Recruiting Participants With Programming Skills: A Comparison of Four Crowdsourcing Platforms and a CS Student Mailing List

Mohammad Tahaei
mohammad.tahaei@bristol.ac.uk
Department of Computer Science
University of Bristol
United Kingdom

Kami Vaniea
kami.vaniea@ed.ac.uk
School of Informatics
University of Edinburgh
United Kingdom

ABSTRACT

Reliably recruiting participants who have programming skills is an ongoing challenge for empirical studies involving software development technologies, often leading to the use of crowdsourcing platforms and computer science (CS) students. In this work, we use five existing survey instruments to explore the programming skills, privacy and security attitudes, and secure development self-efficacy of CS student participants and participants from four crowdsourcing platforms (Appen, Clickworker, MTurk, and Prolific). We recruited 613 participants who claimed to have programming skills and assessed recruitment channels in regards to costs, quality, programming skills, and privacy/security attitudes. We find that 27% of crowdsourcing participants, 40% of self-reported developers from crowdsourcing participants, and 89% of CS students got all programming skill questions correct. CS students are the most cost-effective recruitment channel and rate themselves lower than crowdsourcing participants in terms of secure development self-efficacy.

CCS CONCEPTS

• **Security and privacy → Software and application security**; **Human and societal aspects of security and privacy**; **Usability in security and privacy**; • **Human-centered computing → Human computer interaction (HCI)**; **HCI design and evaluation methods**; • **Software and its engineering**;

KEYWORDS

recruitment, software developers, crowdsourcing, usable privacy and security, programming, datasets

ACM Reference Format:
Mohammad Tahaei and Kami Vaniea. 2022. Recruiting Participants With Programming Skills: A Comparison of Four Crowdsourcing Platforms and a CS Student Mailing List. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3491102.3501957

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3501957

1 INTRODUCTION

When studying any type of user population it is important to consider how the recruitment channels and methods might impact the resulting research. Different populations of users are well-known to differ culturally, have different biases, and different expectations [5, 6, 53]. Amazon Mechanical Turk workers, for example, are known to be more privacy sensitive than the general population [28]. While other sources have their own limitations, such as populations based in specific countries are likely biased by that country's norms and culture [54]. All these issues should be considered when selecting a recruitment channel and approach to ensure that the resulting research produces outcomes that are externally valid for the main question being asked. In this work, we consider the issue of recruiting people with programming skills with a particular eye towards online studies that may need such a population to test new programming paradigms, tools, or programming experience and attitudes.

The various areas touching on software development such as end-user programming [30], developer-centered security and privacy [62, 63, 66], and developer tool support have all been growing in focus due to the key role developers play in society and the need to provide them with effective support. There has also been a strong push to make software development easier and more accessible to a wider set of users leading to a focus on how to improve usability for the different groups that engage in programming [30, 38] (e.g., by making privacy and security technologies such as APIs and notifications directed to developers usable [63]). However finding people with programming skills to study can be non-trivial since it is a skill that takes effort to learn and people who are highly skilled at it are often paid well by companies, which can make them more challenging to recruit. To recruit a broad range of participants with programming skills some researchers have turned to crowdsourcing platforms [4, 25, 59, 68]. However, the ability to generalize the results to a larger population or justify the choice of crowdsourcing platform are often discussed as a limitation. Computer science students have also been a long standing population for such studies, however, the discussions about the external validity of results involving this population has a long history as well [15, 17, 31, 40, 51, 61].

As an alternative solution, some researchers started to use websites that developers use to learn coding or upload code (e.g., GitHub and Google Play). While developers' contact information can be harvested from these services, it is against the terms of services of these platforms and it is also not sustainable solution. Even if the terms of services was not an issue, contacting users of these

platforms multiple times a year by several researchers will likely feel like spamming to developers and begin to feel overwhelming.

We aim to add context to the choice of recruitment channels to aid researchers when recruiting participants with programming skills by looking at crowdsourcing platforms and computer science students in the context of existing survey instruments. Our research questions were:

RQ1: Which recruitment channels are suitable for recruiting participants with programming skills?

RQ2: Which self-reported information correlates well with passing all programming screening questions proposed by Danilova et al. [10]?

RQ3: How do participants with programming skills vary across recruitment channels in terms of privacy and security attitudes as well as secure development self-efficacy?

We recruited 613 participants from five recruitment channels (four crowdsourcing platforms and an University CS student mailing list) who expressed having computer programming skills and asked them to answer questions from five survey instruments designed by prior work to measure programming skills and experiences [10, 16], secure programming self-efficacy [68], user privacy concerns [34], and general computer security attitudes [14].

We find that recruiting CS students from our University's mailing list resulted in the highest data quality in terms of programming skills (highest), costs (lowest), number of duplicates (low), and passing attention check questions (high) compared to all tested crowdsourcing platforms. Among crowdsourcing platforms, we find that Prolific generated a higher number of participants who could pass basic programming questions and is more cost-effective compared to Appen, Clickworker, and MTurk. We suggest future researchers to consider using CS students for studies that require a programming skilled population and recommend crowdsourcing platforms to ask specific questions such as experience with object-oriented programming and years of experience in software development instead of asking a generic question like "Do you have programming skills?" as the generic question can result in broad interpretation of "programming skills" by a crowdworker audience. These adjustments would greatly assist researchers in finding populations that better match their research needs.

