Technical and Personal Factors Influencing Developers' Adoption of Security Tools

Jim Witschey, Shundan Xiao, and Emerson Murphy-Hill
North Carolina State University
Raleigh, North Carolina
{jim_witschey, sxiao}@ncsu.edu, emerson@csc.ncsu.edu

ABSTRACT

Security tools analyze programs to help software developers write more secure code. Although these tools have been demonstrated to find vulnerabilities that human developers may not, many developers do not use them, leaving software needlessly vulnerable. To help understand why, we describe a theoretical account of factors that influence developers' adoption decisions. This model was developed based on interviews with 42 professional developers, and is a first step toward a comprehensive theory of security tool adoption based on diffusion of innovations theory.

1. INTRODUCTION

Secure software development tools, or *security tools*, are programs that analyze software to help developers find and fix vulnerabilities. Some, like `cppcheck` and some Valgrind plugins [8], are general-purpose static and dynamic analysis tools that help developers find programming errors. Others, like Klocwork¹, are designed specifically for security analysis. Some, like `cppcheck`, are even free and open-source, so developers have easy access to them.

However, many developers do not use security tools, which leads to the release of software containing vulnerabilities that could have been found automatically. For example, the famous `goto fail`; vulnerability in Apple's SSL implementation could have been found with proper application of static analysis tools, according to a blog post from Coverity². It seems, then, that developers' current security tool adoption behavior leaves many vulnerabilities in code when they could be found automatically and fixed before being released. We call this the *security tool adoption problem*.

The goal of our research is to increase adoption of security tools so more vulnerabilities in software are caught early.

¹<http://www.klocwork.com/>

²<http://security.coverity.com/blog/2014/Feb/a-quick-post-on-apple-security-55471-aka-goto-fail.html>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SIW'14 November 07 2014, Scottsdale, AZ, USA. Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 978-1-4503-3152-4/14/11 ...\$15.00

<http://dx.doi.org/10.1145/2663887.2663898>

We focus on tools for software developers because of the importance of software in many secure systems. Even if users of such systems behaved in security-preserving ways, security flaws in the software they used could make the data and resources used by the software available to adversaries. Increased adoption of security tools would help close this class of security vulnerabilities. This focus also allows us to specifically investigate developers' motivations for writing secure software, rather than their motivations for using tools more generally.

This paper describes a more theoretical model of security tool adoption, which identifies a number of factors influencing adoption. The factors chosen for the model and its high-level structure are informed by diffusion of innovations (DOI) theory and interviews with 42 professional developers.

Many aspects of our model conform with previous findings in DOI and developer tool use research. For instance, we found factors involving developer interaction, such as seeing others using security tools or working with security experts, may influence their adoption behavior.

In summary, this paper presents technical and personal factors that influence security tool adoption, informed by our interview study. Technical factors are captured in our **Innovation** factor group, and personal factors in our **Potential Adopter** factor group. The model is a first step toward a quantitative theory to describe and help promote the adoption of security tools. This theory can support future research on adoption of security tools and to prescribe practices to increase adoption in, and thus improve the quality of the code produced by, their companies and communities.

First, we discuss related work in Section 2. Then, we define and describe the model in Sections 4 and 5. Finally, we discuss threats to validity in Section 6 and implications for future research in Section 7.

2. RELATED WORK

We use theory from diffusion of innovations (DOI) research [9] as a starting point for our research. Since the 1960s, DOI has been applied to the adoption many technologies, as diverse as personal digital workstations, breeds of corn, and dance moves. DOI is a broad field of adoption study that accounts for many aspects of adoption, including temporal aspects of how innovations spread and how individuals make decisions about adoption, influences that promote or inhibit adoption, and how individuals and institutions can promote adoption. Our work draws upon parts of the theory that name factors influencing adoption as a starting point. Future work may also account for temporal

aspects of security tool adoption, and can build upon our work to develop interventions to promote adoption.

There are other theories that explain adoption [1, 4], but we chose DOI because it accounts for many social aspects of tool adoption. For instance, when explaining individuals' adoption choices, DOI explicitly accounts for the communication channel through which the innovation was communicated. We sought a theoretical grounding incorporating social aspects of adoption because of the results of our previous work, which investigated how developers learn about new tools and found that learning about tools through peers effectively leads to adoption [7]. In other previous work [5], we identified a number of strictly technical and usability concerns that influence developers' decisions to use or not use program analysis tools. The present paper differs from our previous research in that it presents technical and personal factors, broadening the foundation of our theory of security tool adoption.

