# Quantifying Developers' Adoption of Security Tools

Jim Witschey[†•], Olga Zielinska[†◦], Allaire Welk[†◦],
Emerson Murphy-Hill[†•], Chris Mayhorn[†◦], and Thomas Zimmermann[‡]
[†]North Carolina State University and [‡]Microsoft Research
[•]Department of Computer Science and [◦]Department of Psychology
{jim_witschey,oazielin,akwelk,ermurph3,chris_mayhorn}@ncsu.edu, tzimmer@microsoft.com

## ABSTRACT

Security tools could help developers find critical vulnerabilities, yet such tools remain underused. We surveyed developers from 14 companies and 5 mailing lists about their reasons for using and not using security tools. The resulting thirty-nine predictors of security tool use provide both expected and unexpected insights. As we expected, developers who perceive security to be important are more likely to use security tools than those who do not. But that was not the strongest predictor of security tool use, it was instead developers' ability to observe their peers using security tools.

## Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering

## Keywords

security, tools, adoption, developers

## 1. INTRODUCTION

In 2014, a buffer overflow vulnerability was discovered in the X server, part of the widely-used X Window System [48]. Developers found that "a BDF font file containing a longer than expected string could overflow the buffer on the stack". This vulnerability was found with the static analysis tool cppcheck [12], which warned developers of a call to `scanf` without field width limits. This `scanf` bug could "lead to an unprivileged user acquiring root privileges in some systems", according to the announcement from X.org [1].

This vulnerability was introduced in 1991. We wonder why it took developers a full 23 years to use tools to find and repair this vulnerability. Even prior to cppcheck's introduction in 2009, several security tools could have found this error. Nonetheless, it seems developers did not use them when working with X.

We define *security tools* as analysis tools that help developers find and fix vulnerabilities. Some security tools, such as Klokwork [21] and Coverity [6], are designed specifically to find vulnerabilities. Our definition also encompasses general-purpose analysis tools for finding programming errors, such as static analysis tools like FindBugs [18] and dynamic analysis tools like some Valgrind plugins [31]. Tools like cppcheck and FindBugs are free to use, yet as demonstrated by the X vulnerability, the mere existence and availability of these tools does not make software more secure. Developers must use them, or will leave many vulnerabilities in code when they could be quickly found and fixed.

We aim to help practitioners find ways to increase adoption of security tools so developers find more vulnerabilities early. The use of security tools is an important part of secure software development methodologies, such as the Microsoft Secure Development Lifecycle [19] and McGraw's recommendations for secure development [25]. Thus, our work focuses on security tools because increasing their adoption could have great impact on developers and software users, whose data and computing resources would be more secure. We also focus on security tools because, unlike most other software development tools, they are a *preventative innovation*; this class of innovations is especially challenging to adopt [35] because the payoff for adopting them tends to be a non-event, that is, something bad *not happening*.

The main contribution of this paper is a quantification of the relative importance of factors that predict security tool adoption. We collected quantitative data on the factors' relative importance with an online survey of software developers working in 14 companies, as well as developers on 5 mailing lists. Surprisingly, we found developers' perceptions of the importance of security may not be the most important factor in their adoption of security tools. Social and cultural factors, like the perceived prestige of security tool users and the frequency of interaction with security experts, may be a more important influence. Our results can help practitioners choose areas of focus when attempting to promote security tool adoption, and help researchers prioritize areas of study for understanding security tool adoption.

## 2. RELATED WORK

### 2.1 Adoption

The theory for our work is drawn from *diffusion of innovations* (DOI) research [34]. One reason we chose DOI is because it has proven versatile in its long history. Rogers originally developed the theory to explain the adoption of new breeds of corn among farmers, but it has also been used to analyze adoption of such innovations as dance moves and personal digital workstations.

DOI is a rich field of adoption study that accounts for many aspects of adoption. For instance, DOI can account for temporal aspects of the diffusion of new technologies, such as the patterns in which innovations spread through populations, and the processes by which individuals make decisions about trying and adopting new innovations [34]. In our approach, we focus on factors that foster or inhibit adoption, so we can recommend particular practices that promote security tool adoption.

We also chose DOI so we could analyze social aspects of tool adoption. Our previous work investigated how developers learn about new tools, and found developers can learn about tools from their peers [30]. Where some of our other previous work identifies usability concerns and technical factors in tool use [20], this research investigates social factors as well. We aim to create a more comprehensive understanding of security tool adoption.

Our research in security tool adoption is not the first to use DOI as a theoretical foundation. For instance, Meyerovich and Rabkin [26] applied DOI to programming language adoption. They found social factors and socially-driven factors, like the availability of free and open-source libraries, affected developers' decisions when choosing to adopt or not adopt a programming language. DOI theory also helped Singer develop persuasive interventions to increase the use of Git among student developers [38]. Our research is novel in that we are the first to apply DOI to security tool adoption.

## 2.2 Quantifying Influences on Adoption

Other researchers have quantified influences on adoption through the use of surveys. In Meyerovich and Rabkin's previously-mentioned work [26], they surveyed open-source developers about what factors influence their choices when choosing a programming language for a new project. The survey presented here is largely inspired by surveys outside computer science research. For example, Tan and Teo [42] used a survey to develop a regression model describing the relationship between intent to adopt online banking and various variables such as experience with the Internet and perceived risk. Baaren used a survey study [3] to quantify the importance of factors in the Technology Acceptance Model [13] in HDTV adoption. We apply similar techniques to quantify security tool adoption.

## 2.3 Security Tool Adoption

In previous qualitative work, we built on Diffusion of Innovations theory to qualitatively interview 42 professional developers [46, 45]. Based on those interviews, we organized adoption factors into four high-level factor groups:

- **Innovation** factors describe how the properties of the innovations themselves – the security tools – affect developer adoption.
- **Social System** factors describe how the company or community in which developers write software can affect their adoption of security tools.
- **Communication Channel** factors describe how different ways of communicating about security tools, such as through advertisements, social media, or face-to-face communication, affect developers' adoption decisions.
- **Potential Adopter** factors describe how qualities of the individual developers themselves affect their adoption decisions.

Although we derived factors in these groups in our interviews, the results were not very actionable. For example, should a manager who wants to improve tool adoption focus on promoting a more usable security tool (a property of the innovation) or should they focus on implementing company standards regarding security (a property of the social system)? Our aim in the present study is to provide a justification for policy-makers and toolsmiths to focus their efforts, by quantifying adoption factors' relative importance. To our knowledge, the present paper is the first to study security tool adoption quantitatively.

# 3. METHODOLOGY

As in other adoption research [3, 42], we conducted a survey[1]. We first discuss the design and development of the survey instrument (Section 3.1). We then discuss and compare two iterations of the survey with different participant selection strategies (Sections 3.2 and 3.3). We close this section with a description of the data analysis (Section 3.4) and respondent characterization (Section 3.5).