2 RELATED WORK

Our work spans across the literature in crowdsourcing services and empirical studies with developers. Below, we cover both angles.

2.1 Crowdsourcing Platforms

Crowdsourcing platforms enable researchers to have access to a large pool of participants without the need to go through the traditional fliers and recruitment processes. Examples of tasks that the HCI community employs crowd workers for are labeling data [50] and eliciting users privacy concerns regarding online messaging tools [26]. Other studies measure the quality of work in crowdsourcing platforms such as MTurk, Prolific, and CrowdFlower [44, 45]. In a 2017 study [44], MTurkers were found to be more experts in completing online tasks compared to Prolific and CrowdFlower participants; while CrowdFlower may result in higher response rate, more participants failed attention check questions compared to the other two platforms; overall, Prolific and MTurk participants produced higher quality data compared to CrowdFlower.

2.2 Empirical Studies with Developers

Recruiting participants with programming skills have been a growing field to improve developer experience through human-computer interaction methods such as field studies, surveys, and interviews [38, 39, 56]. It includes testing developer experience of programming languages [67], building new programming tools/APIs/plugins [39, 69], and understanding how developers seek information [32].

Developer-centered privacy and security—as a subset of empirical studies with developers with a specific focus—looks at how developers interact with privacy and security technologies directed to developers either in the code level or in graphical interfaces [63] such as security warnings in APIs [11, 22] and static analysis tools [65] as well as graphical privacy interfaces in ad networks [59, 64]. CS students are sometimes used a convenience sample for developer studies [31]; however, there are comments about validity of the results in a broader context [15, 17, 40]. Efforts to compare CS students, freelancers, and professional developers who work in companies show that in terms of secure coding skills there are similarities between the three groups and CS students as a convenient sample may produce high quality data [40, 41, 43]. Although, professional company developers may perform better in secure coding tasks perhaps because of the training and experience they receive at work [40].

In summary, a limitation and critic on empirical studies with developers is often finding the right population and validity of the results [31]. It is also notable that recruiting developers have been reported as a difficulty in many studies [25, 31, 40, 57, 65, 68, 70]. Our work contributes in this line of research by comparing various recruitment channels with suggested survey instruments in the literature with regard to participants' secure development self-efficacy (i.e., "belief in one's ability to successfully perform a task—which correlates with actual skill in other contexts" [68, p. 1]) as well as general privacy and security attitudes. Such approach enables future researchers to compare their results with our dataset and results because we used published survey instruments rather than a custom built instrument. We also make recommendations about costs, quality, and programming skills of each recruitment channel.

3 METHOD

We distributed five survey instruments from prior research on five recruitment channels to understand the similarities and differences between these channels and the communities they draw from, as well as the quality of data and cost-effectiveness of recruiting from each channel.

We aimed to build a ground for future researchers who want to recruit participants with programming skills. Our research questions were (RQ1) Which recruitment channels are suitable for recruiting participants with programming skills? (RQ2) Which self-reported information correlates well with passing all programming screening questions proposed by Danilova et al. [10]? And, (RQ3) How do participants with programming skills vary across recruitment channels in terms of privacy and security attitudes as well as secure development self-efficacy?

RQ1's goal is to help future researchers find the right channel for recruiting participants with programming skills. RQ2 would help to have a smaller set of questions and potentially rely on a set of self-reported or demographics questions to find participants with programming skills instead of (or in combination with) asking a set of programming questions. RQ3's results would assist researchers in developer-centered privacy and security to understand where to look for participants with certain privacy and security attitudes as well as secure development self-efficacy. The study was conducted in accordance with the ethics procedures of our institution.

3.1 Recruitment Channels

We used Appen, Clickworker, MTurk, Prolific, and our local mailing list for computer science students to recruit our participants.[1] Our recruitment started in July 9, 2021 and ended in August 31, 2021. We also provide a reflection of our experience with these channel in Section 5.4.

Appen: Formerly known as CrowdFlower is a platform for running micro tasks. It is current mission focuses on building training data for artificial intelligence systems. CrowdFlower was used by researchers in the usable security and privacy community to run studies to elicit users' perceptions around privacy [1] and security [37] technologies. Appen claims to have over one million workers around the world [2].

Clickworker: Focuses on training data sets, however it has been used for running surveys in usable security and privacy studies [33, 36] and finding developers [10]. Clickworker claims to have over 2.8 million workers around the world [9].

MTurk: A classic crowdsourcing platform that has been widely used for running surveys and creating training datasets. It is estimated in 2019 that the MTurk's population was over 250,000 [49]. MTurk is also a common recruitment channel for usable privacy and security studies (e.g., [26, 52]).

Prolific: Advertises itself as a tool for recruiting participants for market, behavioral research, and user studies. It has over 150,000 participants [46]. Prolific has been used in general usable security and privacy literature [19, 71] and also studies directed to developers [59, 65, 68].

