You Are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users

Yasemin Acar and Sascha Fahl
CISPA, Saarland University
{acar,fahl}@cs.uni-saarland.de

Michelle L. Mazurek
University of Maryland
mmazurek@umd.edu

***Abstract*—While researchers have developed many tools, techniques, and protocols for improving software security, exploits and breaches are only becoming more frequent. Some of this gap between theoretical security and actual vulnerability can be explained by insufficient consideration of human factors, broadly termed usability, when developing these mechanisms. In particular, security mechanisms may be difficult to use, may conflict with other priorities, or may assume more security knowledge than users possess. For almost 20 years, the usable security community has investigated how to improve the usability of security tools and interfaces aimed at end users. More recently, the community has begun to apply similar techniques in the context of improving security tools—such as APIs and bug-finding software—aimed not at end users but at developers, whose security errors are magnified across all users of their products. In this paper, we review key lessons learned from usable security for end users and consider how to apply them in the context of developers. We propose a research agenda aimed at developing a high-quality, comprehensive literature for usable security for developers, including: investigating how to conduct reliable research in this context; understanding developers' attitudes, knowledge, and priorities; measuring the status quo; and developing improved tools and interventions in the future.**

I. Introduction

In recent years, attacks and data breaches have become commonplace. Personal and corporate data is attacked over and over again [1]. Security researchers have demonstrated cryptographic algorithms, memory-safe applications, and access control that can deliver provably strong (if not perfect) protection from many such attacks. Nonetheless, the rate of cyber attacks seems to only increase [2]. Historically, the huge gap between the theoretical strong security offered by these mechanisms and actual low security in practice is often caused by the poor usability of security solutions.

Email encryption is one impressive example: asymmetric encryption dates from the 1970s [3], [4] and PGP [5] was introduced in 1991, but even in today's environment of organized cybercrime and nation-state surveillance, almost no one uses end-to-end email encryption. In 1999, Whitten and Tygar's seminal paper analyzing email encryption as a usability problem [6] helped to establish a new research field, which later became the usable security and privacy community. Since then, usability problems—broadly defined to include other human and social science factors, such as economics and cognitive biases—have been identified as a major factor in users disregarding existing security and privacy mechanisms. The usable security and privacy research community aims to improve the usability of existing mechanisms as well as to offer guidelines for designing new mechanisms with better usability built in. Topics that have received significant attention from this community include email encryption [6]–[11], passwords and alternative authentication mechanisms [12]–[17], and security-relevant user interactions such as warning messages and security indicators [18]–[23].

While progress has been made in improving end-users' adherence and sometimes even comprehension of security-critical issues, a key constituency has thus far been understudied: Software developers make security and privacy decisions that have a huge impact on end-user (and therefore overall ecosystem) security, and they suffer from similar comprehension and adherence problems to end users. Although usable security and privacy research focusing on developers is still in an early stage, preliminary results illustrate a common theme: Developers are regular users of security and privacy mechanisms (e.g., security APIs, protocols, and tools), but are by no means security experts [24], [25]. We argue for a systematic approach to studying developers within the security ecosystem. While developer-usability studies targeting specific security tools and APIs are becoming more common [26]–[28], topics are fragmented and quality research norms have not yet been firmly established. In this position paper, we argue for systematizing future research on usable security [1] for developers, including working to validate promising research methods and identifying key areas of focus.

II. Lessons Learned from Studying Usable Security for End-Users

We briefly discuss key lessons learned from more than 15 years of research into usable security for end users, and how these lessons can apply in the developer space.

[1]For simplicity, throughout this paper we refer to *security and privacy* as *security*. We find that privacy-preserving or -enhancing behavior often requires the use of secure mechanisms, while good security practice often protects privacy. The research techniques and approaches we discuss generally apply well for both.

