If You Build It, Will They Come? Developer Recruitment for Security Studies

Nikhil Patnaik, Joseph Hallett, Mohammad Tahaei, Awais Rashid
firstname.lastname@bristol.ac.uk
University of Bristol, UK

ABSTRACT

Recruiting participants for security-focused developer studies is a challenge. We discuss why recruiting for security studies is so challenging, why finding suitable developers is hard, and why the issues faced vary for different research groups. We conclude by suggesting the need for developer test-beds, for all to access, to facilitate the mass study of developers and their security practices.

ACM Reference Format:
Nikhil Patnaik, Joseph Hallett, Mohammad Tahaei, Awais Rashid. 2022. If You Build It, Will They Come? Developer Recruitment for Security Studies. In *Proceedings of 1st International Workshop on Recruiting Participants for Empirical Software Engineering - The 44th International Conference on Software Engineering (ROPES - ICSE 2022).* ACM, New York, NY, USA, 2 pages.

1 INTRODUCTION

To help developers write more secure code, security researchers study how developers engage with the existing security tools and cryptographic APIs available. That said recruiting developers for such studies presents a set of challenges on its own. We reflect on the recruitment challenges of 5 security-focused studies over the last 7 years. We propose a developer test-bed based on the lessons learned from the recruitment challenges discussed.

2 WHAT'S SO SPECIAL ABOUT SECURITY?

Security studies require participants to not only have experience with programming but also general security know-how [1, 3, 4]. However, developers are rarely both skilled programmers and experienced with implementing security. Recruitment becomes even more challenging when considering the developers' expertise. Recruiting from a spectrum of experienced, intermediate, and novice developers allows for an in-depth, unskewed analysis, which reduces threats to validity and may present more interesting findings.

Hallett et al. recruited 138 developers from Prolific to study whether the act of writing a specification for a piece of security sensitive code lead to developers writing more secure code. Not only did most developers fail to store a password safely, but despite developers claiming expertise (and the study platform claiming to recruit developers with expertise) there was also no statistical difference between developers who claimed security expertise and those who had no experience [3]. Acar et al. observed 256 Python developers tasked with performing common cryptographic processes using 5 Python cryptographic APIs [1]. They examined the developers' code for functional correctness and security, and compared the results to the claims of usability made by the cryptographic APIs. Nadi et al. performed 2 surveys with a total of 48 developers to understand the cryptographic tasks they perform and the challenges they face [4]. Nadi et al. recruited developers, not based on their programming competence, but rather the questions they asked on Stack Overflow regarding Java-based cryptographic APIs. Wu et al. ran 19 semi-structured phone interviews with participants, using a combination of standard interview techniques and a diagramming exercise. Wu et al. identified 4 mental models ultimately derived from an abstraction of restrictive access control and symmetric encryption [7].

In an ongoing study, we elicited developers' mental models and ran a code comprehension exercise where developers attempted to explain different implementations of Public Key Cryptography written using different C-based cryptographic APIs. The developers required experience with C programming along with some cryptographic expertise. The study group was comprised of professional programmers and cryptography students. Both Wu et al. [7] and our ongoing study recruited a far lower number of participants compared to the other 3 studies [1, 3, 4]. Recruiting 20 participants required multiple recruitment sources. We also found that each interview was taking around 2 hours to complete. Developers cannot usually provide more than 1 hour of their time, so the interviews did not allow room for long engaging discussions, but instead were rather short intense discussions because the purpose of the study was to elicit the mental model of the developer.

> **Takeaway:** To ensure a range of expertise with ample time to explore developers' ideas is really hard when recruiting.

3 RECRUITMENT POOLS & AVENUES

If a researcher wants to run a study with developers they first need to recruit people who know how to code. This can be more challenging than one might hope. When recruiting participants for our studies through Prolific, we tried specifying knowledge of programming as an inclusion criteria [3, 5]. Despite this we found that many of our participants had never programmed before and so were rejected after starting (or completing) the study. Others had questionable programming knowledge (despite claiming the skill). We found that although we recruited developers based on their familiarity with programming from Prolific, the sample may give an inaccurate representation of how developers as a whole behave, a threat to the ecological validity. This may be because the developers that apply to work through Prolific are of a freelance, independent

ROPES - ICSE 2022, May 21–29, 2022, Pittsburgh, PA, USA

nature. Along with the ecological validity, this issue also effects the demographic of study groups recruited through Prolific. When using the same survey tool with no exclusion criteria, Wu et al. found that roughly a third of the participants they recruited were students and only a third of them had full time jobs [7]. Although Wu et al. did not actively seek to filter by socioeconomic demographics, the unclear experience of the participants threatens the validity of the study. Acar et al. recruited Python developers by sampling Python contributors from popular GitHub repositories, maintaining ecological validity [1].