Computer science students: As it is often challenging to find developers for empirical studies, CS students have been used as proxy to understand developers' attitudes and perceptions towards privacy and security technologies directed to developers [40, 42, 61, 63]. Therefore, we decided to recruit from our local CS students through an internal mailing list.

We decided to use the above recruitment channels as they have been used in prior research. Although, harvesting developers' emails from GitHub had been used in prior research to recruit developers, we decided not to use this channel because it is against GitHub's terms of services and privacy policy to collect users' emails [20, 21]. Use of social networks such as LinkedIn was not also suitable for running a survey as it was used previously for recruiting smaller samples (e.g., for an interview study [60]) and may not had been a suitable channel for a survey-based study that requires a larger sample compared to an interview study. We acknowledge that there might be other places to find developers, therefore, we call future researcher to try other recruitment strategies and compare their results with our findings.

[1]Disclaimer: we are not funded and associated with any of these platforms. This was a purely academic project funded by our institute which is a public research university.

3.2 Survey Instruments

We first ran a short screening survey described below and then invited participants who passed our criteria to the main survey. Both surveys were implemented on Qualtrics [48]. We set aside a budget of $1200 for each recruitment channel and deployed surveys on each channel until we reached our budget limit (MTurk and Prolific), or reached a point where no more participants were taking our survey (Appen and Clickworker), as determined by no new participants in at least 3 days. All answer options (Likert items and multiple-choice options) in the both surveys were randomized.

3.2.1 Screening Survey. We were interested in participants who potentially work in a software development related role as these are often the target population for studies that require participants with programming skills. Therefore, we set four criteria to satisfy our requirement: (1) employment status must be full-time, (2) they must not be students, as we covered student participants from a separate distribution channel (CS students mailing list), (3) must have programming skills, and (4) they must be fluent in English as our survey was in English. The other reason for these four criteria was that Prolific's provided screening criteria included all these points allowing for easy filtering without a separate screening survey; Prolific is the only channel to offer such a service. Hence, we did not need to run a separate survey on Prolific. But for the other crowdsourcing channels, Appen, Clickworker, and MTurk, we first sent out the four-question survey, and sent out the main survey to the selected participants who passed our criteria.

We did not run a screening survey with the CS students because it is reasonable to assume they have some as everyone on the list would have completed at minimum one year of CS education taught in English, and also because of the friction it would cause leading to lower participation.

Participants received $0.21 for the screening survey (in accordance with minimum wage in our institute's home country). The survey is included in the Supplementary Materials.

3.2.2 Main Survey. The main survey consisted of a randomized order of the five surveys suggest in prior work to assess participants' programming experience, privacy and security attitudes, as well as secure development self-efficacy. Instruments for measuring programming experience for all types of developers are not yet fully developed as "programming experience" is a broad term, developers come from a wide range of skills and backgrounds, and measuring their experience still requires further research [31]. Therefore, we decided to include two surveys (PROGEX and REALCODE) to cover a broad set of questions.

PROGEX "Measuring programming experience" [16]: Built to assess programming experiences, the original study recruited students. Three questions were found particularly correlated with students' ability to find the correct answers to programming tasks were (1) self-reported programming experience (scale 0–10), (2) how participant compare their programming skills with their classmates (scale 1–5), and (3)

experience with object-oriented programming (scale 1–5). We were able to access the original dataset as it was available online [8].

REALCODE "Do you Really Code? Designing and Evaluating Screening Questions for Online Surveys with Programmers" [10]: Built to assess programming skills of participants which was also used for screening crowdsourcing workers (Clickworker and Qualtrics recruitment panel). Despite being a recent publication, we find this survey closely related to our work as it aims facilitate the recruitment progress from crowdsourcing services by building a short survey to assess subjects' programming skills. We used five multiple-choice programming questions (note that the survey consisted of 16 questions in total and we included five recommended questions) that asked about (1) frequent visited website as aid for programming, (2) description of a compiler's function, (3) description of a recursive function, (4) value of a Boolean value, and (5) parameter of the function in a sample code) as the basis for having basic programming skills. As suggested, the first four questions were timed (30-60 seconds), and the fifth question was not timed. As opposed to other surveys these set of questions are not self-reported and a ground-truth group of developers in the original study got all these questions correct, while other participants did not. We contacted the authors and they kindly agreed to share the original dataset with confidentiality agreements.

SSDSES "Building and Validating a Scale for Secure Software Development Self-Efficacy" [68]: Built to assess developers secure development self-efficacy. The original study recruited from various channels (e.g., LinkedIn, Prolific, and personal contacts). We included this survey instrument because we were also interested to find out about security skills of our subjects as recruiting developers for security-related studies is also a known challenge [25, 40, 63, 65, 68]. This survey enabled us to provide insights into secure programming self-efficacy in addition to programming skills. We contacted the authors for the dataset but due to data sharing constraints, we were not able to access it.

SEBIS "Scaling the Security Wall: Developing a Security Behavior Intentions Scale (SeBIS)" [14]: Built to assess general security behaviors, directed to all types of users. We included this survey because it has been used extensively by literature in the past five years (e.g., [13, 23, 24]) and could provide insights into general security behavior of our subjects. We contacted the authors and they kindly agreed to share the original dataset.