A. You Are Not Your User

Plentiful research has demonstrated that unusable end-user security tools and interfaces frequently arise when the developers of these tools make unfounded assumptions about what the intended users know and understand. Examples include everything from encryption tools that expect users to understand the difference between encryption and signing [6], to browser warnings and app permission descriptions that use too much security jargon [18], to expecting users to understand the importance of software updates [29]–[32]. In each of these cases, security experts have expected end users to know and care about security, perhaps because of assumed similarity bias, in which people often assume that everyone is similar to themselves and the people they know [33].

This lesson applies even more strongly when considering security tools and APIs used by developers. Because developers by definition have some level of technical expertise, it is easy for security experts to mistakenly believe that developer-users also understand security, or that expert tools need not be designed with usability in mind. It seems likely this fallacy is at the root of unusable cryptography APIs, as well as difficult-to-interpret outputs from bug-finding tools.

In the case of end users, these problems have been mitigated somewhat by reminding tool developers to consider the different needs and attitudes of end users, and by explicitly evaluating usability rather than making assumptions about what is usable [22], [34]. We believe similar solutions can be helpful when building security tools for developers; in Section III-C, we discuss potential targets for usability evaluation.

B. Security Is A Secondary Concern

The usable-security field has firmly established that security is a secondary concern for end users; when it gets in the way of a user's primary goal, security becomes an annoyance to be worked around or ignored. As examples, end users adopt insecure password practices when requirements become too onerous [35], [36] and ignore security icons and warnings when they are motivated to proceed to their goal [19], [37].

This concept applies equally to developers, who have priorities—functional correctness, time to market, maintainability, economics, compliance with other corporate policies—that sometimes appear to conflict with security and are often more salient [38]. For end users, the usable-security community frequently recommends taking users out of the loop as much as possible [39], such as by making updates automatic, choosing secure defaults, and forcing browsers to use HTTPS. When removing the user from the loop is infeasible, the community has often emphasized *opinionated design*, also called nudging or soft paternalism, which encourages users to make more secure choices even if they do not entirely understand the situation. For example, browser certificate warnings are designed to discourage click-through [40]. In Section IV we discuss ideas for applying these approaches to developers as well.

C. More is Not Always Better

A third key lesson from usable security for end users is that simply adding more and more security advice and recommendations is not a viable solution. Piling on advice can overwhelm users and lead them to give up on taking any steps to improve security; similarly, encountering too many warnings that don't lead to actual harms causes habituation and disengagement. While the usable-security community continues to struggle with this problem, recently researchers are acknowledging the overabundance of unhelpful advice and even advocating rollback of some overzealous policies, such as password expiration [41]–[43]. This overabundance of security advice has shifted the problem for end users to choosing which information sources they trust or rely on the most [44].

Related issues are beginning to be seen in the advocacy of secure development; for example, the proliferation of new, sometimes incompatible, encryption libraries claiming both security and usability with little or no empirical evaluation. While the end-user security community has not identified any comprehensive solution to this problem, we encourage the developer security community to bear in mind that simply asking developers to do more and more in the name of security is unlikely to help and may even exacerbate the problem.

III. A Research Agenda for Usable Security for Developers

We believe that thus far, usable security for developers has been a critically under-investigated area. Recently, the topic has begun to receive more attention, and we expect that in the near future many researchers will address it. In this section, we lay out a high-level research agenda covering what we believe are the most important needs in this area. We organize our suggestions into four areas to investigate: how best to conduct usable security research with developers; how developers think about security in the context of their needs and priorities; how usable current security tools and APIs are and where they fall short; and how to build more usable tools and paradigms in the future.

A. Methodology and Ecological Validity

One major concern with studying usable security for developers is ecological validity: whether or not the circumstances of a study accurately reflect the real world [45]. While this is a challenge for most user studies, it's especially challenging when targeting usable security for developers, for several reasons. Because security is a secondary concern, asking users about it directly may not effectively reflect realistic circumstances, in which developers may not be thinking about security or in which other priorities may outweigh security concerns. In addition, recruiting professional developers to study can be challenging: depending on the researcher's geographical area, there may not be many developers locally available, and those who are may be too busy to attend studies. The hourly rates these highly specialized people are typically paid will often exceed the researcher's available

budget. Finally, real-life development tasks are complicated and may be difficult to simulate in a study environment.