## 3.1 Instrument Development

We began the development of our survey by adapting an existing survey instrument, originally developed by Moore and Benbasat [28]. Their survey concerned personal digital workstations; we altered it to concern security tools instead. We added questions to quantify the findings from our interviews and to help collaborating managers in industry evaluate the state of security awareness in their companies. For instance, since our interviewees indicated that developers who believe insecure software would cause problems for customers are more likely to use security tools, we added questions about these beliefs. We refined the survey based on piloting and feedback from colleagues, industry partners, and industry developers.

The initial survey consisted of 74 Likert-scale questions about factors that influenced respondents' security tool adoption decisions, 4 questions about respondents' experience and role in development, and 14 free-response questions to elicit factors not present in the survey and general comments about adoption. For the Likert scale we used Strongly Agree, Agree, Neither Agree nor Disagree, Disagree, Strongly Disagree, and included a "Don't Know" option. Each survey began with a page defining security and security tools in the same way as for our interview study [45, 46].

We measured security tool usage by asking respondents what we call the 'usage question':

*Which of the following statements describes you best?*

- *I usually use security tools when I develop software, or security tools automatically analyze the code I develop when I check in or build my code.*
- *I use security tools only occasionally or when I am performing specific tasks, like looking for vulnerabilities.*
- *I never or almost never use security tools.*

We chose the 3 points in a way that ensured each scale point had a concrete meaning [24], avoiding potentially ambiguous scale points such as 'fairly often' and 'often'.

[1]An empty survey can be found at `http://go.ncsu.edu/SecurityToolAdoptionSurvey`. To obtain as credible survey responses as possible, we assured respondents that only summary data would be publically disclosed; thus, raw survey data is not available.

**Table 1: Distributions of Security Tool Use**

| | Frequent | Occasional | Never |
|---|---|---|---|
| Iteration 1 | 16 | 48 | 130 |
| Iteration 2 | 46 | 13 | 2 |

## 3.2 First Survey Iteration

We distributed the survey through a convenience sample of our contacts in software development companies. Such non-probabilistic sampling reduces our ability to draw generalizable conclusions about the population of developers as a whole [17], but is reasonable when no practical way to randomly sample from the entire population exists.

We distributed the survey in one of two ways, depending on privacy concerns in the companies surveyed. In the first way, for nine companies, we distributed the survey directly to respondents after obtaining their email addresses from managers. We asked these managers to randomly select a set of developers at their company, including testers and other managers, then inform the developers that they would be contacted for the survey. We then emailed developers with a link to the online survey. During the following week, we emailed developers who had not participated to remind them about the survey. In the second way, for five companies, we were unable to obtain developers' email addresses due to privacy concerns. In these companies, we asked our contact in the company to again select developers, then send a form email to these developers containing a link to the survey, as well as a reminder email in the week following.

To incentivize participation, we offered a drawing for two $100 Amazon gift cards. The survey was open to each group of developers for one to two weeks. Of the 257 developers to whom we directly distributed the survey – those in the nine companies where we could obtain developers' emails – 105 completed it. This is a 40.8% completion rate, which is at the high end of typical response rates for surveys of software developers [39]. The rest of the respondents were contacted about the survey by our industry contacts; 14 completed it. In total, 119 completed the first iteration of the survey. However, because incomplete surveys contained some usable data, where possible we analyzed data from such surveys.

To mitigate the negative effects that the survey's length would have on participation and completion, we used a split survey design [33]. All respondents were asked about their knowledge of secure development practices and their experience and role in software development. We grouped the remaining questions by topic, then grouped these topics into four pages with roughly equal numbers of questions. Each respondent was presented with two of the four pages. We used the Qualtrics online survey platform, which randomly assigned pages of the survey to respondents in a way that ensured approximately equal numbers of respondents were presented with each page. As a result, each page of the survey was presented to the same number of respondents, plus or minus four respondents.

The distribution of responses to this question for the first iteration of the survey are shown in the first row of Table 1 (Iteration 1). What stands out about this distribution is how many non-users of security tools responded, compared to how few users of security tools did. While the perspective of non-users is important, to meaningfully draw conclusions about the differences between users and non-users, we aimed to collect more responses from security tool users.

## 3.3 Second Survey Iteration

We ran a second iteration of our survey, compensating for the lack of tool users in the first sample, by using a different sampling method. Additionally, we reduced the size of the survey so that every respondent could answer every question.

### 3.3.1 Sampling Method

To sample developers who were more likely to use security tools, we distributed this survey to 5 mailing lists for users of different security tools. The first 50 participants from each mailing list received a $15 Amazon gift card for participating. There was evidence that many surveys were completed programatically, so we cleaned the data by removing all participants who took less than 5 minutes to complete the survey. We chose 5 minutes because there was a clear discontinuity in the distribution between participants who took less than 5 minutes and those who took more than 5 minutes. This removed 313 responses and left 61. The distribution of responses to the question about security tool use for this iteration are shown in the second row of Table 1, which shows that this sample of respondents complements the first sample in terms of frequency of tool use.

### 3.3.2 Survey Size Reduction

Another disadvantage of the first iteration of the survey was that, due to our split survey design, most adoption factor questions were answered by only about half of respondents. This reduces statistical power [10]. For this iteration, we compensated by reducing the number of questions so that every participant could answer every question.

We shortened the survey by including only the questions that significantly correlated with usage in the first iteration. We analyzed 70 of the questions, which asked about factors that encourage or discourage adoption, by finding Spearman's rank correlation between responses to the questions and respondents' self-reported security tool use frequency. When we computed Spearman's correlation using SAS statistical software, we treated the usage question as an ordinal response with three levels.

Using R [43], our initial analysis found 41 out of 70 questions correlated significantly ($\alpha = .05$) with respondents' self-reported security tool use. When conducting so many independent statistical tests, there is a high probability of false discovery. We accounted for this chance of false discovery using a Benjamini-Hochberg adjustment [4]. On the basis of this adjustment, correlations with $p$-values above .0276 were treated as insignificant. At this level, responses to 39 of 70 questions correlated significantly with respondents' self-reported security tool use. We used these 39 questions on the second iteration of the survey.

## 3.4 Data Analysis

We analyzed the results by combining the responses of both survey iterations, analyzing just the questions that the two iterations had in common. This has two advantages. First, combining results enables increased statistical power, compared to analyzing each iteration separately. Second, combining results compensates for the usage skew (fewer security tool users in the first iteration and more in the second) that would make it difficult to detect effects when analyz-

ing the iterations separately. From a sampling standpoint, combining results is reasonable in two aspects. First, both samples are drawn from the same population of interest: software developers as a whole. Second, our survey should capture any differences between the two populations to the extent that our prior interviews rigorously exposed all the factors that contribute to tool adoption. Nonetheless, we cannot rule out the possibility of such differences. Like all empirical study designs, ours entail tradeoffs, and we judge that the benefits are worth the potential risks. We discuss these tradeoffs further in Section 5.