IUIPC "Internet Users' Information Privacy Concerns (IUIPC): The Construct, the Scale, and a Causal Model" [34]: A classic survey built to assess general privacy concerns, directed to all types of users. We included this survey to understand our participants' general privacy concerns and provided a basis for comparing our sample with other publications. We contacted the authors for the dataset but due to data sharing constraints, we were not able to access it.

The survey ended with a set of demographics questions such as primary role in a software team, age, gender, and latest employment status. We also included two attention check questions among the Likert items. Crowdsourcing participants received $3.09 for their time (in accordance with minimum wage in our institute's home country) and CS students received a $62.37 gift card per 20 participants. Other than a chance to enter a raffle for a gift card, there were no other incentives offered to the students. We decided to offer students a raffle-based gift card instead of paying them individually because of the payment difficulties that might have occurred with paying a large number of participants without a platform like a crowdsourcing platform that manages payments. The survey is included in the Supplementary Materials.

3.3 Analysis

In addition to our main RQs, we were also interested in descriptive statistics such as costs, quality of data (duplicates and passing attention questions), and demographics of each recruitment channel. We report these descriptive exploratory findings in the results section after describing the collected data in Sections 4.2 and 4.3.

For RQ1 and RQ2, we run regression analyses to be able to make suggestions for future research. For RQ1 and RQ2, unless otherwise noted, regression analyses were conducted in R using the `glm` function [18] with the binomial family (`logit` (logistic regression) as the link function). The model took a binary value as output for whether a participant got all the five REALCODE questions correct (coded as 1) or not (coded as 0); and the input was a set of self-reported programming experiences, demographics, and recruitment channels. We removed Appen from the model as it had no participants who passed all the programming questions and the number of participants in the main survey was small as well (N=9). Sections 4.4 and 4.5 are based on the results of these model.

For RQ3, we did an exploratory descriptive analysis. We included participants who got all the five REALCODE questions correct because we were interested to understand how recruitment channels vary in terms of privacy and security for programming skilled participants (means that Appen was excluded again because no participants from Appen got all the programming questions correct). Section 4.6 is based on these findings.

3.4 Limitations

Out of the six recommended questions by Danilova et al. [10], we picked five questions. We decided to include one code comprehension question instead of the two recommended code comprehension questions because of time constraints and reducing the load on participants. The original study does not require all questions to be included and the choice of which questions to include is left to future researchers. Also, the programmers in the original study answered all the five questions that we selected correctly, giving us a ground to compare our results with a set of questions the programmers answered 100% correctly. Future work may look at including the other questions and comparing the results with our findings and the original study.

We screened participants based on four criteria, but there might be other criteria that could better filter out participants that are not suitable for our target population. Using a screening question such as primary role as a software developer may have resulted in a sample with higher programming skills; however, we were

not aware of existing studies that have looked at these issues and our study would be the answer to this limitation. We were also interested to find out if it is possible to use screening criteria on Prolific instead of running an extra screening survey, therefore, included questions that are identical to the screening criteria in Prolific to test their usefulness for future studies which could reduce recruitment cost and effort.

Our results show that CS students have a higher programming skill with regards to the tested survey instrument, but this must be taken with a caveat that we recruited CS students from one CS department in a single European University. The University is a public research university and is among top 50 universities for CS programs in World University Rankings 2021 [12]. Therefore, other universities may have a different experience recruiting from their CS mailing lists. We compared self-reported programming experience of our sample with PROGEX's sample which was conducted in 2012 with undergraduate students [16] and found that our students considered themselves having a higher programming experience compared to PROGEX's student sample (μ = 5.74, σ = 1.55 vs. μ = 4.63, σ = 1.81, see Section 4.4.2 for details). One explanation might be that our mailing list includes postgraduate students as well and PROGEX's sample included only undergraduate students. However, our results are still consistent with prior findings that CS students may be a valid recruitment channel for participants with programming skills [31], e.g., for secure programming studies [40, 41, 43].

We focus on one survey instrument (REALCODE) as our main output variable and consider a programming skilled participant who passes all the five REALCODE programming questions correct. However, we acknowledge that programming is a broad term and developers also have a diverse range of skills, experience levels, and backgrounds. Using REALCODE as our basis means that we measured participants basic understanding of programming (Table 6 in the Appendix) as the questions in this instrument are fundamental programming questions (e.g., description of a compiler and value of a Boolean variable). Our intention was to first find an approach to screen participants who potentially have no programming knowledge and experience, and second compare various recruitment channels to make suggestions for future research which we believe the results would make tangible recommendations for the community. Future research may run experiments with other sets of questions to assess capabilities of various survey instruments in screening participants with programming skills.

Moreover, our study was conducted during the COVID-19 pandemic when many businesses were closed or working remotely. Such shifts in the labor market may have influenced crowdsourcing platforms as well. Arechar and Rand [3] looked at 23 studies with crowd workers in 2020 (during pandemic) and showed that MTurk has become more diverse in terms political views, ethnicity, and experience with conducting tasks which means that the population is more diverse when we ran our experiment compared to a pre-pandemic population. However, they also find that participants have become less attentive. Based on these findings, there might be a chance that the high number of duplicate responses and participants who failed attention check questions from crowdsourcing platforms (Section 4.3) may be related to the pandemic, though, we do not have the data to either confirm or reject this idea. Future research may replicate a study that was done pre-pandemic to compare the results with current crowd workers to investigate the differences and similarities.