To address this challenge, we need methodological research investigating *how* to study developers' security behavior. One critical question is whether and in what circumstances computer science students, who are often studied out of convenience, can effectively substitute for professional developers. In our work examining how information resources impact developers' decision making, we asked both students and professionals to complete four time-limited, security-related programming tasks. We found that professionals outperformed students in functional correctness, but were no more secure [46]. While this result is intriguing, further investigation is needed. Is this result reproducible with other security tasks and environments? What constitutes a professional? How do professionals from big and small companies differ, and how do they compare to graduate and undergraduate students from different universities? Are lab studies necessary, or can online studies be useful? To answer these questions, controlled comparison studies are needed; we are currently conducting one such study comparing students and professionals.

Researchers should also investigate what kind of study tasks work best for evaluating security tools and behaviors; to do this, researchers should aim to compare controlled studies with field observations. We have previously applied similar methods to evaluate ecological validity for password studies [15], [16], while other researchers have addressed ecological validity, e.g., for studies of security indicators [19]. We can also learn from the software engineering community's work investigating developers and their tools and behaviors in non-security domains [47]–[50].

Key research questions:

- Which recruitment strategies provide representative samples efficiently?
- Which study and task designs are most appropriate to measure developers' motivations, attitudes and knowledge?

B. Understanding Developers' Motivations, Attitudes, and Knowledge

In a landmark 1999 article, Adams and Sasse challenged the conventional wisdom that users reject security behaviors—in this case password policies—due primarily to laziness or carelessness [35]. Instead, they argued, misbehavior stemmed primarily from misunderstandings, competing priorities, and challenging interfaces. A similar consensus is starting to emerge with respect to developers' security behaviors: although historically developers have been seen as "experts" in contrast to less knowledgeable end users, many (most) developers are not experts in security, and make errors through misunderstandings and difficult-to-use interfaces. In addition, developers have priorities—such as adding functionality, optimizing the end-user experience, reducing time-to-market, and reducing development costs—that often appear to be in conflict with best security practices. Before we can develop better tools, interfaces, and educational interventions to promote secure development, we must investigate what developers understand about security and how they view secure development in the context of their overall goals.

Acquiring this understanding can be approached in several ways. We can use qualitative interviews and quantitative surveys to ask developers directly about their security knowledge, attitudes, and decision-making processes. This parallels Adams and Sasse's work [35], as well as many subsequent papers evaluating end-user security attitudes and behaviors [30], [44], [51]–[53]. Balebako et al. used this approach to investigate how mobile app developers make privacy-relevant decisions, finding that lack of awareness and lack of resources contribute to poor privacy decisions [25]. In the same study, the authors report on some use of third-party security tools considered more secure than homemade implementations. We expect that a similar study focused explicitly on security attitudes and behaviors would find related barriers and more in-depth analysis of why and how third-party security tools are and are not used.

While studies in which participants are asked explicitly about their attitudes and behaviors provide valuable data and important context, self-reporting is inherently limited by human recall and by well-known psychological biases [54], [55]. To get a complete picture, therefore, we must supplement these findings with measurements of actual behavior. This can be obtained via in-situ observational studies (e.g., following developers to design meetings, observing their work in progress, etc.), and by field or diary studies in which developers report on their security-relevant decisions as they make them. This might include observing decisions like which libraries to use, what security threat model is appropriate, and whether to use, e.g., bug-finding or fuzzing tools. While these studies can be complicated, expensive, and time-consuming, they provide rich data with strong validity that often cannot be obtained any other way.

Key research questions:

- What motivates developers to use secure mechanisms and concepts, and how can we use this to improve the status quo?
- What prevents developers from adhering to secure recommendations, and how can we counter this?
- Which information sources do developers turn to and trust, and how can we use this to improve security?
- Where do developers lack knowledge, and how can we either provide them with secure information sources or secure their software without requiring security education?

