Lessons Learned in Five Years of Conducting Security Studies With Software Developers

Alena Naiakshina
Ruhr University Bochum
alena.naiakshina@rub.de

Anastasia Danilova
University of Bonn
danilova@cs.uni-bonn.de

Matthew Smith
University of Bonn, Fraunhofer FKIE
smith@cs.uni-bonn.de

ABSTRACT

End-user-based research was conducted for over 20 years aiming at supporting end-users with software security issues. Thus ample knowledge exists on how to conduct security studies with end-users. However, critical security incidents in the last decade showed that, like end-users, software developers are usually not security experts and need support with security-critical tasks as well. While this motivated researchers to conduct more security studies with software developers, it also highlighted a lack of methodological knowledge considering security studies with software developers. We derived six recommendations concerning security developer studies' ecological validity and recruitment challenges based on our lessons learned from the last five years of research with software developers.

ACM Reference Format:
Alena Naiakshina, Anastasia Danilova, and Matthew Smith. 2022. Lessons Learned in Five Years of Conducting Security Studies With Software Developers. In *International Workshop on Recruiting Participants for Empirical Software Engineering (RoPES '22), May 17, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Ecological validity issues constitute a significant concern of usable security studies with software developers [5]. It is unclear how knowing they participate in a study or being explicitly instructed to consider security aspects in a task description affects software developers' security behavior. Like end-users, developers might consider security a secondary task and thus might behave differently in real-world circumstances [5, 16]. Further, it is challenging to recruit enough professionals to obtain a significant amount of relevant study data due to time, spread out geographical locations, and financial constraints [4–6, 19, 27, 31]. A common approach in software engineering studies is the recruitment of convenience samples, such as computer science (CS) students (e.g., [4, 17, 20, 25, 26, 29]). While recent work indicates that CS students can be acceptable proxies for professionals in research studies [4, 6, 18, 31], there are still some caveats [31]. To widen the recruitment pool and include non-student participants, it is common for researchers to resort to online studies and recruit participants online (e.g., [6–8, 15, 21, 22, 30, 32]). Diverse recruitment strategies have been used, such as cold-calling programmers on platforms such as Stack Overflow, GitHub, Meet-up groups, etc. or posting open invitations on social media, in forums, newsletters, and events, with the expectation being that participants without programming knowledge will not sign up for the studies [7, 9, 28]. However, since researchers often offer significantly higher compensation than for end-user studies [22–24], there can be an incentive for participants to take part in studies despite having no programming skill. We conducted lab, online, and field studies with students, freelancers, and company developers investigating their security behavior and different study design variables. In the following, we contribute our lessons learned from the last five years of research with software developers.

RoPES '22, May 17, 2022, Pittsburgh, PA, USA

2 RESEARCH IMPLICATIONS

2.1 Security deception

To investigate software developers' security behavior with user-password storage, we conducted a qualitative study with 20 CS students in a laboratory setting [23]. Participants were asked to develop a user registration function for a university social networking platform. To examine whether software developers think about security on their own, half the participants were explicitly prompted to store user passwords securely, while the other half were told that the study was about API usability (non-prompted). The study also investigated whether an application programming interface (API) offering built-in functionalities for secure password storage would help developers produce more secure code than an API requiring developers to choose secure password storage mechanisms manually. Based on their popularity, Spring [2] and JavaServer Faces (JSF) [1] were selected as frameworks for the study. Spring was chosen to represent a more supportive framework with inbuilt functionalities for secure password storage. In contrast, with JSF, developers had to implement secure password storage on their own. None of the participants stored user passwords securely without being prompted in this study. If researchers are testing the usability of security APIs but do not instruct participants to think about security, participants might not use the security features. Our results indicated that security deception might simulate a more realistic environment, as employers or end clients without a background in technology or security rarely ask for secure implementation but rather for software functionality. Researchers might consider including functionality requirements in security tasks to increase the ecological validity of security studies with developers.

Security deception might increase the ecological validity of security studies with developers.