4 FINDINGS

We report our collected data in Section 4.1, then describe our participants' demographics in Section 4.2, and discuss the quality of responses in Section 4.3. Sections 4.4, 4.5, and 4.6 present our findings to our RQs.

4.1 Collected Data

4.1.1 Screening Survey. 3,990 participants from Appen, Clickworker, and MTurk, completed the screening survey on Qualtrics, out of which 563 (14.1%) were removed because they were flagged as a duplicate by Qualtrics' `Q_RelevantIDDuplicate` (i.e., a Boolean value and if True it means that the response is likely to be a duplicate: "This technology checks if the respondent is cheating by taking the survey multiple times or whether a survey taker is fraudulent by analyzing a user's browser, operating system, and location to provide a fraud score" [47]). We invited 789 (19.8%) participants to the main survey who passed our screening criteria (Section 3.2.1). In total, we spent $1,012.61 on the screening survey (Table 1).

4.1.2 Main Survey. 714 participant (all crowdsourcing + CS students) completed the main survey out of which 636 (89.1%) passed both attention questions. We excluded a further 23 (3.6%) of the these participants because they were duplicate responses flagged by Qualtrics. Our final set for the main survey included 613 responses. The rest of the paper is based on the results of the main survey (N=613). On average the survey took 18.3 minutes (SD = 82.7 minutes, potentially some participants left the survey open on their browser and did not fill out the survey all at once). In total, we spent $2,801.88 on the main survey (Table 1). The anonymized dataset for the main survey is publicly available online for future references and potential replication studies (10.7488/ds/3184).

4.2 Demographics

In our sample of 613 participants, 71.8% reported being male which falls between the the proportion of male software developers (over 90% [58]) and the general population in crowdsourcing platforms (45-56% are male [35, 44]). Participants reported an average of 5.5 years of experience in software development (SD = 9.1), and an average age of 32.6 years (SD = 6.3), 63.1% work full-time, 35.9% have a primary role as a software developer, 19.4% do not have a CS-related job, and 15.5% work in a management role in a software team. Most participants come from Europe (54.8%, note that CS students were recruited from a European University mailing list), North America (27.7%), and Asia (10.8%). In the Appendix, Table 4 shows a summary of participants' demographics and Table 5 shows a summary of answers to all five survey instruments per recruitment channel.

4.3 Response Quality and Costs

4.3.1 Response ality. Table 1 shows a summary of responses per recruitment channel. In the screening survey, we find that

Table 1: Summary of recruitment across channels. Costs were converted to USD using xe.com on August 31, 2021. Percentages were calculated with "completed on Qualtrics" as the denominator, unless stated otherwise. Payment for the surveys were screening: $0.21 and main $3.09. CS students received a $62.37 gift card per 20 participants. Payments were in accordance with minimum wage in our institute's home country.

	Appen	Clickworker	MTurk	Prolific	CS Students	Total
Screening survey						
Requested	2,500	1,610	1,933	Used Prolific's screening (Eligible Participants: 7,797 of 262,334)	Wasn't screened (Students on the mailing list: 2,728)	6,043
Completed on platform	1,684	1,050	1,225			3,959
Completed on Qualtrics	1,680	1,082	1,228			3,990
Duplicates	512 (30.5%)	12 (1.1%)	39 (3.2%)			563 (14.1%)
Passed, invited to main	50 (3%)	132 (12.2%)	265 (21.6%)			789 (19.8%)
Cost	$358.48	$297.06	$357.07			$1,012.61
Cost per invitation	$7.17	$2.25	$1.35			-
Main survey						
Completed on platform	21	56	217	389	-	683
Completed on Qualtrics	16	58	219	341	80	714
Passed both attentions	9 (56.3%)	38 (65.5%)	189 (86.3%)	325 (95.3)	75 (93.8%)	636 (89.1%)
Duplicates (of passed att.)	0	0	22 (10%)	0	1 (1.3%)	23 (3.6%)
Final set	9 (56.3%)	38 (65.5%)	167 (76.3%)	325 (100%)	74 (98.7%)	613 (85.9%)
Cost	$56.91	$210.62	$928.76	$1,357.66	$247.93	$2,801.88
Cost per response	$6.32	$5.54	$5.56	$4.18	$3.35	$4.57
Total						
Population size	9	38	167	325	74	613
Cost Screen + Main	$415.39	$507.68	$1,285.83	$1,357.66	$247.93	$3,814.49
Cost per valid response	$46.15	$23.53	$9.25	$4.18	$3.35	-
All passed REALCODE						
Pass all five programming questions	0 (0%)	24 (63.2%)	14 (8.4%)	108 (33.2%)	66 (89.2%)	212 (34.6%)
Cost per programming skilled participant	-	$21.15	$91.85	$12.57	$3.76	-

Appen created a large number of duplicates, potentially because the completion validation is limited to a reused code (not randomized per participant). Duplicates for Clickworker and MTurk were under 4%.