C. Investigating the Status Quo

In addition to understanding developers' knowledge and attitudes, we must investigate how existing APIs, documen-

tation, and tools encourage or discourage good security behaviors. By identifying which tools work well and which fail, and why, we can improve existing tools and build new ones that are more likely to be effective.

Existing tools and APIs can be evaluated via field and measurement studies that capture security behaviors, implementations, and mistakes across a broad swathe of software. For example, several studies have examined the use of TLS and cryptography more generally in mobile apps and identified common pitfalls and errors [24], [56]–[58]. We propose further measurements, such as examining how insecure code propagates on GitHub, or studying how popularity of different security libraries correlates with common errors. These kinds of measurements can potentially be extended by contacting involved developers for follow-up interviews concerning how libraries were chosen or how errors were made.

While field measurements provide a valuable large-scale look at how tools and APIs are used in practice, they do not allow researchers to isolate and test specific hypotheses. Thus, we also recommend controlled lab experiments to measure how concrete factors affect developers' decisions. For example, in recent work we examined how using Stack Overflow compared to official documentation affected the security of code Android developers wrote in response to short programming tasks [46]. We are currently deploying an experiment comparing how different cryptography APIs affect the code developers write. Researchers should also measure the usability of existing bug-finding and fuzzing tools to identify problems and pain points; these studies could be modeled on investigations of usability for security tools used by end users, such as [6]–[11], [19], [59].

In addition to field studies and lab measurements, expert review (including, e.g., *cognitive walkthroughs* and *heuristic evaluations*) of tools and APIs for usability can provide valuable feedback to their authors with less time and expense. We propose that researchers evaluate groups of related APIs and tools to provide clear evidence of the benefits and drawbacks of each. Expert reviews are frequently used in HCI generally and in usable security specifically [6], [60]–[62].

Key research questions:

- How well do current APIs, documentation, and tools support secure behavior?
- In which ways should future APIs, documentation, and tools be designed to encourage secure behavior?
- Which of APIs, documentation, and tools has the most promising impact on security; where should we place the focus of our research?

IV. Promising Concrete Next Steps

In this section we discuss how to apply the lessons learned for end users, described in Section II, in a developer context to improve existing mechanisms and build better ones.

a) Usable security APIs: The software engineering community has developed guidelines for designing usable APIs and tools generally [63]–[67], and security researchers have considered API usability at a high level as well [68], [69]. Guidelines from all these sources should be synthesized and extended to provide concrete objectives for security APIs. We are currently developing a framework for measuring the usability of security APIs, and applying this framework to evaluate security APIs in the wild. Further work should be done both to make existing APIs more usable—including via better documentation—as well as to introduce new APIs that balance security and usability.

b) Secure, usable information resources: We have shown that developers make insecure choices when the (usable) resource they turn to for help is offering quick but insecure fixes [46]. To address this, we advocate making official documentation (which already promotes security) more interactive and usable, and to introduce security monitoring to usable resources. More research is needed on how to best combine usability with security in developer resources.

c) Developer tool support: Integrating tool support into developer environments can both raise security awareness and provide direct security feedback. For example, we are working on an exemplar Android Studio plugin that applies static code analysis to help developers to turn insecure choices into more secure ones [70]. While developer support and IDEs that make developing faster and easier exist, no security tools to speak of are in use.

d) Taking developers out of the loop: We recommend removing developers from the security loop whenever possible. We have shown in the past that developers who implement custom SSL/TLS handling nearly always make insecure choices; in response, we suggested configurable TLS handling at the OS level [24]. In a similar vein, we recommend further research aimed at moving security management and security-critical decisions from apps to the OS and framework levels whenever possible. This includes but is not limited to automatic security library updates, or automatic permission requests on Android. Not only could this reduce developers' opportunities to make errors, but it is also compatible with the tendency to prioritize reducing development time and effort over security correctness. Research is needed to identify cases where this is possible as well as to suggest effective ways to remove developers from the security loop without overly restricting functionality.