Other software engineering researchers have also used DOI as a basis for their work. Meyerovich and Rabkin [6] used DOI as part of their study of programming language adoption, and found that a number of social and socially-driven factors, such as the availability of free and open-source libraries, drove and fostered programming language adoption. Singer [10] also used DOI theory to develop interventions that increased student developers' adoption of version control practices. While these other researchers found DOI research applicable to software development problems, we are the first to apply it to security tools. In addition, we use DOI as a starting point for a security tool-specific theory, rather than applying DOI directly.

In a previous paper [11], we used the same 42 interviews described in the present paper to examine social factors influencing security tool adoption. We direct the reader to that paper for our description of Social System and Communication Channel factors. In the present paper, we draw data from these interviews again to explore two other groups of factors, presented in Sections 5.1 and 5.2. Together, the factors form our model of security tool adoption.

3. METHODOLOGY

We conducted 42 one-on-one interviews with professionals in software development. We recruited these participants through personal and professional contacts, fliers in a major software development company, and through companies hiring software developers on Guru, a recruitment website. We also employed snowball sampling, asking participants to publicize the study to friends and colleagues. We incentivized participation with \$50 gift cards.

Participants completed a consent form and screening survey online before participating in the interviews. All participants were at least 18 years of age, worked professionally in software development, and reported writing code as part of their jobs in the past 6 months. Five developers also held management roles, 4 held secure software development roles, and 2 were consultants.

The 42 participants worked in many countries and developed many kinds of software, including web browsers, banking systems and software-as-a-service systems. They had a wide range of professional development experience, with a mean of 11 years and a median of 7.3 years experience.

Our interviews each consisted of 45 minutes to one hour of video chat or audio conversations. The interviews were

semi-structured: the interviewer followed a script, but could deviate from the script if necessary to explore something interesting a participant said. If these questions yielded interesting information, they were added to the script for subsequent interviews; if a question stopped yielding interesting information, it was removed from the script.

We transcribed the recordings of the interviews, then coded the interviews using Atlas.ti qualitative analysis software. We began with codes drawn from DOI theory and added codes as necessary to account for new patterns in the data. The model presented in the following section was developed by analyzing and synthesizing these codes.

For a more detailed discussion of our methodology, please see our previous paper drawing on this study [11].

4. SECURITY TOOL ADOPTION MODEL

In this paper, we present our security tool adoption model. This model identifies and categorizes factors that influence developers' security tool adoption decisions. These 14 factors are divided into four categories:

- **Innovation** factors describe how the technical 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 personal qualities of the individual developers themselves affect their adoption decisions.

Our model uses these factors to account for influences on developers' adoption decisions, which we identified in our interview study.

In our previous paper based on these interviews, we described how **Social System** factors, including the *policies and standards* and *culture* surrounding security, the *security concern* of the software being built, the *structure* of the company where a developer works, and the *education and training* available affect adoption. We also described how properties of different **Communication Channels**, including developers' *trust* in them and *exposure* to security tools through them, affect adoption [11].

In this paper, we describe **Innovation** and **Potential Adopter** for the first time, in Sections 5.1 and 5.2, respectively. We direct the reader to our previous paper [11] for a detailed discussion of Social System and Communication Channel factors and the interview data that informed these factors.

5. FACTORS INFLUENCING ADOPTION

In the following subsections, we describe the technical and personal factors in our model. These technical and personal factors fall under the **Innovation** and **Potential Adopter** factor groups, respectively. The name of each factor is introduced in *italics*. If a number of participants is presented

alone, the data comes not from a direct response to a question, but from our coding of responses to other questions. Where possible, we will use phrases like “ x out of y interviewees said A ” when reporting how many interviewees were influenced by a given factor. Although we interviewed 42 developers, y will not always be 42, since we did not ask every interviewee every question. We do not recommend the reader infer any statistical information from these proportions. We cannot claim, for instance, that a statement made by 10 participants is more important than a statement made by 5 participants; we include these numbers only to give the reader a sense of our data.

5.1 Innovation Factors

In our interviews, we found that the properties of security tools themselves affect the likelihood that developers will adopt them. We use five concepts from DOI research to describe how these properties affect adoption.