#### 3.4.1 Individual Logistic Regression Models

The aim of our first analysis is to determine which factors predict security tool usage. We first coded security tool adoption dichotomously, by converting the three-point scale used in the usage question by mapping the 'never' response to 'no adoption' and 'occasionally' and 'frequently' to 'adoption'. For each question, we performed a logistic regression analysis using R [43] to determine the extent to which each factor predicted the adoption of security tools. We again used a Benjamini-Hochberg correction to control the false discovery rate; the adjusted $p$-value for significance was .03. Additionally, we calculated odds-ratios to estimate effect size [41] using SPSS [2].

In the first iteration, four questions concerned developers' trust of different sources of information about security tools. These were qualitatively different from the other questions, and are not meant to predict tool use. Thus, we compare them to one another as described in Section 4.4.

For both individual and combined regression models, we assume that if a respondent did not know the answer to a question, their lack of knowledge did not affect the respondent's adoption decisions. This assumption was supported by Fisher-Exact value tests, which found no significant differences in security tool adoption between the responses with missing and the responses without missing values in the first survey iteration. Thus, we treated all "Don't Know" responses to Likert-scale questions as though they were missing at random (MAR). Treating such responses as MAR is considered a reasonable treatment of "Don't Know" responses, where models replacing them may not help characterize respondents any better [36].

#### 3.4.2 Combined Logistic Regression Model

One limitation of presenting a series of single-variable models is that their effects might be cross-correlated, which reduces our ability to isolate independent phenomena. To show that the factors in this study have independent effects, we built a combined logistic regression model using all 39 factors in R [43]. The combined model also included the role and experience of participants. To reduce the number of factors, we used stepwise regression, which deletes variables that improve the model the most by being deleted (according to the Akaike Information Criterion [37]) and repeats this process until no further improvement is possible.

### 3.5 Respondent Characteristics

Here we characterize the self-reported background of respondents across both iterations. About 73% of respondents were developers, 13% testers, 10% managers, and 4% other. About 2% had less than 1 year of professional software development experience, 10% 1-2 years, 16% 3-5 years, 28% 6-10 years, 27% 11-20 years, and 17% more than 20 years. The common software domains participants worked in were web, mobile, desktop, enterprise, client, and tiered applications; software as a service and cloud; analytics and statistics; developer tools and application lifecycle management; computer aided design; graphics; ecommerce; and databases.

## 4. RESULTS & IMPLICATIONS

**Interpreting the Results Table:** Table 2 summarizes our results. Each row of the table describes one statement for which survey respondents rated their agreement, as well as the results of our analysis for each question. For example, 130 respondents answered [S8], with a Cox and Snell regression $r^2$ coefficient of 0.24. Our regression model for [S8] showed a significant relationship ($p < .001$) between participants' levels of agreement with the statement and their adoption. Finally, the last column *Exp(B)* indicates an odds ratio of 3.18. If the value of an odds ratio is greater than 1, this means that respondents are more likely to adopt security tools with increases in the Likert scores, i.e., when they agree with a statement more. For example, for [S8], the odd ratio is 3.18, which means that for every one point Likert scale increase (more agreement, e.g., Neutral to Agree, or Agree to Strongly Agree), the odds that participants adopt security tools increase by a factor of 3.18. For two points of increase, e.g, from Neutral to Strongly Agree, the odds would increase by a factor of $3.18 * 3.18 = 10.1124$. Conversely, if the value is less than 1, individuals are less likely to adopt security tools. For every one point Likert scale increase, the odds decrease by a factor of Exp(B). For example, for [S39] with Exp(B) = .57, participants are less likely to adopt security tools; for every one Likert scale unit increase the odds drop by 1.75 times ($\frac{1}{.57}$). And in the case of [S39], an odds-ratio less than one is expected, given the statement's framing. In Table 2, statements are labeled S1 to S39; the order and number is based on highest odds ratio ([S1]) to lowest odds ratio ([S39]). The results show security tool usage is a significant predictor for all 39 statements.

To give an overview of the strength of the effects for each factor group, we use small bar charts. In these charts, the number of bars represents the number of statements that significantly predict tool adoption, the height of each bar represents an odds ratio, and the bars are ordered by descending height. For example, the header of Section 4.2.2 shows  , which indicates of the four Likert scale statements that significantly predicted tool adoption, one was relatively strong (4.37×), one medium (2.97×), and two weak (1.84× and 1.78×).

Questions with large values for $N$ ([S23], [S26], [S30], [S33], [S34]) were presented to all respondents of each iteration. Additional variation in $N$ is due to non-response for some questions and "Don't Know" responses.

**Discussing Implications:** Throughout, we will also discuss the implications our findings have for those who want to spread security tools in organizations. We do this here, rather than discussing implications in their own section, to make clear the connections between our findings and the course of action they imply. To differentiate the findings of our survey from their implications, the latter will be in paragraphs beginning with **Implications**, in bold. As we mentioned in Section 3.4, while our methodology precludes strong claims of generalizability, in our implications we take some liberties in assuming that our results apply in wider

contexts beyond the participants surveyed for this study.

As is typical for surveys, statistically significant relations alone do not imply causality. For some relationships, however, we have evidence of causation from our interview studies. Where appropriate, we will discuss possible causal links and their implications. We do so to provide questions for other researchers to answer in the future, and to show these results' potential impact.

## 4.1 Summary of Results

Next we discuss the factors in terms of the four factor groups: the innovation factors, the social system factors, the communication channel factors, and the potential adopter factors. We contextualize survey respondents' answers in qualitative data provided by our prior interviewees.

## 4.2 Social System Factors

### 4.2.1 *Security Concern* ■■■ *& Awareness* ■■■■■

*Security concern* is the perceived importance of security in a social system. In our interviews, higher security concern led to the adoption of tools. For example, some developers reported that the software they developed was only used by authenticated, trusted users, so they thought security was not important. As a result, these developers did not use security tools [46].

In the survey, security concern predicted security tool use, but the three *security concern* statements ([S36],[S37],[S38]) had among the weakest effects of all. For instance, a 1-point increase in agreement with the statement "I work on software for which security is very important" [S37] only accounted for a 1.49× increased odds of using security tools.

Likewise, the five *security awareness* statements ([S23], [S26], [S30], [S33], and [S34],) predicted tool use, but also had smaller effects than most other factors. Combined with our interview study, where some developers did use security tools because of this attitude [46], these results imply a causal link between security concern and security tool use, though a relatively weak one.

**Implications:** While importance of security is influential, our results on the whole indicate that the perceived importance of security does not influence adoption as much as other factors. Our results comport with Xie and colleagues' findings that security concerns influence secure coding practices [47], but our results suggest it is not the most important factor. Surprisingly, based on our results' relative effect sizes, making developers more security-conscious may not be as effective as other interventions for promoting tool use.