V. Conclusion

In this paper we advocate a systematic, organized effort to understand developers' attitudes, needs, and priorities toward security. Based on this understanding, security tools and APIs can be improved to increase adoption and adherence. Advancing usable security for developers will be challenging, but it has the potential to bring already-known solutions into greater use and provide enormous benefits to the overall security ecosystem.

REFERENCES

[1] S. Ramanan, "The top 10 security breaches of 2015," http://www.forbes.com/sites/quora/2015/12/31/the-top-10-security-breaches-of-2015/#7a67d9d5694f, 2015.

[2] Symantec, "2016 Internet Security Threat Report."

[3] W. Diffie and M. Hellman, "New directions in cryptography," in *IEEE Transactions on Information Theory*, 1976.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," in *Communications of the ACM*, 1978.

[5] P. Zimmermann, "PGP version 2.6.2 user's guide," ftp://ftp.pgpi.org/pub/pgp/2.x/doc/pgpdoc1.txt, 1994.

[6] A. Whitten and J. D. Tygar, "Why johnny can't encrypt: A usability evaluation of pgp 5.0," in *USENIX Security*, 1999.

[7] S. Ruoti, N. Kim, B. Ben, T. van der Horst, and K. Seamons, "Confused Johnny: When automatic encryption leads to confusion and mistakes," in *Symposium on Usable Privacy and Security*, 2013.

[8] S. Fahl, M. Harbach, T. Muders, M. Smith, and U. Sander, "Helping johnny 2.0 to encrypt his facebook conversations," in *Symposium on Usable Privacy and Security*, 2012.

[9] S. Sheng, L. Broderick, C. A. Koranda, and J. J. Hyland, "Why johnny still can't encrypt: evaluating the usability of email encryption software," in *Symposium on Usable Privacy and Security*, 2006.

[10] S. L. Garfinkel and R. C. Miller, "Johnny 2: a user test of key continuity management with s/mime and outlook express," in *Symposium on Usable Privacy and Security*, 2005.

[11] S. Ruoti, J. Andersen, S. Heidbrink, M. O'Neill, E. Vaziripour, J. Wu, D. Zappala, and K. Seamons, ""we're on the same page": A usability study of secure email using pairs of novice users," in *Conference on Human Factors in Computing Systems*, 2016.

[12] R. Biddle, S. Chiasson, and P. C. van Oorschot, "Graphical passwords: Learning from the first twelve years," in *Computing Surveys*, 2012.

[13] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of passwords and people: measuring the effect of password-composition policies," in *Conference on Human Factors in Computing Systems*, 2011.

[14] M. Harbach, A. De Luca, and S. Egelman, "The Anatomy of Smartphone Unlocking," in *Conference on Human Factors in Computing Systems*, 2016.

[15] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *Conference on Computer and Communications Security*, 2013.

[16] S. Fahl, M. Harbach, Y. Acar, and M. Smith, "On The Ecological Validity of a Password Study," in *Symposium on Usable Privacy and Security*, 2013.

[17] F. Stajano, "Pico: No More Passwords!" in *International Workshop on Security Protocols*, 2011.

[18] J. Sunshine, S. Egelman, H. Almuhimedi, and N. Atri, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness." in *USENIX Security*, 2009.

[19] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, "The Emperor's New Security Indicators," in *IEEE Symposium on Security and Privacy*, 2007.

[20] D. Akhawe and A. P. Felt, "Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness." in *USENIX Security*, 2013.

[21] A. P. Felt, R. W. Reeder, A. Ainslie, H. Harris, M. Walker, C. Thompson, M. E. Acer, E. Morant, and S. Consolvo, "Rethinking connection security indicators," in *Symposium on Usable Privacy and Security*, 2016.

[22] J. Weinberger and A. P. Felt, "A week to remember: The impact of browser warning storage policies," in *Symposium on Usable Privacy and Security*, 2016.