In the main survey, Prolific and CS students passed both attention questions the most often with over 93% passing, MTurk was second with 86.3%, Clickworker was the fourth with 65.5%, and Appen was the last with 56.3%. Including only the participants who passed the attention questions still resulted in duplicated responses in MTurk (10%) and 1 (1.3%) duplicate response from the CS students. Appen and Clickworker had no duplicates in the participants who passed the attention questions.

Overall, our findings here are consistent with prior work on comparing Appen (known at the time as CrowdFlower), MTurk, and Prolific [44]. Prolific produced higher quality data in terms of duplicates and passing attention questions compared Appen as well as fewer duplicates and failed attention participants compared to MTurk (though, Peer et al. [44] found that MTurk is comparable to Prolific). Appen had the highest number of duplicates in the screening survey which is consistent with findings of Peer et al. [45].

4.3.2 Costs. We find that CS students are the most cost-effective channel for recruiting participants with programming skills. We were able to recruit 66 participants with an average cost of $3.76 per participant (total cost: $247.93). Prolific also created a sample size of 108 participants with programming skills, however, each response cost $12.57 (total cost: $1357.66). It is notable that a separate screening survey was not needed on Prolific (used screening criteria on Prolific that matched with our screening survey with no additional costs) and CS students which reduced costs. Clickworker and MTurk were overall expensive and may not be the most cost-effective channels if a programming skilled sample is required. Appen produced zero responses with programming skills, hence,

we cannot recommend it for this type of recruitment and restrict our analyses to the other channels.

4.4 Where to Find Participants With Programming Skills?

We used the REALCODE and PROGEX survey instruments to assess basic knowledge of programming and self-reported programming experience. Here we look at both instruments separately by channel and compared to what was found by prior work. We also look at how they compare to each other.

4.4.1 REALCODE. Unlike the other survey instruments which use self-report for assessment, the REALCODE instrument asks participants five basic programming questions aimed at assessing if they have experience with writing code. The questions asked are multiple choice and focus on common programming-related experiences such as what values go in binary variables, or what website is a common place to find solutions to programming problems (Stack Overflow). In the original work by Danilova et al. [10], 100% of real programmers answered all five questions correctly and 2% of non-programmers answered all five questions correctly.

In total, 212 (34.6%) of participants correctly answered all the five programming questions. CS students produced the highest percentage (66, 89.2%), followed by Clickworker (24, 63.2%), Prolific (108, 33.2%), and MTurk (14, 8.4%). None of the Appen participants answered all five questions correctly (hence, we cannot recommend it for this type of recruitment), which is at odds with their self-reported years of experience in software development (on average 11.78 years). Appendix Table 6 shows the number of correct answers per channel for each of these five questions.

Regression model. We first built a generalized linear (logit) regression model predicting completely correct (5/5) REALCODE answers as the outcome, with channel as a categorical predictor (Table 2). We omitted other covariates in this model as our focus here is on the marginal effect of the channel, rather than questions of whether channels' participants are different in ways that cannot be explained by other covariates. We find that CS students are 16.6 times (p Ÿ .001) and Clickworkers are 3.4 times (p Ÿ .001) more likely to pass all question compared to Prolific participants. For MTurk the odds of getting all the programming questions correct is 82% (p Ÿ .001) lower compared to Prolific participants.

4.4.2 PROGEX. Looking at self-reported experience around programming using the three recommended questions from PROGEX (Appendix Tables 5 and 7), we can see a variation across different channels in terms of self-reported programming experience (10-point scale) with MTurk (μ = 7.2, σ = 2.06) more self-confident in their programming experience than Clickworker (μ = 6.3, σ = 2.04), CS students (μ = 5.7, σ = 1.55), and Prolific (μ = 5.2, σ = 2.37). Questions about experience compared to classmates and familiarity with object-oriented programming showed a similar pattern, with MTurk showing high confidence and Prolific showing the lowest self-confidence.

We also compared our CS student sample against the student sample used by Feigenspan et al. [16] in the work that introduces PROGEX. The self-reported programming experience of our CS students (μ = 5.74, σ = 1.55) was statistically significantly higher than the CS students studied in Feigenspan et al. [16] (μ = 4.63, σ = 1.81), $t(136)$= −3.95, p Ÿ .001. We hypothesize that this difference likely comes from sampling students at different universities. Our mailing list also includes postgraduate students, but Feigenspan et al. [16]'s sample only included undergraduate students. In short, our sample of CS students reported lower programming experience compared to crowdsourcing platforms but higher programming experience compared to their classmates from prior work.

Table 2: Generalized linear model regression. Outcome variable is a binary variable that represents whether a participant got all the five REALCODE questions correct (coded as 1) or not (coded as 0). OR: odds ratios, CI: confidence intervals, Tjur's R2: .267, No. observations: 604 (excludes Appen), $*p$ Ÿ .05, $**p$ Ÿ .01, $***p$ Ÿ .001.