Relative advantage in DOI theory accounts for the perceived advantages and disadvantages of adopting one tool over another, or over using no tool. Three interviewees believed security tools had similar functionality and were about as effective as each other. One did not see an immediate benefit after installing a tool, and so did not continue using it. Five chose not to use security tools because the ones they had used were too slow or used too many resources. Seven said mature security tools were too expensive to be cost-effective and chose not to use security tools at all. For instance, one interviewee’s company, rather than purchasing an expensive security tool, chose to implement quality-assurance practices like code review and build tools in-house to ensure SQL queries are constructed safely.

The time it took to install, configure, run, and interpret the output of security tools also contributed to interviewees’ perceptions of their cost. Eleven interviewees said security tools generated many false positives, and one noted that this greatly increased the time “cost” of tool use without increasing its benefits.

Compatibility, another factor from DOI, with existing tools and technologies was also important to interviewees. Three interviewees who used dynamic programming languages did not use security tools because they did not believe good security tools existed for these languages. One interviewee thought tools for Ruby and other dynamic languages had only recently become as mature as those for Java. In general, interviewees chose security tools that were compatible with the language, IDE, and operating system they already used, and if they could not find a satisfactory security tool compatible with their current way of working, they would not use any security tool. Security tools were not so important to interviewees that they would switch to different technologies for which security tools were available. Overall, however, interviewees were satisfied with the compatibility of existing tools with the technologies they used. We believe that this is because most of the interviewees we interviewed used widely-used languages, IDEs, and operating systems, and most security tools target those common platforms.

Some interviewees did not adopt security tools because of the tools’ *complexity*, as predicted by DOI theory. Some tools are highly complex, have a lot of functionality, and support many analyses. While rich functionality can be a positive thing, three interviewees are put off by this complexity and prefer more focused tools; one interviewee even

said that performing simple tasks in these “overdeveloped” tools was a “a nightmare”. Three interviewees were discouraged from using a tool by the complexity of its interface, and four said they had difficulty understanding a tool’s output. Three other interviewees said security tools were so complex that only a security expert could use them. In general, many interviewees did not use security tools due to their complexity, and would prefer more focused and easier-to-use tools. These findings are consistent with our previous work, which found that developers do not use analysis tools because their results are difficult to understand [5].

Trialability is defined in DOI theory as the ease with which individuals can try out and evaluate an innovation. In DOI, more trialable tools are more likely to be adopted, and our findings demonstrate that this generalization holds for security tools. In the case of security tools, a free tool that searches code for strings that can allow SQL injection attacks could be relatively trialable. In comparison, an expensive security tool that requires a long configuration process and does not offer a free trial would be less trialable, due to the financial and time costs of trying it out. Many interviewees indicated they had rejected some tool in the past because it was difficult to try out and evaluate. One, for instance, was unwilling to stop other, more pressing work for a long enough period to install and configure a tool.

Another element of trialability is the ease or difficulty of learning to use a security tool. Eleven of twenty-three interviewees said they would read a tool’s documentation to learn how to use a security tool, seven would ask coworkers for guidance, and three would look for tutorials online. Two said they would prefer playing with a tool to other forms of instruction. We note that this creates a relationship between a tool’s complexity and its trialability: more complex tools, whose functionality is not obvious without instruction, will be less trialable, and thus less adopted, by developers who prefer to learn in this way.

Observability is one important property of innovations identified by Rogers [9]: innovations whose use and effects of use are easy for others to see are more likely to spread. Our findings indicate that security tools are not very observable; no interviewees indicated they had opportunistically observed other developers using security tools. One interviewee said another developer opportunistically saw her using an SQL query tool, but this is not a security tool. This indicates that our previous findings [7], that developers do not often observe their peers using tools, apply for security tools, perhaps even more so than tools in general.

5.2 Potential Adopter Factors

Personal qualities of developers, the potential adopters of security tools, also have influence on developers’ likelihood to adopt or not adopt security tools. The factors we discovered in our interviews are *experience* and *inquisitiveness*, and are described below.

We noticed a correlation between the length of interviewees’ *experience* as software developers and their security tool adoption decisions, though our study was not designed to determine if such a correlation was significant. Such a correlation could occur simply because developers who have developed software longer are more likely to have had software compromised in the past, and thus have been cued to adopt security tools, as described in our previous paper on social influences in security tool adoption [11]. It also could occur

because senior developers are able to adopt tools without going through the same approval process as less experienced developers, so security tools are more trialable for them. If this correlation holds in future studies, it would agree with Chau and Lung Hui’s findings showing that more experienced computer users were more likely to be early adopters of Windows 98 [2]. Thus, security tools may be similar to other IT innovations with respect to adoption.