[23] C. Bravo-Lillo, S. Komanduri, L. F. Cranor, R. W. Reeder, M. Sleeper, J. Downs, and S. Schechter, "Your attention please: Designing security-decision UIs to make genuine risks harder to ignore," in *Symposium on Usable Privacy and Security*, 2013.

[24] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith, "Rethinking SSL Development in an Appified World," in *Conference on Computer and Communications Security*, 2013.

[25] R. Balebako, A. Marsh, J. Lin, and J. Hong, "The Privacy and Security Behaviors of Smartphone App Developers," in *Workshop on Usable Security*, 2014.

[26] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, and B. Fisher, "Towards understanding it security professionals and their tools," in *Symposium on Usable Privacy and Security*, 2007.

[27] J. Xie, B. Chu, H. R. Lipford, and J. T. Melton, "Aside: Ide support for web application security," in *Computer Security Applications Conference*, 2011.

[28] K. Y, S. D. Dechand, E. Gerhards-Padilla, and M. Smith, "Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study," in *IEEE Symposium on Security and Privacy*, 2016.

[29] A. Mathur, J. Engel, S. Sobti, V. Chang, and M. Chetty, ""they keep coming back like zombies": Improving software updating interfaces," in *Symposium on Usable Privacy and Security*, 2016.

[30] K. E. Vaniea, E. Rader, and R. Wash, "Betrayed by updates," in *Conference on Human Factors in Computing Systems*, 2014.

[31] M. Oltrogge, Y. Acar, S. Dechand, M. Smith, and S. Fahl, "To pin or not to pin—helping app developers bullet proof their tls connections," in *USENIX Security*, 2015.

[32] S. Fahl, S. Dechand, H. Perl, F. Fischer, J. Smrcek, and M. Smith, "Hey, nsa: Stay away from my market! future proofing app markets against powerful attackers," in *Conference on Computer and Communications Security*, 2014.

[33] R. Holtz and N. Miller, "Assumed similarity and opinion certainty," in *Journal of Personality and Social Psychology*, 1985.

[34] R. Reeder, E. C. Kowalczyk, and A. Shostack, "Helping engineers design neat security warnings," in *Symposium On Usable Privacy and Security*, 2011.

[35] A. Adams and M. A. Sasse, "Users are not the Enemy," in *Communications of the ACM*, 1999.

[36] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: an algorithmic framework and empirical analysis," in *Conference on Computer and Communications Security*, 2010.

[37] J. Lee, L. Bauer, and M. L. Mazurek, "The Effectiveness of Security Images in Internet Banking," in *IEEE Internet Computing*, 2015.

[38] R. Werlinger, K. Hawkey, and K. Beznosov, "An integrated view of human, organizational, and technological challenges of it security management," in *Information Management & Computer Security*, 2009.

[39] L. Cranor, "A Framework for Reasoning About the Human in the Loop," in *Usability, Psychology and Security*, 2008.

[40] A. P. Felt, A. Ainslie, R. W. Reeder, S. Consolvo, S. Thyagaraja, A. Bettes, H. Harris, and J. Grimes, "Improving ssl warnings: Comprehension and adherence," in *Conference on Human Factors and Computing Systems*, 2015.

[41] L. Cranor, "Time to rethink mandatory password changes," https://www.ftc.gov/news-events/blogs/techftc/2016/03/time-rethink-mandatory-password-changes/, 2016.

[42] C. Herley, "More Is Not the Answer," in *IEEE Security & Privacy*, 2014.

[43] S. Chiasson and P. C. van Oorschot, "Quantifying the security advantage of password expiration policies," in *Designs, Codes and Cryptography*, 2015.

[44] E. M. Redmiles, A. R. Malone, and M. L. Mazurek, "I Think They're Trying to Tell Me Something: Advice Sources and Selection for Digital Security," in *IEEE Symposium on Security and Privacy*, 2016.

[45] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects: A comparative study ofstudents and professionals in lead-time impact assessment," in *Empirical Software Engineering*, 2000.