2.2 Qualitative vs. quantitative studies

To further explore methodological requirements, the previous student study was extended to include an additional 20 CS students,

and quantitative analysis was performed [24]. The findings of the qualitative study with students described in Section 2.1, provided already good indications for the results of this follow-up quantitative study. We concluded that qualitative studies might offer valuable insights without recruiting many professionals.

Researchers might choose a qualitative approach rather than a quantitative one if sample sizes are expected to be small.

2.3 Study deception

Since participants from the previous student studies reported that they might store user-passwords securely in a real company, we conducted a follow-up study with freelancers. In the hope of increasing the ecological validity, we used a deception study design wherein freelance developers were hired on Freelancer.com for a regular job using a company front created for the study, instead of openly telling them that they were taking part in a study [22]. After completing the programming task, participants were informed about the research context. In this study, participants' behavior was similar to that of CS students concerning secure password storage practices. While deception in this study was used to ensure that the results would reflect the real work of online freelancers, it can entail additional study design work and negotiations with ethical oversight bodies. Therefore, we replicated the study but announced and ran it as a study on Freelancer.com [11]. Our findings suggested that study deception did not significantly affect this password storage study, and the open recruitment without deception was a viable recruitment method. While the results certainly do not generalize to all developer security studies, it is an essential first indication that freelance developers recruited as part of a study behave similarly to when they are hired for a regular job. We found Freelancer.com to be a suitable source for recruiting enough willing professional developers to work on (study) projects.

Researchers might get ecologically valid results in software developer studies without using study deception.

2.4 Sample comparison

To investigate whether professional developers employed by companies behave the same as CS students and freelance developers, an additional online study was conducted with professional developers from different companies [21]. However, it was not realistic to hire professionals from companies for a small task without revealing the research context of the study. Therefore, participants were informed that the programming task was requested for research purposes. In this study, professional developers employed in companies chose higher security implementations for password storage than CS students or freelancers. However, similar to CS students and freelancers their security behavior varied when they were not prompted for security. In addition, like the students, professional developers working in companies made rather better security choices with Spring than JSF. In summary, it was demonstrated that the findings regarding relative behavior applied to all developer samples. Since the usable security and privacy community is concerned with increasing secure development rather than with which samples perform better, these are promising results for the ecological validity of developer studies conducted with CS students.

Researchers might recruit CS students when they want to investigate security behavior in relative terms.

2.5 Programming vs. code reviewing

The previous programming studies were time consuming and the sample size was not as large as we would have wished. Therefore, researchers often tend to design programming tasks in such a way that software developers only need to solve small and short tasks (e.g., [3, 4, 6]). To provide deeper insights into this research field, we tested code reviewing as a promising methodology for security studies with developers. Instead of asking developers to program a piece of code, we showed them functional code snippets and asked them to write code reviews about the snippets. We conducted an online code reviewing study with 44 freelance developers showing each of them an insecure password storage code snippet [13]. Participants needed less time to complete the code review study compared to the programming studies in [11, 21–24] and we still found similar results concerning security awareness and security prompting. While we do not argue to replace programming tasks with code-reviewing tasks in security developer studies, funding is often limited within academia, and smaller tasks yielding similar effects could enable more future research with developers.

Researchers might consider using code reviews as methodology for security developer studies.

2.6 Screener questions

In previous online studies with programmers, researchers often relied on participants' claims to have programming skills or used programming tasks or knowledge questions to verify these [7, 14, 28]. However, our work showed that designing programming screener questions is not trivial, and we would not recommend using questions without testing them before. We surveyed a total of 249 people to find questions that can be used to filter participants with programming skills. We designed 16 questions and tested them with programmers, non-programmers and under adversarial conditions. We recommend six screener questions for use in online studies based on our evaluation (see [12]). Since the most reliable screeners were also those that took the most time, we extended the pool of screeners and made recommendations on improving the process and introduced time limits allowing us to create more efficient (i.e., quicker but still reliable) screeners (see [10]). Future researchers can use our questions to improve their data quality by screening out participants without any programming skill.