Our ongoing study on mental models was advertised to lecturers who shared it through the university student boards. We also reached out to other universities who taught C and Cryptography in their syllabus. We shared our advertisement through LinkedIn with C programming and cryptography groups, and chat rooms: another avenue for recruitment. We also utilised connections with industry to recruit suitable software developers. We performed snowball sampling by asking participants if they could pass on the advertisement to others who may be interested in the study [2, 4]. Despite trying through all these different methods, and offering a reasonable financial incentive, we could only recruit 20 participants.

Takeaway: Ensuring participants represent software developers as a whole is really hard no matter the incentives or sources—platform tools pose threats to ecological validity!

4 ETHICAL & INSTITUTIONAL CONCERNS

Several studies have taken the approach of directly contacting developers by scraping email addresses from Stack Overflow and Github commits [1, 4, 6]. Whilst these approaches can potentially offer a large number of participants to recruit (over 50,000 invites were sent in one study with a completion rate of 0.5% [1]), our own experience of these approaches is that many developers find unsolicited emailing unwelcome (even though we use GDPR compliant services to send such invitations) [6] and regard it as spam email, and that the overall response rate is low. Whilst using users' emails for recruitment is prohibited by platforms, it is possible to collect user emails independently, and so we should consider the *ethics* of this method. Mass-scraping requires abusing web technologies to retrieve data unavailable through APIs. If a significant number of the people we are trying to recruit find our recruitment methods invasive perhaps we should rethink our strategies.

If we are not going to recruit developers from mailing lists, perhaps we can recruit through our own universities: after all we train the next generation of developers, can we not study them too? Our experience of this process is that the ability to do so varies greatly at different institutions. In some universities running experiments with students is normal and there are clear and easily available channels to do so. Other universities prevent any researcher with teaching commitments from running studies with department students to ensure that students' grades cannot be influenced by any research studies in which they may optionally take part on. Whilst some universities have mailing lists and channels that can be used to contact and recruit students *en masse*, these mechanisms are not universal and researchers must rely on student societies and the good will of administrators to varying degrees of success.

Takeaway: When devising recruitment strategies, we need to consider the challenges of unsolicited approaches, differing institutional practices, and platform constraints.

5 CONCLUSION

Security-focused studies aim to identify and address issues faced by developers, but are effected by the recruitment challenges we have discussed. Each of these studies independently recruited developers for their studies, resulting in repeated effort and possible participation fatigue for developers. Although the existing recruitment platforms available provide participants, there is a notable limitation regarding the validity of the findings. We need collaboration across research groups to achieve this because current approaches are not scalable across individual universities and we need more developers than any one research group can reach.

We propose a developer test-bed—accessible to all—that works as a dedicated medium between developers and security researchers. The developer test-bed is a pool of developers who join based on incentive programs, like payment contributions to open source projects in which they are involved on. Researchers can use the test-bed to find suitable participants with suitable skills and host their studies in the form of hackathons and user studies, which can engage groups of developers en masse. Instead of individual researchers trying to get developers to come and take part in studies; we can as a research community build developer test-beds to study how programmers work securely in practice. The developers define their skills and experience, which researchers can filter through to recruit suitable participants. To address ethical challenges, researchers and participants will interact through a portal system, instead of spam emails being sent directly using personal email. If we can build a test-bed to study developers then instead of worrying about recruitment issues, researchers will come to study developers, because current recruitment strategies are unsustainable.

ACKNOWLEDGMENTS

This research is supported in part by EPSRC Grant EP/P011799/2 and the National Cyber Security Centre.

REFERENCES

[1] 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)*. IEEE.
[2] Leo A Goodman. 1961. Snowball sampling. *The annals of mathematical statistics* (1961).
[3] Joseph Hallett, Nikhil Patnaik, Benjamin Shreeve, and Awais Rashid. 2021. "Do this! Do that!, And nothing will happen" Do specifications lead to securely stored passwords?. In *ICSE*. IEEE.
[4] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. 2016. Jumping through hoops: Why do Java developers struggle with cryptography APIs?. In *Proceedings of the 38th International Conference on Software Engineering*.
[5] Mohammad Tahaei, Kopo M. Ramokapane, Tianshi Li, Jason I. Hong, and Awais Rashid. 2022. Charting App Developers' Journey Through Privacy Regulation Features in Ad Networks. In *Proceedings on Privacy Enhancing Technologies*. 1–24.
[6] Dirk Van Der Linden, Pauline Anthonysamy, Bashar Nuseibeh, Thein Than Tun, Marian Petre, Mark Levine, John Towse, and Awais Rashid. 2020. Schrödinger's security: opening the box on app developers' security rationale. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 149–160.
[7] Justin Wu and Daniel Zappala. 2018. When is a tree really a truck? Exploring mental models of encryption. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*.