### 4.2.2 *Policies & Standards* ■■■

In our interviews, the *policies and standards* relating to security tools affected their adoption as well. For instance, all interviewees required to use security tools did so [46].

Different policies and standards in the companies that developers worked in also predicted tool use. Organizations' application of explicit standards [S31] for software security predicted security tool use. Adoption is also predicted by the application of secure development processes, like those from OWASP [32] and the Microsoft Secure Development Lifecycle [27] [S23].

The second strongest effect in the survey was that adopters were more likely to report their superiors expect them to use security tools [S2]. Though not as strong an effect, respondents who used security tools were more likely to say that their superiors reward them for writing secure software [S29].

**Implications:** Sociological research has found that workers in security-critical fields often circumvent security policies. Even in hospitals, where the cost of mistakes can be fatal, workers often circumvent policies requiring the use of barcodes to identify patients and drugs [22].

Our findings, in contrast, indicate that developers who are required to use security tools actually do so. As mentioned in Section 4.3.1, this may be because developers generally believe security tools help them do their jobs better and faster. These findings suggest that organizations should firstly prescribe tool use as part of a secure development methodology and secondly explicitly reward developers for the security of their software.

While our findings show some respondents were rewarded for writing secure code, we do not know how they were rewarded or what rewards are most effective. Future research could help companies determine the best ways to reward developers and make them feel valued for their secure development efforts. Many companies, such as Mozilla [29], GitHub [16], Facebook [15], and Twitter [44], use bug bounties, or rewards for outsiders who discover vulnerabilities.

### 4.2.3 *Structure* ■■■

The *structure* of a company or other social system in which developers write software, with respect to how security experts and security auditing teams interact with other developers, affects adoption as well. For instance, we found in our interview study that developers who interact with security experts feel more personally responsible for security. Thus, we asked a number of questions about the structure of the environment in which developers worked – about how respondents interact with security and testing teams, for example. We also asked about respondents' perceptions of their responsibility for security, as Xie and colleagues [47] found these perceptions led some developers to not adopt security practices.

Respondents who used security tools were more likely to report they felt personally responsible for the security of the software they develop [S27], and that they interacted frequently with others in their organization who help improve the security of their software [S22]. Respondents were also more likely to say their peers thoroughly reviewed their software for security [S35]. While structural relationships were significant, the effects were weaker than most other factors'.

**Implications:** These results strengthen the findings from our interviews that interviewees who interacted frequently with security teams were more likely to use security tools, out of a greater sense of personal responsibility for security. The strongest structure effect in the survey was with the frequency with which developers interact with others who work in security. Thus, we suggest that policy should encourage security experts to conduct audits alongside other developers, rather than simply running analyses and sending reports. Security experts and other developers could also conduct pair programming sessions; pair programming can effectively transfer knowledge between programmers [9], and thus also might be used to transfer knowledge about security tools.

### 4.2.4 *Education & Training* ■■■

We found in our previous interviews that different companies provide different resources for *education and training*

**Table 2: Individual Regression Models Predicting Security Tool Use**

| ID | Statement | Factor Group | $N$ | $r^2$ | $p$ | $Exp(B)$ |
|---|---|---|---|---|---|---|
| S1 | I have seen what others do using security tools. | Observability | 141 | 0.35 | <.001 | 4.46 |
| S2 | My superiors expect me to use security tools. | Policies | 134 | 0.36 | <.001 | 4.37 |
| S3 | Using security tools {is/would be} cost-effective. | Advantages | 130 | 0.20 | <.001 | 3.95 |
| S4 | I actively seek out information about security tools. | Inquisitiveness | 146 | 0.29 | <.001 | 3.70 |
| S5 | Using security tools {make/would make} it easier to do my job. | Advantages | 136 | 0.25 | <.001 | 3.62 |
| S6 | It is easy for me to observe others using security tools in my organization. | Observability | 138 | 0.25 | <.001 | 3.36 |
| S7 | Using security tools {helps/would help} me do my work more quickly. | Advantages | 137 | 0.22 | <.001 | 3.21 |
| S8 | I know how I can satisfactorily try out various uses of security tools. | Trial Ease | 130 | 0.24 | <.001 | 3.18 |
| S9 | People in my organization who use security tools have more prestige than those who do not. | Culture | 128 | 0.17 | <.001 | 3.16 |
| S10 | The software I develop is analyzed by security tools when it is built or tested. | Policies | 132 | 0.27 | <.001 | 2.97 |
| S11 | I frequently learn about security tools from blogs and technical websites. | Exposure | 148 | 0.25 | <.001 | 2.84 |
| S12 | I was permitted to use security tools on a trial basis long enough to see what it could do. | Trial Ease | 116 | 0.17 | <.001 | 2.77 |
| S13 | My organization holds frequent trainings on security tools. | Education | 142 | 0.19 | <.001 | 2.65 |
| S14 | My organization holds frequent trainings on software security. | Education | 142 | 0.21 | <.001 | 2.64 |
| S15 | Using security tools {improves/would improve} my job performance. | Advantages | 135 | 0.17 | <.001 | 2.60 |
| S16 | Using security tools {is/would be} a good use of my time. | Advantages | 140 | 0.11 | <.001 | 2.51 |
| S17 | I frequently learn about security tools from other developers. | Exposure | 147 | 0.18 | <.001 | 2.44 |
| S18 | I frequently learn about security tools from managers in my organization. | Exposure | 147 | 0.19 | <.001 | 2.41 |
| S19 | Using security tools {improves/would improve} my image within my organization. | Culture | 129 | 0.09 | .001 | 2.23 |
| S20 | Given multiple security tools I can easily choose which to use for a given task. | Advantages | 132 | 0.12 | <.001 | 2.15 |
| S21 | Security tools are available to me to adequately try out. | Trial Ease | 118 | 0.10 | <.001 | 2.13 |
| S22 | I interact frequently with others in my organization who help improve the security of the software I develop. | Structure | 147 | 0.15 | <.001 | 2.12 |
| S23 | I apply secure development standards such as those from OWASP or the Microsoft Secure Development Lifecycle. | Security Awareness | 234 | 0.15 | <.001 | 2.12 |
| S24 | I learned about security tools in university courses. | Education | 149 | 0.15 | <.001 | 2.08 |
| S25 | Security tools present their analyses in understandable ways. | Complexity | 109 | 0.05 | .024 | 1.96 |
| S26 | I could explain software security design to a new developer on my project. | Security Awareness | 242 | 0.12 | <.001 | 1.93 |
| S27 | I am personally responsible for the security of the software I develop. | Structure | 144 | 0.10 | <.001 | 1.91 |
| S28 | The internal workings of security tools are complex. | Complexity | 110 | 0.05 | .025 | 1.88 |
| S29 | My superiors reward me for writing secure software. | Policies | 130 | 0.08 | .001 | 1.84 |
| S30 | I am aware of secure development standards such as those from OWASP or the Microsoft Secure Development Lifecycle. | Security Awareness | 241 | 0.12 | <.001 | 1.83 |
| S31 | In my organization there are explicit standards for the security of the software I develop. | Policies | 127 | 0.10 | <.001 | 1.78 |
| S32 | If the software I develop were insecure I would be embarrassed. | Culture | 139 | 0.05 | .008 | 1.68 |
| S33 | Security is best emphasized primarily in end-stage testing. | Security Awareness | 238 | 0.08 | <.001 | 1.67 |
| S34 | Adding security functionality is important to the developers I work with. | Security Awareness | 230 | 0.06 | <.001 | 1.61 |
| S35 | My peers thoroughly review the software I develop to ensure it is secure. | Structure | 144 | 0.07 | .002 | 1.61 |
| S36 | If the software I work on were insecure it would put important resources at risk. | Security Concern | 150 | 0.05 | .009 | 1.53 |
| S37 | I work on software for which security is very important. | Security Concern | 151 | 0.05 | .008 | 1.49 |
| S38 | If the software I work on were insecure it would cause problems for customers and users. | Security Concern | 150 | 0.04 | .021 | 1.46 |
| S39 | Security tools are not very visible in my organization. | Observability | 138 | 0.06 | .005 | 0.57 |

