end-users. Other reported impacts include increased confidence among

team members ($n = 1$), adherence to higher standards ($n = 1$), and a decrease in the number of bugs ($n = 1$).

However, these activities may have some negative effects as well. For instance, one respondent mentioned that these activities *"extended delivery date since a lot of code was often stuck waiting for approval"* (P30). P19 also mentioned that their teams adopted security practices *"rarely"* impacted the product's security. Our findings demonstrate the positive influence of security practices on software products in terms of their security posture and broader aspects such as customer trust, team morale, and overall product quality. However, it becomes evident that incorporating these activities requires careful planning to manage and avoid effectively delaying software feature deployment.

5.3.2 Organization. Next, we asked how these security activities affected the organization. While a few participants noted positive effects, such as the likely inclusion of more security practices ($n = 1$), an improved overall culture ($n = 1$), building company reputation ($n = 1$), and increased customer confidence ($n = 1$), the majority of respondents indicated either no effect ($n = 4$) or a minimal impact ($n = 4$) on their organizations. These findings suggest that while some organizations experience tangible benefits, others may not perceive substantial changes resulting from implementing these practices. Further analysis is necessary to understand the specific factors influencing these variations and to identify strategies for optimizing the integration of security practices within organizational contexts. However, it can also be said that the benefits of these activities are not explicitly visible since there is no way of knowing how many different security threats the organization faces and how many of them were prevented due to the inclusion of security practices.

5.3.3 Confidence. We studied how participants' confidence in the security of their software product was influenced by the security practices adopted by their Agile teams. The results, as shown in Figure 2, Participants generated various responses, providing insights into the perceived impact of security practices on confidence levels. a majority, 50% of the respondents reported feeling "fairly confident" in the security of the software they build, 25% responded "somewhat confident", and 25% indicated being "completely confident". These findings indicate a positive influence of security practices in instilling trust and assurance among the software practitioners involved and underscore the significance of integrating robust security measures into software development processes. However, more work is needed to increase the confidence of software engineers.

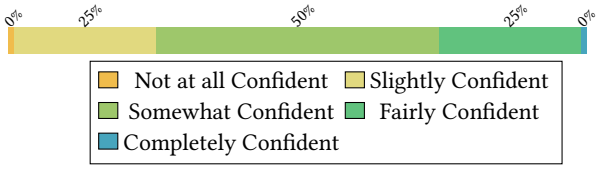

Figure 2: Participants' confidence in software security was influenced by the adoption of security practices by their Agile teams

6 DISCUSSION

Our study provides valuable insights into integrating security activities within Agile development, shedding light on software practitioners' perceptions and the impact on various aspects of the Agile development process. The results show that software practitioners positively perceive incorporating security activities into Agile despite the potential for conflicts between these domains. This alignment with security practices underscores the growing awareness of the importance of software security. Integration of security activities also had minimal impact on software practitioners' productivity and generally increased the security of software products. While some participants noted increased time and occasional delays in feature deployment related to security activities, these concerns were outweighed by the perceived benefits of enhanced security. This suggests that, with careful planning and efficient implementation, Agile teams can successfully integrate security practices without compromising productivity.

However, it is important to address the concerns raised by some participants regarding the need for improved security activities. Some participants were only "somewhat" or "fairly" confident that their team's adopted security activities improved the protection of their software. In light of their feedback, it becomes evident that there is room for enhancing the effectiveness of security practices within Agile development. To minimize the impact of software security practices on wasted time and improve confidence in system security, we recommend a two-pronged approach to address this issue: *increasing automation* and *improving feedback*.

6.1 Increase Automation

Our results highlight the positive perception and willingness to adopt Agile processes involving automation and security tools. For example, while we noticed no significant difference in the eight security activities in our survey, we found more automated approaches (i.e., automatic testing) had better perception and higher willingness to adopt among participants than more manual practices (i.e., adding a security specialist). This is consistent with prior work, which shows automated penetration testing techniques are more efficient than manual approaches [54]. However, Edwards and colleagues suggest automated security techniques can be harmful due to various social and technical issues, such as inaccurate output, not meeting stakeholder requirements, and information overload for users [17].