Independent Variables	ORs	CI (95%)	p–value
Recruitment channel			
Prolific	*Reference*		
Clickworker	3.44	1.73 – 7.08	.001***
MTurk	0.18	0.10 – 0.32	<.001***
CS students	16.58	8.12 – 38.56	<.001***
(Intercept)	0.50	0.39 – 0.63	<.001***

4.4.3 REALCODE vs. PROGEX. Taking a brief look at how the most general question in PROGEX—their self-assessed programming experience—in relation to if they answered all the REALCODE answers correctly or not, we reassuringly see that the self-estimation of those who passed all REALCODE questions (σ = 6.38, μ = 1.85) is statistically significantly higher than those who did not (σ = 5.62, μ = 2.53), $t(550)$= 4.23, p Ÿ .001.

We treated the Likert data as an interval variable [7] and ran t-tests on them, which is common practice in the CHI community [29]. We also used Bonferroni correction for the t-tests because they were not part of main study's objectives.

4.5 Demographics and Programming Skills

We built another generalized linear (logit) regression model also predicting completely correct (5/5) REALCODE answers as the outcome, with categorical predictors of channel, the most recent primary job role, years of experience in software development, and PROGEX to the model and found that all channels still differ from Prolific in terms of participants who got all the programming questions correct; however, the coefficients change (Table 3). CS students are 26 times (p Ÿ .0001) and Clickworkers are 2.2 times (p = .049) more likely to pass all questions compared to Prolific. It is possible that the addition of years of experience in software development explained some of the variation since CS students had fewer years of experience with 71.6% having three years or less compared to 50% of crowdsourcing participants having three years or less. The odds of getting all the questions right is 90% (p Ÿ .001) lower compared to Prolific for MTurk.

Table 3: Generalized linear model regression that includes self-reported demographics and programming experiences. Outcome variable is a binary variable that represents whether a participant got all the five REALCODE questions correct (coded as 1) or not (coded as 0). OR: odds ratios, CI: confidence intervals, Tjur's R2: .417, No. observations: 604 (excludes Appen), $*p$ Ÿ .05, $**p$ Ÿ .01, $***p$ Ÿ .001.

Independent Variables	ORs	CI (95%)	*p*–value
Years of experience in software development	1.08	1.04 – 1.12	<.001***
PROGEX			
Rating of programming experience	1.07	0.92 – 1.24	.416
Programming experience compared to class/work mates	1.08	0.79 – 1.48	.636
Experience in object-oriented programming	1.58	1.22 – 2.06	.001***
Recruitment channel			
Prolific	*Reference*		
Clickworker	2.23	1.01 – 5.04	.049*
MTurk	0.10	0.05 – 0.19	<.001***
CS students	25.99	10.22 – 81.51	<.001***
Primary role in a software team			
Not CS-related	*Reference*		
Developer	2.44	1.25 – 4.87	.010**
Designer	0.94	0.28 – 2.78	.911
Manager	1.00	0.44 – 2.25	.994
Privacy/Security	1.38	0.25 – 6.68	.692
Tester	1.33	0.48 – 3.51	.568
Unemployed	0.80	0.21 – 2.64	.724
Other	1.59	0.55 – 4.39	.380
(Intercept)	0.02	0.01 – 0.07	<.001***

We also find that asking participants about their years of experience in software development is significantly correlated with getting all programming questions correct. One unit increase in years of experience in software development would increase the odds of getting all questions correct by 8% (p Ÿ .0001).

When it comes to primary role in a software team, we find that self-reporting as software developer is significantly correlated with answering all programming questions correct. Participants who reported that they are developer are 2.4 times (p = .01) more likely to get all programming questions corrected compared to participants who do not work in a computer science related role.

The correlation for two questions from PROGEX, ratings of programming experience and comparing programming experience with class/work mates, were not significant and conclusive. However, one unit increase in experience with object-oriented programming is correlated with 58% (p = .001) increase in the odds of getting all programming question correct.

4.6 Privacy and Security

We were interested in facilitating the recruitment process for developer-centered privacy and security studies as well. RQ1 and RQ2 could assist researchers in screening participants with programming skills, however, still it is not clear whether participants from different channels vary in regards to privacy and security attitudes and secure development self-efficacy. To answer RQ3, we explore the differences between recruitment channels in relation to the three privacy and security related survey instruments. We present boxplots (Figures 1, 2, and 3) of the three survey instruments broken down by channel and discuss observations. The results in this section are based on the participants who passed all five REALCODE questions.

4.6.1 SSDSES. This survey instrument focuses on secure software development self-efficacy and is made up of two components: security communication and vulnerability identification. The secure communication component focuses on the ability to communicate security issues effectively with others and to upper level management. The vulnerability component focuses on the ability to identify various threats to systems and code as well as perform a risk analysis. Across all recruitment channels, participants generally rated their communication skills for security higher that their skills in identifying and mitigating security vulnerabilities (Figure 1). In the original SSDSES paper, participants also often rated their identification and mitigation skills lower than their communication skills [68]. When looking at the differences between channels, we see that MTurkers rated themselves higher than all other channels on average and also had the most similar scores between the components with an average difference of 0.35 between the components compared to 0.58 for the other channels. Crowdsourcing participants also rated themselves higher than CS students. CS students had the lowest rating for vulnerability identification and mitigation which we hypothesize comes from knowing that how difficult doing some of these tasks might be (e.g., identify potential attack vectors and mimic potential threats).