about security and security tools. Some companies hold occasional security seminars, while in other companies, knowledge of security is part of the interview process and is generally assumed of all developers [46]. We conducted these survey studies partly to study the effects these differences may have on security tool adoption.

Security tool adopters were more likely to report they were educated or had opportunities to continue their education in security; use was predicted by having learned about security tools in university courses [S24] and working in an organization that holds frequent trainings on security tools [S13] and on software security [S14]. Respondents who reported having access to organizational trainings were more likely to report using security tools (2.64× and 2.65×) than those taught about security tools in university courses (2.08×).

**Implications:** Hiring developers who have been trained in software security may be an effective means of spreading security tools in an organization. Even if they do not currently use the tool the organization wants to diffuse, they may be more likely to adopt the tool. These findings also suggest that organizations should hold trainings on not just security, but on security tools as well.

#### 4.2.5 Culture

The *culture* of a social system, with respect to security tool adoption, consists of the beliefs and social norms concerning security and security tool use in that social system. We found in our previous interview study that, in some companies, developers considered using new security tools "cool", while others had cultures, shaped by policies, that encouraged just using tools that were already approved for use in the company [46].

We asked several questions on our survey to determine the effects that such cultural attitudes had on adoption. Respondents who used security tools were more likely to report that the company they worked in had a culture that encouraged secure development and the use of security tools. They reported developers who use security tools were more prestigious than those who do not [S9], to relatively large effect (3.16×), and that using security tools improved their image within their organizations [S19]. Adopters were also more likely to report that making software secure is important to the developers they worked with [S34]. Adopters were more likely than non-adopters to report they would be embarrassed if software they developed were not secure [S32].

**Implications:** Naturally, affecting the culture of a social system is a hard problem. However, Singer's work [38] indicates that gamification and other means of externally influencing the prestige of using software development practices can increase their adoption. Building on that work, future research might apply social gamification to the security tool adoption problem.

### 4.3 Innovation Factors

#### 4.3.1 Relative Advantage

We asked several questions to determine how well properties of the security tool itself, or *relative advantage*, predicts tool adoption. If a developer thinks using a tool is superior to not using it, or to using a different tool, it has a high relative advantage. Relative advantage includes developers' perceptions of technical aspects like soundness and false positive rates. Another aspect of relative advantage is expense; prior interviewees thought that security tools were too expensive, in financial or time costs, to be worth using [45]. Such tools had a low perceived relative advantage.

In our surveys, adopters tended to believe that security tools helped respondents do their jobs faster [S7] and at a higher level of performance [S15]. Respondents who use security tools are more likely than non-adopters to believe that security tools are cost-effective [S3], were a good use of their time [S16], and feel confident choosing between different security tools [S20]. The relative advantage factors had relatively strong effects on adoption.

**Implications:** In our previous study, all interviewees who were required to use security tools did so [46]. One interpretation of those results is that developers only use security tools because of these requirements. Findings in managerial psychology indicate that fear of punishment for breaking workplace rules can influence employee behavior [14]. This could imply that developers only use security tools to avoid breaking workplace policies. Circumvention of policy is common in other workplaces, such as hospitals [22].

In contrast, our results indicate that developers have their own reasons for using security tools: in general, adopters think security tools have a positive impact on their work. In this sense, the workplace requirement to use security tools is different from other requirements in that developers think it actually helps them do better work.

#### 4.3.2 Observability

We also asked developers questions about the *observability* of security tools. According to diffusion of innovations (DOI) theory, a technology is more observable if others can easy to tell that an adopter is using it. For example, cell phones are more observable than desktop computers, since an adopter of a cell phone will use it in public, while an adopter of a desktop will typically only use it in private. DOI theory states that more observable innovations are more likely to be adopted [34].

Survey responses to three questions show a significant relationship between security tool use and aspects of their observability [S1,6,39]. In fact, the strongest of all the relationships we found was with the statement "I have seen what others do using security tools" [S1].

**Implications:** Toolsmiths and policymakers may be able to impact tool adoption by increasing observability. For instance, pair programming or developer-led demonstrations is one way to improve observability in an organization.

#### 4.3.3 Complexity

DOI theory states that innovations perceived to be *complex* are less likely to be adopted, than simpler ones. For instance, Black and colleagues found that less experienced people with computers were less likely to adopt online banking technologies, partly because of the perceived complexity of the system [7]. Our previous interviews suggested developers are equally effected: one developer reported they did not use a security tool because the tool's interface was so complex that it made performing simple tasks "a nightmare" [45].

The survey did confirm that complexity was a significant factor in tool adoption, but not particularly strongly. Adopters were more likely than non-adopters to report that security tools presented their analyses in understandable ways [S25]. In addition, adopters were more likely to believe that security tools are internally complex than non-adopters

were [S28]. Both factors' effects were weaker than those from most other factors.

**Implications:** To some degree, complexity is linked to the power of a security tool, yet toolsmiths may be able to increase adoption through reduced complexity by, for example, building tools that communicate with developers using familiar terminology.

#### 4.3.4 Trialability ■■■

In DOI theory, innovations have higher *trialability* if they are easier to use on a trial basis, without significant investment in the as-yet-untested technology. More trialable innovations are more likely to be adopted than less trialable ones. In our previous interview study, some developers reported not using security tools because it was not worth the time it took to install and configure a tool [45].

Adopters were more likely to say they knew how to satisfactorily try out security tools [S8]. Security tool adopters also were more likely to say they had access to security tools [S15] and were permitted to try security tools to adequately see what they do [S12].