Therefore, there is an opportunity to streamline security activities through automation further. Our results suggest that teams should consider implementing automated testing, vulnerability assessments, penetration testing, and static security analysis tools to enhance the security of the software- as these security practices had higher perceived effectiveness and willingness to adopt metrics. To that end, automation can help support software engineers utilizing security activities in iterative development processes. For example, prior work has explored continuous security testing or integrating automated security tools into CI/CD processes for repositories [43]. Similarly, DevOps, a development methodology that incorporates Development (Dev) and Operations (Ops) aspects of software development, has been modified as DevSecOps to incorporate concepts and tools related to security into DevOps pipelines [31]. These

practices are also commonly adopted with Agile and iterative development processes because they focus on rapidly delivering software to users [38]. To that end, incorporating security activities in these processes can further automate and streamline the protection of software projects.

Our results demonstrate notable scores for recommended security practices involving automation and tools—particularly automatic testing, vulnerability & penetration testing, and security static analysis—indicating higher perceived effectiveness and willingness to adopt Agile processes.

6.2 Improve Feedback

However, blindly automating security tasks is not a valid solution. For instance, prior work suggests users are unlikely to trust and use automated development tools that fail to meet personal, interactive, and technical values–such as incorrect or incomprehensible results [28]. We also observed this in our study, with some participants reporting feeling the adopted security activities among their teams had little to no impact on the security of products and lacked confidence in these processes. Moreover, too much information without context can be detrimental to developers using security tools [17]. Security activities should also incorporate useful and actionable feedback to users to improve this. Prior work suggests poor feedback hinders security tool adoption [42, 51]. Providing clear and actionable feedback through interventions such as collaborations between security experts and developers [13] can improve the impact of automated security tools and increase software engineers' confidence in the security of their systems.

For example, systems could incorporate additional information instead of just reporting vulnerabilities to users. Prior work suggests developers lack knowledge of the consequences of generated security output [51]. To mitigate this, researchers have explored using vignettes and brief stories to illustrate concepts to increase understanding and behavior related to software security [11]. To that end, automated tools can also provide insight into the consequences of potential vulnerabilities. Developers also lack knowledge on how to address security issues reported by security-related tools. To mitigate this, Bhandari et al. inspected open-source repositories to collect security vulnerabilities and their corresponding fixes [10]. Research has shown that automated program repair can improve general debugging tasks [33], and we speculate automated tools suggesting security fixes can be useful in providing specific feedback to software practitioners and enhancing the security of software products.

7 LIMITATIONS AND FUTURE WORK

A limitation of this work is that participants' responses may not generalize to all Agile software practitioners. To mitigate this, we recruited participants from different backgrounds and experience levels to understand the impact of security practices on Agile development. However, this pool included a substantial representation of graduate students ($n = 11$), from which a majority ($n = 9$) were from Virginia Tech, which could introduce a bias in the findings. Future work can further investigate software practitioners' perspectives by surveying a wider audience. Additionally, our survey relies on participants' memory and estimations to report how security practices impact team velocity, which can be inaccurate. Further research is needed to investigate the impact of security activities on software practitioner productivity. This could be explored by analyzing Github[8] repositories to uncover practices used and measure productivity metrics.

A potential avenue for future research could involve conducting a comprehensive use-case study that examines integrating specific security practices into an Agile development process within a software engineering team. This study would focus on soliciting feedback from team members regarding their firsthand experiences during the incorporation. By obtaining insights directly from the individuals involved, valuable information can be gathered to assess the practical implications, challenges, and benefits of adopting security practices in an Agile environment. This research approach would provide an opportunity to gain a deeper understanding of the team dynamics, the impact on workflow, and the overall effectiveness of the security practice within the Agile development process. Such a study would contribute to the existing body of knowledge by offering real-world insights into the concepts discussed in this study.

Finally, future work can explore developing novel security tools and methods to avoid conflicts with Agile software development. For example, reducing the amount of documentation by providing concise and actionable feedback to software engineers to prevent and fix vulnerabilities. To achieve this, such a tool could use the power of Large Language Models (LLM) to understand and summarize lengthy security reports generated by security tools often loathed by software practitioners. Another tool could be developed to scan the static code in the CI/CD pipeline to look for vulnerabilities, thereby providing DevOps engineers with a quick and easy way to identify and fix security issues. Additionally, increasing automation in security-related practices and tools can benefit software engineers by saving time and incorporating security analysis into their Agile development workflows, such as CI/CD or DevOps processes.