Researchers might use screener questions to ensure participants have programming skills in online studies.

ACKNOWLEDGMENTS

This work was partially funded by the ERC Grant 678341: Frontiers of Usable Security.

REFERENCES

[1] [n.d.]. JavaServer Faces (JSF). Website. Online: https://javaee.github.io/javaserverfaces-spec/; last accessed January 21, 2022.

[2] [n.d.]. Spring. Website. Online: https://spring.io/projects/spring-framework; last accessed January 21, 2022.

[3] Yasemin Acar, Michael Backes, Sascha Fahl, Simson Garfinkel, Doowon Kim, Michelle L Mazurek, and Christian Stransky. 2017. Comparing the Usability of Cryptographic APIs. In *2017 IEEE Symposium on Security and Privacy (SP'17)*. IEEE, 154–171. https://doi.org/10.1109/SP.2017.52

[4] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky. 2016. You Get Where You're Looking for: The Impact of Information Sources on Code Security. In *2016 IEEE Symposium on Security and Privacy (SP'16)*. 289–305. https://doi.org/10.1109/SP.2016.25

[5] Yasemin Acar, Sascha Fahl, and Michelle L Mazurek. 2016. You are not your developer, either: A research agenda for usable security and privacy research beyond end users. In *2016 IEEE Cybersecurity Development (SecDev)*. IEEE, 3–8.

[6] Yasemin Acar, Christian Stransky, Dominik Wermke, Michelle L Mazurek, and Sascha Fahl. 2017. Security Developer Studies with GitHub Users: Exploring a Convenience Sample. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS'17)*. 81–95.

[7] Hala Assal and Sonia Chiasson. 2019. 'Think Secure from the Beginning': A Survey with Software Developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI'19)*. ACM, New York, NY, USA, Article 289, 13 pages. https://doi.org/10.1145/3290605.3300519

[8] Jason Bau, Frank Wang, Elie Bursztein, Patrick Mutchler, and John C Mitchell. 2012. Vulnerability Factors in New Web Applications: Audit Tools, Developer Selection & Languages. *Stanford, Tech. Rep* (2012).

[9] Moritz Beller, Niels Spruit, Diomidis Spinellis, and Andy Zaidman. 2018. On the Dichotomy of Debugging Behavior Among Programmers. In *Proceedings of the 40th International Conference on Software Engineering (ICSE'18)*. 572–583. https://doi.org/10.1145/3180155.3180175

[10] Anastasia Danilova, Stefan Horstmann, Matthew Smith, and Alena Naiakshina. 2022. To appear: Testing Time Limits in Screener Questions for Online Surveys with Programmers. In *2022 IEEE/ACM International Conference on Software Engineering (ICSE)*. IEEE.

[11] Anastasia Danilova, Alena Naiakshina, Johanna Deuter, and Matthew Smith. 2020. Replication: On the Ecological Validity of Online Security Developer Studies: Exploring Deception in a Password-Storage Study with Freelancers. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. 165–183.

[12] Anastasia Danilova, Alena Naiakshina, Stefan Horstmann, and Matthew Smith. 2021. Do you really code? Designing and Evaluating Screening Questions for Online Surveys with Programmers. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 537–548.

[13] Anastasia Danilova, Alena Naiakshina, Anna Rasgauski, and Matthew Smith. 2021. Code Reviewing as Methodology for Online Security Studies with Developers-A Case Study with Freelancers on Password Storage. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. 397–416.

[14] Anastasia Danilova, Alena Naiakshina, and Matthew Smith. 2020. One Size Does Not Fit All: A Grounded Theory and Online Survey Study of Developer Preferences for Security Warning Types. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE'20)*. https://doi.org/10.1145/3377811.3380387

[15] Peter Leo Gorski, Luigi Lo Iacono, Dominik Wermke, Christian Stransky, Sebastian Möller, Yasemin Acar, and Sascha Fahl. 2018. Developers Deserve Security Warnings, Too: On the Effect of Integrated Security Advice on Cryptographic API Misuse. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS'18)*. 265–281.