We also found that a developer’s *inquisitiveness* about security tools affects the likelihood she will adopt a security tool. Six of thirty-one interviewees did not seek out new security tools or information about them because doing so was not required for their jobs, and none of these interviewees used security tools regularly. Twenty-five of thirty-one interviewees said that, if they needed to write more secure code, they would seek out information about security tools. Some would seek out security tools through web search. Others would follow technical blogs and news sites for information about security tools. Some would even go to security conferences to learn about tools and other security topics.

6. THREATS TO VALIDITY

There are a number of limitations that the reader should keep in mind when interpreting our model and the results of our interview study.

Like all retrospective interviews, our study is limited by the *recall problem*. Some interviewees explicitly mentioned they could not remember all the tools they had used. We tried to mitigate this limitation by providing interviewees with Wikipedia’s list of static analysis tools³, which is grouped by the languages with which the tools are compatible. We asked participants to read this list after listing the tools they recalled without help.

Our model is also limited in that it only uses non-temporal aspects of DOI theory. DOI accounts for many parts of the process of adoption decision-making, but we deliberately focused on finding factors to develop our model.

Our interviews were conducted with industry developers. Thus, our model may not account for factors that influence adoption in software development outside of industry, such as that of some open-source software.

We have used DOI theory both to design our study and to analyze our results. Thus, we may have biased the results toward the factors that we hypothesized would influence adoption. We tried to reduce this risk by asking participants open-ended questions, and integrating factors into our model if they were not previously included in it.

7. FUTURE WORK

We believe this model can be of greater use to practitioners and researchers alike if enriched with quantitative data about developers’ attitudes toward security tools and their influence on adoption. To this end, we are currently surveying developers to discover the strength of these relationships.

DOI research indicates that individuals can be *champions* for an innovation and try to spread it within a social group. Das and colleagues found a number of ways individuals act as champions for personal data security practices, discussing and spreading awareness in their own social groups [3]. More study into security tool adoption could find how developers

discuss and spread security tools, and research on potential adopter factors could help practitioners better identify potential champions

These studies will facilitate future experimental research that can attempt to observe the effects that different interventions have on adoption behavior. The results of such studies could help practitioners choose the best ways to promote security tool adoption in their organizations and communities, and improve the security of their software.

8. 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 model accounting for factors that influence security tool adoption and their relative importance. This model provides a theoretical framework to help toolsmiths and policy-makers understand why tools are or are not adopted.

Acknowledgments

Thanks to Dr. Jorge Aranda, Dr. Leif Singer, Da Young Lee, the Developer Liberation Front, and the Science of Security Lablet for their helpful feedback. Thanks for the National Security Agency for funding this research. Special thanks to participants for their time and input.

9. REFERENCES

- [1] I. Ajzen. The theory of planned behavior. *Organizational behavior and human decision processes*, 1991.
- [2] 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.
- [3] S. Das, T. H.-J. Kim, L. A. Dabbish, and J. I. Hong. The effect of social influence on security sensitivity. *SOUPS*, 2014.
- [4] F. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 13(3):319–340, 1989.
- [5] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge. Why don’t software developers use static analysis tools to find bugs? *ICSE*, pages 672–681, 2013.
- [6] L. Meyerovich and A. Rabkin. Empirical analysis of programming language adoption. *OOPSLA*, 48(10):1–18, Nov. 2013.
- [7] E. Murphy-Hill and G. C. Murphy. Peer interaction effectively, yet infrequently, enables programmers to discover new tools. *CSCW*, pages 405–414, 2011.
- [8] N. Nethercote and J. Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. *ACM SIGPLAN Notices*, 42(6):89–100, 2007.
- [9] E. M. Rogers. *Diffusion of innovations*. 1995.
- [10] L. Singer. *Improving the adoption of software engineering practices through persuasive interventions*. PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2013.
- [11] S. Xiao, J. Witschey, and E. Murphy-Hill. Social Influences on Secure Development Tool Adoption: Why Security Tools Spread. *CSCW*, pages 1095–1106, 2014.

³http://en.m.wikipedia.org/wiki/List_of_tools_for_static_code_analysis