[46] Y. Acar, M. Backes, S. Fahl, D. Kim, and M. L. Mazurek, "You Get Where You're Looking For: The Impact Of Information Sources On Code Security," in *IEEE Symposium on Security and Privacy*, 2016.

[47] T. Scheller and E. Kühn, "Usability Evaluation of Configuration-Based API Design Concepts," in *Human Factors in Computing and Informatics*, 2013.

[48] B. Ellis, J. Stylos, and B. Myers, "The Factory Pattern in API Design: A Usability Evaluation," in *International Conference on Software Engineering*, 2007.

[49] J. Stylos and B. A. Myers, "The implications of method placement on API learnability," in *ACM SIGSOFT International Symposium*, 2008.

[50] C. Burns, J. Ferreira, T. D. Hellmann, and F. Maurer, "Usable results from the field of API usability: A systematic mapping and further analysis," in *IEEE Symposium on Visual Languages and Human-Centric Computing*, 2012.

[51] R. Wash and E. Rader, "Too Much Knowledge? Security Beliefs and Protective Behaviors Among United States Internet Users," in *Symposium on Usable Privacy and Security*, 2015.
[52] E. Chin, A. P. Felt, V. Sekar, and D. Wagner, "Measuring user confidence in smartphone security and privacy," in *Symposium on Usable Privacy and Security*, 2012.
[53] D. K. Smetters and N. Good, "How users use access control," in *Symposium on Usable Privacy and Security*, 2009.
[54] K. Kelley, "Good practice in the conduct and reporting of survey research," in *International Journal for Quality in Health Care*, vol. 15, 2003.
[55] A. Tversky and D. Kahneman, "Judgment under Uncertainty: Heuristics and Biases," in *Utility, Probability, and Human Decision Making*, 1975.
[56] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An Empirical Study of Cryptographic Misuse in Android Applications," in *Conference on Computer and Communications Security*, 2013.
[57] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous Code in the World: Validating SSL Vertificates in non-browser Software," in *Conference on Computer and Communications Security*, 2012.
[58] B. Reaves, N. Scaife, A. Bates, and P. Traynor, "Mo(bile) Money, Mo(bile) Problems: Analysis of Branchless Banking Applications in the Developing World," in *USENIX Security*, 2015.
[59] A. P. Felt, R. W. Reeder, A. Ainslie, H. Harris, M. Walker, C. Thompson, M. E. Acer, E. Morant, and S. Consolvo, "Rethinking connection security indicators," in *Symposium on Usable Privacy and Security*, 2016.
[60] C. A. Brodie, C.-M. Karat, and J. Karat, "An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench," in *Symposium on Usable Privacy and Security*, 2006.
[61] J. Clark, P. C. van Oorschot, and C. Adams, "Usability of anonymous web browsing: an examination of Tor interfaces and deployability," in *Symposium on Usable Privacy and Security*, 2007.
[62] S. Eskandari, D. Barrera, and E. Stobert, "A first look at the usability of bitcoin key management," in *Workshop on Usable Security*, 2015.
[63] B. A. Myers and J. Stylos, "Improving API usability," in *Communications of the ACM*, 2016.
[64] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1993.
[65] S. Clarke, "Using the cognitive dimensions framework to design usable APIs," https://blogs.msdn.microsoft.com/stevencl/2003/11/14/using-the-cognitive-dimensions-framework-to-design-usable-apis/.
[66] M. Henning, "API design matters," in *Queue*, 2007.
[67] J. Bloch, "How to design a good API and why it matters," in *Companion to the ACM SIGPLAN conference*, 2006.
[68] M. Green and M. Smith, "Developers are not the enemy! the need for usable security apis," in *IEEE Security & Privacy*, To appear.
[69] G. Wurster and P. C. van Oorschot, "The developer is the enemy," in *New Security Paradigms Workshop*, 2008.
[70] D. Cuong Nguyen, Y. Acar, S. Fahl, and M. Backes, "POSTER: Developers Are Users Too: Helping Developers Write Privacy Preserving and Secure (Android) Code," in *Symposium on Usable Privacy and Security*, 2016.