8 CONCLUSION

Securing software systems is challenging, especially when security activities conflict with software development processes like Agile. We surveyed software practitioners to understand their experiences incorporating security activities into Agile development processes and their perception of state-of-the-art approaches. Our results show that software practitioners find security activities useful, and their integration did not negatively impact team productivity. Our work provides practical implications to improve current and future security activities with increased automation and improved feedback to increase confidence in security practices for Agile software development processes, ultimately improving the security of software products.

REFERENCES

[1] Ashish Agrawal, Mohd Aurangzeb Atiq, and LS Maurya. 2016. A current study on the limitations of agile methods in industry using secure google forms. *Procedia Computer Science* 78 (2016), 291–297.

[8] https://github.com/

[2] Samar Al-Saqqa, Samer Sawalha, and Hiba AbdelNabi. 2020. Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies* 14, 11 (2020).

[3] Hala Assal and Sonia Chiasson. 2018. Security in the software development lifecycle. In *Fourteenth symposium on usable privacy and security (SOUPS 2018)*. 281–296.

[4] Tigist Ayalew, Tigist Kidane, and Bengt Carlsson. 2013. Identification and evaluation of security activities in agile projects. In *Secure IT Systems: 18th Nordic Conference, NordSec 2013, Ilulissat, Greenland, October 18-21, 2013, Proceedings 18*. Springer, 139–153.

[5] Dejan Baca and Bengt Carlsson. 2011. Agile development with security engineering activities. In *Proceedings of the 2011 international conference on software and systems process*. 149–158.

[6] Dejan Baca and Kai Petersen. 2013. Countermeasure graphs for software security risk assessment: An action research. *Journal of Systems and Software* 86, 9 (2013), 2411–2428.

[7] Steffen Bartsch. 2011. Practitioners' perspectives on security in agile development. In *2011 Sixth International Conference on Availability, Reliability and Security*. IEEE, 479–484.

[8] Lotfi ben Othmane, Pelin Angin, Harold Weffers, and Bharat Bhargava. 2014. Extending the agile development process to develop acceptably secure software. *IEEE Transactions on dependable and secure computing* 11, 6 (2014), 497–509.

[9] Konstantin Beznosov and Philippe Kruchten. 2004. Towards agile security assurance. In *Proceedings of the 2004 workshop on New security paradigms*. 47–54.

[10] Guru Bhandari, Amara Naseer, and Leon Moonen. 2021. CVEfixes: automated collection of vulnerabilities and their fixes from open-source software. In *Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering*. 30–39.

[11] John Blythe. 2013. Cyber security in the workplace: Understanding and promoting behaviour change. *Proceedings of CHItaly 2013 Doctoral Consortium* 1065 (2013), 92–101.

[12] Brian Chess and Gary McGraw. 2004. Static analysis for security. *IEEE security & privacy* 2, 6 (2004), 76–79.

[13] Partha Das Chowdhury, Joseph Hallett, Nikhil Patnaik, Mohammad Tahaei, and Awais Rashid. 2021. Developers are neither enemies nor users: they are collaborators. In *2021 IEEE Secure Development Conference (SecDev)*. IEEE, 47–55.

[14] Kieran Conboy. 2009. Agility from first principles: Reconstructing the concept of agility in information systems development. *Information systems research* 20, 3 (2009), 329–354.

[15] Sébastien Dupont, Guillaume Ginis, Mirko Malacario, Claudio Porretti, Nicolò Maunero, Christophe Ponsard, and Philippe Massonet. 2021. Incremental common criteria certification processes using DeVSecOps practices. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 12–23.

[16] Anne Edmundson, Brian Holtkamp, Emanuel Rivera, Matthew Finifter, Adrian Mettler, and David Wagner. 2013. An empirical study on the effectiveness of security code review. In *Engineering Secure Software and Systems: 5th International Symposium, ESSoS 2013, Paris, France, February 27-March 1, 2013. Proceedings 5*. Springer, 197–212.