**Implications:** These relationships suggest that companies trying to foster security tool adoption should make policies to make it easy for employees to try new tools. These findings also indicate that policies that prevent developers from easily trying new tools make them less likely to adopt them. Offering resources to show developers how to install and try out tools may encourage adoption by making developers feel they are able to evaluate their functionality. Toolsmiths may be able to increase adoption of their tools by reducing barriers to demonstrating and trying the functionality of security tools. For instance, future research could find ways for companies to host flexible analysis platforms to help developers, with minimal effort, try different security tools on their code.

### 4.4 Communication Channel Factors

#### 4.4.1 Trust

Developers *trust* different sources of information about security tools different amounts. In our previous interviews, participants trusted prominent developers they know of on the internet more than managers in their organization [46]. DOI theory indicates that individuals are more likely to adopt innovations they learn about from trusted parties.

On the first iteration of the survey (but not the second iteration), we asked four questions about channels participants trusted for information about security tools. Each question was in the form "If I learned about a security tool from {Source}, I would trust that information", where {Source} was "a manager in my organization", "another developer", "a blog or technical website", or "an advertisement, online or otherwise". A 85 respondents answered at least one trust question, of which 4 respondents did not answer between one and three trust questions. We ignored such missing data in this analysis.

The median trust in the manager and the developer was "Agree", while blog or technical website was "Neither Agree nor Disagree", and advertisement was "Disagree". Using a pairwise Wilcoxon signed-ranks test, these differences were statistically significant from one another at $p < .05$. We did not find a significant difference between trust for managers and developers. No trust differences emerged between respondents at different levels of tool usage (Kruskal-Wallis test, $p > .05$).

**Implications:** These findings indicate that advertisements are not effective ways to inform developers about security tools, as developers do not trust them. Developers' trust in peers helps explain previous findings that peer recommendation is a particularly effective way developers find out about new tools [30].

We were surprised to find that respondents trust managers as much as other developers. Our interviews and other previous work [30] indicate that developers do not trust their managers as much as other sources for information about tools. One explanation might be our sampling methodology; respondents who heard about the survey through their manager might be more inclined to answer it in a manager-friendly way. We expand on this threat further in Section 5.

#### 4.4.2 Exposure ■■■

The extent to which developers are *exposed* to security tools through different channels also affects their likelihood to adopt them. Our previous findings, in our interview studies on security tool adoption [46] and peer interaction [30] indicate that learning about tools from other developers is infrequent, but an effective vector for tool adoption.

Respondents who were exposed to tools were more likely to use them. We found that adopters were more likely than non-adopters to report learning about security tools from managers in their organization [S18] and technical blogs and websites [S11]. Additionally, respondents who used security tools were more likely to report they frequently learned about security tools from other developers [S17]. The effects of each exposure factor was larger than for most other factors.

**Implications:** We were surprised that the effect of learning about security tools through managers (2.41×) was about the same as that of learning security tools through peers (2.44×). This is surprising, given our previous findings that indicate developers do not trust their managers for information about tools [30]. However, our previous findings also indicate that developers learn about tools from their peers only infrequently; these findings may actually reflect how, in general, few developers learn about security tools from their peers. Our findings suggest that companies can use managers as agents of diffusion for security tools – developers who frequently hear about security tools from their managers are more likely to use security tools than those who do not, and developers may trust managers as much as other developers.

### 4.5 Potential Adopter Factors ■

Qualities of the *potential adopter*, the developer who must choose whether or not to adopt a security tool, affect developers' adoption decisions, according to our qualitative interviews. In particular, our interviews identified two relevant factors: developers' *experience*, in programming and software security, and their *inquisitiveness* about security tools. In our interviews, we found, for example, that all developers who did not seek out information about new security tools also did not use security tools [45].

We found security tool adopters were more likely than non-adopters to say they actively sought out information about security tools [S4], a relatively strong effect (3.70×), indicating that adopters are more inquisitive about security

**Table 3: A Combined Model of Security Tool Adoption**

| Variable | Factor Group | Statement | *B* | *p*-value | sig. | *Exp(B)* |
|---|---|---|---|---|---|---|
| (Intercept) | *NA* | Intercept | -11.865 | <.001 | *** | 0.000 |
| S1 | *Observability* | I have seen what others do using security tools. | 1.306 | .001 | *** | 3.691 |
| S5 | *Advantages* | Using security tools {make/would make} it easier to do my job. | 1.069 | .003 | ** | 2.913 |
| S10 | *Policies* | The software I develop is analyzed by security tools when it is built or tested. | 0.983 | .002 | ** | 2.673 |
| S4 | *Inquisitiveness* | I actively seek out information about security tools. | 0.863 | .003 | ** | 2.371 |
| S14 | *Education* | My organization holds frequent trainings on software security. | 0.643 | .021 | * | 1.902 |
| S18 | *Exposure* | I frequently learn about security tools from managers in my organization. | 0.619 | .030 | * | 1.857 |
| S12 | *Trial Ease* | I was permitted to use security tools on a trial basis long enough to see what it could do. | 0.563 | .104 | | 1.756 |
| S11 | *Exposure* | I frequently learn about security tools from blogs and technical websites. | 0.504 | .064 | | 1.655 |
| S27 | *Structure* | I am personally responsible for the security of the software I develop. | -0.427 | .087 | | 0.652 |
| S15 | *Advantages* | Using security tools {improves/would improve} my job performance. | -0.464 | .094 | | 0.629 |
| S29 | *Policies* | My superiors reward me for writing secure software. | -0.563 | .122 | | 0.570 |
| roleDeveloper | *Demographics* | Participant is a developer. | 1.487 | .124 | | 4.423 |
| roleManager | *Demographics* | Participant is a manager. | 0.923 | .405 | | 2.517 |
| roleTester | *Demographics* | Participant is a tester. | -0.154 | .890 | | 0.857 |
| S6 | *Observability* | It is easy for me to observe others using security tools in my organization. | -1.068 | .019 | * | 0.344 |

tools. Surprisingly, however, we did not find a statistically significant relationship between respondents' years of experience in software development and adoption.

**Implications:** Prior research on technology adoption indicates that experience and inquisitiveness have an impact on adoption. For instance, Chau and Hui found that novelty-seeking behavior and length of experience using computers were strongly related to early adoption of Windows 95 [8]. However, our results here indicate that inexperienced developers may be just as likely to use security tools as more experienced ones. Some of our prior results explain this; in an unrelated set of interviews, interviewees indicated that experienced developers can learn about software development tools from interns because interns have more time for experimenting with new tools [30]. These results suggest that organizations should consider employing inexperienced developers as knowledge sources about security tools.