[16] Matthew Green and Matthew Smith. 2016. Developers are Not the Enemy!: The Need for Usable Security APIs. *IEEE Security & Privacy* 14, 5 (2016), 40–46.

[17] Martin Höst, Björn Regnell, and Claes Wohlin. 2000. Using Students as Subjects-A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. *Empirical Software Engineering* 5, 3 (2000), 201–214.

[18] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. 2017. "I Have No Idea What I'm Doing" - On the Usability of Deploying HTTPS. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1339–1356. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/krombholz

[19] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. 2017. "I Have No Idea What I'm Doing" - On the Usability of Deploying HTTPS. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1339–1356. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/krombholz

[20] Thomas D. LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining Mental Models: A Study of Developer Work Habits. In *Proceedings of the 28th International Conference on Software Engineering* (Shanghai, China) *(ICSE '06)*. ACM, New York, NY, USA, 492–501. https://doi.org/10.1145/1134285.1134355

[21] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, and Matthew Smith. 2020. On Conducting Security Developer Studies with CS Students: Examining a Password-Storage Study with CS Students, Freelancers, and Company Developers. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376791

[22] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, and Matthew Smith. 2019. 'If You Want, I Can Store the Encrypted Password': A Password-Storage Field Study with Freelance Developers. In *Proceedings of the 2019 Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI'19)*. ACM, New York, NY, USA, Article 140, 12 pages. https://doi.org/10.1145/3290605.3300370

[23] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. 2017. Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS'17)*. ACM, New York, NY, USA, 311–328. https://doi.org/10.1145/3133956.3134082

[24] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. 2018. Deception Task Design in Developer Password Studies: Exploring a Student Sample. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS'18)*. USENIX Association, Baltimore, MD, 297–313. https://www.usenix.org/conference/soups2018/presentation/naiakshina

[25] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. 2017. A Stitch in Time: Supporting Android Developers in WritingSecure Code. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. ACM, New York, NY, USA, 1065–1077. https://doi.org/10.1145/3133956.3133977

[26] Iflaah Salman, Ayse Tosun Misirli, and Natalia Juristo. 2015. Are students representatives of professionals in software engineering experiments?. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, IEEE, Florence, Italy, 666–676.

[27] Dag IK Sjoberg, Bente Anda, Erik Arisholm, Tore Dyba, Magne Jorgensen, Amela Karahasanovic, Espen Frimann Koren, and Marek Vokác. 2002. Conducting Realistic Experiments in Software Engineering. In *Proceedings international symposium on empirical software engineering*. IEEE, IEEE Press, Piscataway, NJ, USA, 17–26.

[28] Davide Spadini, Gül Çalikli, and Alberto Bacchelli. 2020. Primers or Reminders? The Effects of Existing Review Comments on Code Review. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE'20)*. https://doi.org/10.1145/3377811.3380385

[29] Mikael Svahnberg, Aybüke Aurum, and Claes Wohlin. 2008. Using Students As Subjects - an Empirical Evaluation. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (Kaiserslautern, Germany) *(ESEM '08)*. ACM, New York, NY, USA, 288–290. https://doi.org/10.1145/1414004.1414055

[30] Chamila Wijayarathna and Nalin A. G. Arachchilage. 2018. Why Johnny Can't Store Passwords Securely? A Usability Evaluation of Bouncycastle Password Hashing. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018* (Christchurch, New Zealand) *(EASE'18)*. Association for Computing Machinery, New York, NY, USA, 205–210. https://doi.org/10.1145/3210459.3210483

[31] Khaled Yakdan, Sergej Dechand, Elmar Gerhards-Padilla, and Matthew Smith. 2016. Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study. In *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, IEEE, San Jose, CA, USA, 158–177.

[32] Aiko Yamashita and Leon Moonen. 2013. Do Developers Care about Code Smells? An Exploratory Survey. In *2013 20th Working Conference on Reverse Engineering (WCRE'13)*. IEEE, 242–251. https://doi.org/10.1109/WCRE.2013.6671299