[17] W Keith Edwards, Erika Shehan Poole, and Jennifer Stoll. 2008. Security automation considered harmful?. In *Proceedings of the 2007 Workshop on New Security Paradigms*. 33–42.

[18] Adila Firdaus, Imran Ghani, and Seung Ryul Jeong. 2014. Secure feature driven development (SFDD) model for secure software development. *Procedia-Social and Behavioral Sciences* 129 (2014), 546–553.

[19] Adila Firdaus, Imran Ghani, and Nor Izzaty Mohd Yasin. 2013. Developing secure websites using feature driven development (FDD): a case study. *Journal of Clean Energy Technologies* 1, 4 (2013), 322–326.

[20] Xiaocheng Ge, Richard F Paige, Fiona AC Polack, Howard Chivers, and Phillip J Brooke. 2006. Agile development of secure web applications. In *Proceedings of the 6th international conference on Web engineering*. 305–312.

[21] Karen Mercedes Goertzel, Theodore Winograd, Holly L McKinley, Lyndon Oh, Michael Colon, Thomas McGibbon, Elaine Fedchak, and Robert Vienneau. 2007. Software security assurance: a state-of-art report (sar). *DTIC Document* (2007).

[22] Muhammad Hammad, Irum Inayat, and Maryam Zahid. 2019. Risk management in agile software development: A survey. In *2019 international conference on frontiers of information technology (fit)*. IEEE, 162–1624.

[23] Matthias Hölzl, Axel Rauschmayer, and Martin Wirsing. 2008. Engineering of software-intensive systems: State of the art and research challenges. *Software-Intensive Systems and New Computing Paradigms: Challenges and Visions* (2008), 1–44.

[24] Michael Howard and Steve Lipner. 2006. *The security development lifecycle*. Vol. 8. Microsoft Press Redmond.

[25] Ronald Jabangwe, Kati Kuusinen, Klaus R Riisom, Martin S Hubel, Hasan M Alradhi, and Niels Bonde Nielsen. 2021. Challenges and Solutions for Addressing Software Security in Agile Software Development: A Literature Review and Rigor and Relevance Assessment. *Research Anthology on Recent Trends, Tools, and Implications of Computer Programming* (2021), 1875–1888.

[26] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. 2016. What is DevOps? A systematic mapping study on definitions and practices. In *Proceedings of the scientific workshop proceedings of XP2016*. 1–11.

[27] Kiran Jammalamadaka and V Rama Krishna. 2013. Agile software development and challenges. *International Journal of Research in Engineering and Technology* 2, 08 (2013), 125–129.

[28] Brittany Johnson, Christian Bird, Denae Ford, Nicole Forsgren, and Thomas Zimmermann. 2023. Make Your Tools Sparkle with Trust: The PICSE Framework for Trust in Software Tools. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 409–419.

[29] Hossein Keramati and Seyed-Hassan Mirian-Hosseinabadi. 2008. Integrating software development security activities with agile methodologies. In *2008 IEEE/ACS International Conference on Computer Systems and Applications*. IEEE, 749–754.

[30] Vidar Kongsli. 2006. Towards agile security in web applications. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. 805–808.

[31] Anna Koskinen. 2019. DevSecOps: building security into the core of DevOps. (2019).

[32] McKenzie L Kuhn. 2018. 147 million social security numbers for sale: Developing data protection legislation after mass cybersecurity breaches. *Iowa L. Rev.* 104 (2018), 417.

[33] Claire Le Goues, Michael Dewey-Vogt, Stephanie Forrest, and Westley Weimer. 2012. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 3–13.

[34] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. 2016. Relationship of devops to agile, lean and continuous deployment: A multivocal literature review study. In *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17*. Springer, 399–415.

[35] Patrik Maier, Zhendong Ma, and Roderick Bloem. 2017. Towards a secure scrum process for agile web application development. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*. 1–8.

[36] Runfeng Mao, He Zhang, Qiming Dai, Huang Huang, Guoping Rong, Haifeng Shen, Lianping Chen, and Kaixiang Lu. 2020. Preliminary findings about devsecops from grey literature. In *2020 IEEE 20th international conference on software quality, reliability and security (QRS)*. IEEE, 450–457.