## 4.6 A Combined Security Tool Adoption Model

We built a combined logistic regression model using all 39 factors. For this model, and only for this model, we used imputation [23] to deal with missing values. Missing values occurred when participants were not asked a question because of the split survey design, did not to respond to a question, or selected "Don't Know". We substituted missing values with the median Likert score of the completed responses [23] for that question.

After the stepwise regression, the factors for 12 questions and the role demographics survived; all other factors were eliminated. Among the surviving factors, only 7 were statically significant. The resulting model is shown in Table 3. Since the range of variables related to survey questions is the same (1 for strongly disagree, . . . , 5 strongly agree), the coefficients for the variables can be directly compared. A higher coefficient means a stronger influence.

As with the individual models, in the combined model, the strongest influence remains "I have seen what others do using security tools" [S1] with an odd ratio of 3.69×. The statistically significant factors with *positive* influence on security tool adoption come from different categories: Observability [S1], Advantages [S5], Policies [S10], Inquisitiveness [S4], Education [S14], and Exposure [S18]. The demographics did not matter in the combined model: experience was eliminated in the stepwise regression and role was not statistically significant.

One significant factor has *negative* influence (0.34×) on security tool adoption: "It is easy for me to observe others using security tools in my organization" [S6]. Yet, in isolation this factor had positive influence on adoption (3.36×, Table 2). One explanation is that there is an interaction between [S6] and [S1] ("I have seen what others do using security tools", 3.69× in the combined model). The effect of adoption is strongest for people who observed others using security tools and for whom it was not easy (high scores for [S1] and low scores for [S6]). If observing others was easy (high scores for both [S1] and [S6]), the effect on adoption is still positive (3.69*0.34=1.25×) but not as strong. In other words, having to go through obstacles to discover a security tool may increase the odds of adoption. Another explanation is that the negative effect is a result of data imputation. We tested this explanation by re-running our combined logistic regression model with only [S1] and [S6] and without imputation; we achieved this by only using data from respondents that got the page that contained the Observability factor group. In that model [S1] retained a significant positive influence, and [S6]'s negative effect was reduced and was no longer significant. This suggests that we cannot rule out imputation as the cause of [S6]'s negative effect in the combined, imputed model.

To further validate the model we used ten-fold cross validation. We split the dataset into 10 random folds. Each fold was then used to validate a logistic regression model based

on the factors in Table 3 constructed using the other nine training folds. The mean precision is 79.4% and mean recall is 73.4%, which suggests that the model can successfully predict security tool adoption.

As with the individual regression models, toolsmiths and policy makers can use the combined regression model to identify significant factors as potential avenues for facilitating tool adoption. Additionally, if such stakeholders want to address multiple factors, choosing several positive, significant ones from the combined model may be a way to influence adoption from different perspectives. For example, as we suggested in Section 4.2.3, the importance of "seeing what others do using security tools" suggests that encouraging practices like pair programming can spread knowledge of security tools. Finally, because the significant positive predictors of adoption were also significant in the individual models, the reader can have increased confidence that these trends are not simply artifacts of our analysis techniques.

# 5. THREATS TO VALIDITY

**Threats to Construct Validity:** We cannot guarantee that we have described and measured each factor exactly as it affects adoption. For instance, we wanted to measure respondents' inquisitiveness, but could not directly ask "how inquisitive are you?", since we cannot expect participants to accurately describe such subjective properties of themselves [5]. Instead, we asked questions such as [S4], asking if they sought out information about security tools, to measure the effects of their inquisitiveness. There are many ways to measure inquisitiveness, and for practical reasons, we measured just one of them.

One threat to construct validity is our combining of data from each iteration of the survey, as discussed in Section 3.4. If the samples are systematically different in a way that the second iteration of the survey does not capture, our results will not reflect those differences. For instance, the questions that existed on the first iteration of the survey but not the second may have been sensitive to such differences, but the data we gathered can neither confirm nor deny it. Another threat is the lower statistical power of analyzing questions that appeared on the first survey but not the second, because such questions necessarily had fewer total responses. Consequently, users and non-users of security tools may actually answer these questions differently.

To derive the combined model we use imputation to handle missing values. Different imputation strategies may lead to different models. Thus, the reader should exercise caution when interpreting the combined model.

**Threats to Internal Validity:** The present study is a retrospective survey, not an experimental study, and so our ability to make predictive claims is limited. As we noted earlier, many cases have evidence for causality, based on our interview study, but in other cases, especially for those questions designed based on Moore and Benbasat's work [28], we cannot. Fortunately, this paper provides the necessary foundation for conducting future experimental or action research-based studies [40]; we would recommend such studies focus on the largest significant effects that we have uncovered here.

Since we worked with managers to distribute the survey on the first iteration, we did not have control over the sampling methodology within companies. In addition, distribution with the help of managers could introduce bias. For instance, developers who know the manager who distributed the survey may tend to respond more positively to questions about management than if we were able to distribute the survey directly to developers. However, with the exception of the questions on trust, if we assume that both frequent and infrequent tool users are equally affected by this bias, then the bias would have little effect on our analyses. A similar response bias threat is that respondents may have had more experience or held a more positive attitude towards security tools than non-respondents.

As in most survey studies, the wording of the survey may have biased participants. To address this threat, we tried to word questions in a value-neutral way. Nonetheless, our survey did have section headings like "Inquisitiveness," which may have positive connotations and may have biased respondents towards agreeing with these statements. As with the potential bias introduced by distribution by management, if we assume equal bias on frequent and infrequent tool users, then this bias has little effect on our results.

**Threats to External Validity:** In our first iteration, most responses came from a small subset of the companies sampled; in most of the companies, we surveyed fewer than 10 developers. Thus, while this sample covers a variety of industry developers, respondents hail from a small number of companies. Similarly, respondents across both iterations may not be representative of all developers.

Some questions we asked during our first iteration did not appear significantly related to adoption based on our initial analysis. It may be that some relationship does indeed exist, but that our study lacked the statistical power to detect it.

As mentioned in Section 3, we omitted responses from participants who took less than 5 minutes to complete the survey in an effort to combat ballot stuffing [11]. However, we may have still included invalid responses from ballot stuffers who took more than 5 minutes, and also may have omitted data from real respondents who took the survey quickly.

# 6. CONCLUSION

Security tools are an important part of secure software development, but many developers do not use them, even when they believe security is important. This leaves software less secure than it could be. In this paper, we describe a quantification of factors that influence security tool adoption. Our results, the product of a long-term, mixed-methods research project, provide toolsmiths and policy-makers greater understanding of why tools are or are not adopted. Although toolsmiths and policy-makers have more control over some factors, our results provide a holistic lens through which these stakeholders can better understand adoption. We hope future applications of our findings will result in more developers using security tools, helping increase the security of the software we increasingly depend upon.

# Acknowledgments

Thanks to Jorge Aranda, Jim Clause, Denae Ford, Da Young Lee, Kevin Lubick, Leif Singer, Jon Stallings, the Developer Liberation Front, the Science of Security Lablet, and anonymous reviewers for their helpful feedback. This material is based upon work supported through the Science of Security by the National Security Agency and by the National Science Foundation under grant number 1318323. Special thanks to study participants.

## 7. REFERENCES


[1] Alan Coopersmith. X.org security advisory: Cve-2013-6462: Stack buffer overflow in parsing of bdf font files in libxfont. `http://lists.x.org/archives/xorg-announce/2014-January/002389.html`.

[2] J. L. Arbuckle. Ibm spss amos 22 userŠs guide. *Crawfordville, FL: Amos Development Corporation*, 2013.

[3] E. Baaren. Understanding technology adoption through individual and context characteristics: The case of HDTV. *Journal of Broadcasting And Electronic Media*, (February 2013):37–41, 2011.

[4] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, 1995.

[5] M. Bertrand and S. Mullainathan. Do people mean what they say? implications for subjective survey data. *The American Economic Review*, 91(2):pp. 67–72, 2001.

[6] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler. A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World. *Communications of the ACM*, 2010.

[7] N. Black, A. Lockett, H. Winklhofer, and C. Ennew. The adoption of internet financial services: a qualitative study. *International Journal of Retail & Distribution Management*, 29(8):390–398, 2001.

[8] P. Y. Chau and K. Lung Hui. Identifying early adopters of new IT products: A case of Windows 95. *Information & Management*, 33(5):225–230, May 1998.

[9] T. Chau, M. F. Chau T., and F. Maurer. Knowledge sharing in agile software teams. *Lecture Notes in Computer Science*, 3075:173–183, 2004.

[10] J. Cohen. *Statistical power analysis for the behavioral sciences*. Lawrence Erlbaum Associates, Inc, 1977.

[11] M. Couper, M. Traugott, and M. Lamias. Web Survey Design and Administration. *Public opinion quarterly*, 65:230–253, 2001.

[12] Daniel Marjamäki. cppcheck. `http://cppcheck.sourceforge.net/`.

[13] F. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 13(3):319–340, 1989.

[14] P. Z. M. de Lara, D. V. Tacoronte, and J.-M. T. Ding. Do current anti-cyberloafing disciplinary practices have a replica in research findings?: A study of the effects of coercive strategies on workplace internet misuse. *Internet Research*, 16(4):450–467, 2006.

[15] Facebook, Incorporated. Facebook bug bounty. `https://www.facebook.com/BugBounty`.

[16] GitHub, Incorporated. Github security bug bounty program. `https://bounty.github.com/`.

[17] G. T. Henry. *Practical sampling*, volume 21. Sage, 1990.

[18] D. Hovemeyer and W. Pugh. Finding bugs is easy. *ACM SIGPLAN Notices*, 39(12):92, Dec. 2004.

[19] M. Howard and S. Lipner. *The security development lifecycle*. O'Reilly Media, Incorporated, 2009.

[20] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge. Why don't software developers use static analysis tools to find bugs? In *Proceedings of the 2013 International Conference on Software Engineering*, pages 672–681, 2013.

[21] Klocwork Incorporated. Source code analysis tools. `http://www.klocwork.com/`.

[22] R. Koppel, T. Wetterneck, J. L. Telles, and B.-T. Karsh. Workarounds to barcode medication administration systems: their occurrences, causes, and threats to patient safety. *Journal of the American Medical Informatics Association*, 15(4):408–423, 2008.

[23] R. Little and D. Rubin. *Statistical analysis with missing data*. Wiley and Sons, 2nd edition, 2002.

[24] P. V. Marsden and J. D. Wright. *Handbook of survey research*. Emerald Group Publishing, 2010.

[25] G. McGraw. Software security. *Security & Privacy, IEEE*, 2004.

[26] L. Meyerovich and A. Rabkin. Empirical analysis of programming language adoption. *Proceedings of the 2013 International Conference on Object Oriented Programming Systems, Languages, and Applications*, 48(10):1–18, Nov. 2013.

[27] Microsoft. Microsoft security development lifecycle. `http://www.microsoft.com/security/sdl/default.aspx`.

[28] G. Moore and I. Benbasat. Development of an instrument to measure the perceptions of adopting an information technology innovation. *Information systems research*, 1991.

[29] Mozilla.org. Mozilla security bug bounty program. `https://www.mozilla.org/security/bug-bounty.html`.

[30] E. Murphy-Hill and G. C. Murphy. Peer interaction effectively, yet infrequently, enables programmers to discover new tools. *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, pages 405–414, 2011.

[31] N. Nethercote and J. Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. *ACM SIGPLAN Notices*, 42(6):89–100, 2007.

[32] OWASP Foundation. Owasp. `https://www.owasp.org/ index.php/Main_Page`.

[33] T. Raghunathan and J. Grizzle. A Split Questionnaire Survey Design. *Journal of the American Statistical Association*, 90(429):54–63, 1995.

[34] E. M. Rogers. *Diffusion of innovations*. 1995.

[35] E. M. Rogers. Diffusion of preventive innovations. *Addictive behaviors*, 27(6):989–993, 2002.

[36] D. Rubin, H. Stern, and V. Vehovar. Handling "don't know" survey responses: The case of the Slovenian plebiscite. *Journal of the American Statistical Association*, 90(431):822–828, 1995.

[37] Y. Sakamoto, M. Ishiguro, and G. Kitagawa. *Akaike information criterion statistics*. KTK Scientific Publishers, 1986.

[38] L. Singer. *Improving the adoption of software engineering practices through persuasive interventions*. PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2013.

[39] E. Smith, R. Loftin, E. Murphy-Hill, C. Bird, and

T. Zimmermann. Improving developer participation rates in surveys. In *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 7–10, 2013.

[40] E. T. Stringer. *Action research*. Sage, 2013.

[41] M. Szumilas. Explaining odds ratios. *Journal of the Canadian Academy of Child and Adolescent Psychiatry*, 19(3):227, 2010.

[42] M. Tan and T. Teo. Factors Influencing the Adoption of Internet Banking. *Journal of the AIS*, 1(July), 2000.

[43] R. C. Team. R language definition, 2000.

[44] Twitter. Hackerone - twitter. `https://hackerone.com/twitter`.

[45] J. Witschey, S. Xiao, and E. Murphy-Hill. Technical and personal factors influencing developers' adoption of security tools. *Proceedings of the CCS Workshop on Security Information Workers*, 2014.

[46] S. Xiao, J. Witschey, and E. Murphy-Hill. Social Influences on Secure Development Tool Adoption: Why Security Tools Spread. In *Computer Supported Cooperative Work (CSCW)*, pages 1095–1106, 2014.

[47] J. Xie, H. R. Lipford, and B. Chu. Why do programmers make security errors? In *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 161–164, 2011.

[48] X.Org Foundation. xorg. `http://www.x.org/wiki/`.