[37] Grigori Melnik and Frank Maurer. 2005. A cross-program investigation of students' perceptions of agile methods. In *Proceedings of the 27th international Conference on Software Engineering*. 481–488.

[38] ABM Moniruzzaman and Dr Syed Akhter Hossain. 2013. Comparative Study on Agile software development methodologies. *arXiv preprint arXiv:1307.3356* (2013).

[39] OWASP Foundation. [n. d.]. OWASP Application Security Verification Standard. https://github.com/OWASP/ASVS.

[40] Kai Petersen and Claes Wohlin. 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study. *Empirical Software Engineering* 15 (2010), 654–693.

[41] Christoph Pohl and Hans-Joachim Hof. 2015. Secure scrum: Development of secure software with scrum. *arXiv preprint arXiv:1507.02992* (2015).

[42] Roshan Namal Rajapakse, Mansooreh Zahedi, and Muhammad Ali Babar. 2021. An Empirical Analysis of Practitioners' Perspectives on Security Tool Integration into DevOps. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12.

[43] Thorsten Rangnau, Remco v Buijtenen, Frank Fransen, and Fatih Turkmen. 2020. 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)*. IEEE, 145–154.

[44] Kalle Rindell, Sami Hyrynsalmi, and Ville Leppänen. 2017. Busting a myth: Review of agile security engineering methods. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*. 1–10.

[45] Pilar Rodríguez, Mika Mäntylä, Markku Oivo, Lucy Ellen Lwakatare, Pertti Seppänen, and Pasi Kuvaja. 2019. Advances in using agile and lean processes for software development. In *Advances in Computers*. Vol. 113. Elsevier, 135–224.

[46] Sabbir M Saleh, Syed Maruful Huq, and M Ashikur Rahman. 2019. Comparative study within Scrum, Kanban, XP focused on their practices. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE, 1–6.

[47] P Salini and S Kanmani. 2012. Survey and analysis on security requirements engineering. *Computers & Electrical Engineering* 38, 6 (2012), 1785–1797.

[48] Phillip Schneider, Markus Voggenreiter, Abdullah Gulraiz, and Florian Matthes. 2022. Semantic Similarity-Based Clustering of Findings From Security Testing Tools. *arXiv preprint arXiv:2211.11057* (2022).

[49] Malcolm Shore, Sherali Zeadally, and Astha Keshariya. 2021. Zero trust: the what, how, why, and when. *Computer* 54, 11 (2021), 26–35.

[50] Guttorm Sindre and Andreas L Opdahl. 2005. Eliciting security requirements with misuse cases. *Requirements engineering* 10 (2005), 34–44.

[51] Justin Smith, Lisa Nguyen Do, and Emerson Murphy-Hill. 2020. Why can't johnny fix vulnerabilities: A usability evaluation of static analysis tools for security. In *Proceedings of the Sixteenth Symposium on Usable Privacy and Security*.

[52] Justin Smith, Christopher Theisen, and Titus Barik. 2020. A case study of software security red teams at Microsoft. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–10.

[53] Sonia, Archana Singhal, and Hema Banati. 2014. FISA-XP: An agile-based integration of security activities with extreme programming. *ACM SIGSOFT Software Engineering Notes* 39, 3 (2014), 1–14.

[54] Yaroslav Stefinko, Andrian Piskozub, and Roman Banakh. 2016. Manual and automated penetration testing. Benefits and drawbacks. Modern tendency. In *2016 13th international conference on modern problems of radio engineering, telecommunications and computer science (TCSET)*. IEEE, 488–491.

[55] Marianne Swanson, Joan Hash, and Pauline Bowen. 2006. Guide for Developing Security Plans for Federal Information Systems. National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-18r1.pdf.

[56] Laurie Williams, Gary McGraw, and Sammy Migues. 2018. Engineering security vulnerability prevention, detection, and response. *IEEE Software* 35, 5 (2018), 76–80.

[57] Shundan Xiao, Jim Witschey, and Emerson Murphy-Hill. 2014. Social influences on secure development tool adoption: why security tools spread. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. 1095–1106.

[58] Shahid Kamal Tipu Ziauddin and Shahrukh Zia. 2012. An effort estimation model for agile software development. *Advances in computer science and its applications (ACSA)* 2, 1 (2012), 314–324.