4.6.2 SEBIS. This instrument focuses on general security behaviors and is intended for a general audience rather than specifically developers. It has four components: device, awareness, password, and updating. In terms of general security attitudes, we find that device securement (e.g., locking devices and using PINs) had the highest rating among all participants which is not inline with the original SEBIS dataset collected from MTurk in 2015 [14]. One explanation could be different time periods and general secure attitude for devices have increased over time. The other explanation might be that our population is presumably have some programming skills and their understanding of security and information technology might be higher. Comparing CS students with crowdsourcing participants shows a lower attitude towards updating (e.g., keeping systems up-to-date), password generation (e.g., generating stronger passwords), and proactive awareness (e.g., checking for links before clicking) which is somehow surprising as CS students are exposed to several privacy and security topics at the University.

4.6.3 IUIPC. This instrument is also intended to be used with a general audience and focuses on attitudes users have towards how privacy should be enacted online. It has three components: control, awareness, and collection. General privacy attitudes of our CS students were overall higher than other recruitment channels which is in odds with their general security behaviors in SEBIS (Section 4.6.2). In all crowdsourcing participants, we observed that they rate the awareness component (e.g., awareness about how personal data is being used by companies) higher than control (e.g., control of personal information is at center of user privacy) and collection (e.g., how data collection bothers them). Such a pattern is inline with the original data set that was collected in 2004 [14]; higher awareness compared to control and collection (awareness:

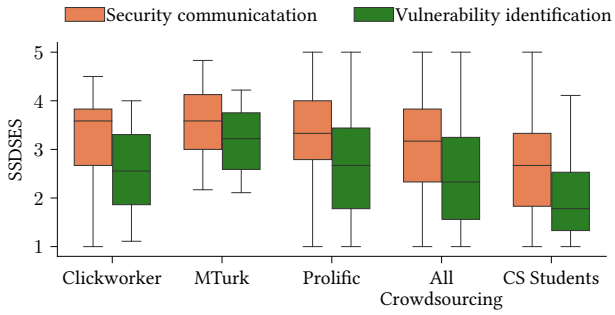

Figure 1: Summary of secure development self-efficacy for only participants who passed REALCODE (scale 1–5). Security communication: being able to discuss security with others; Vulnerability identification: being able to identify and mitigate security vulnerabilities.

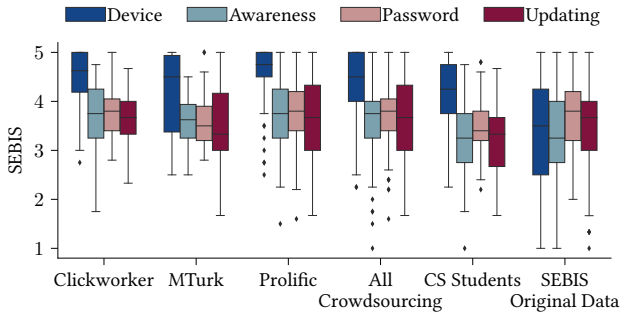

Figure 2: Summary of security behavior intentions for only participants who passed REALCODE (scale 1–5). Last column (SEBIS Original Data) is from the original SEBIS dataset recruited from MTurk in 2015 [14].

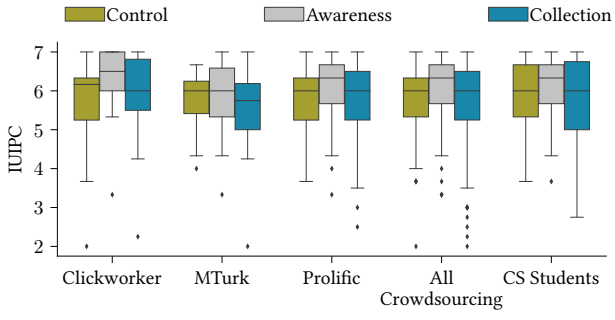

Figure 3: Summary of internet users' information privacy concerns for only participants who passed REALCODE (scale 1–7).

$\mu = 6.21, \sigma = 0.87$, control: $\mu = 5.67, \sigma = 1.06$, and collection: $\mu = 5.63, \sigma = 1.09$).

5 DISCUSSION AND FUTURE WORK

We recruited 613 participants from four crowdsourcing platforms (Appen, Clickworker, MTurk, and Prolific) and an University CS student mailing list; out of which 212 passed all (5/5) REALCODE programming questions suggested by Danilova et al. [10]. We find that overall CS students produce high quality data in terms of programming skills (highest), costs (lowest), number of duplicates (low), and passing attention check questions (high) compared to all crowdsourcing platforms. Among crowdsourcing platforms, we find that Prolific could generate a higher number of participants who would pass the basic programming questions and is more cost-effective compared to Appen, Clickworker, and MTurk. In the following, we discuss implications of our work in regards to self-reported programming skills (Section 5.1) and make suggestions both for researchers (Section 5.2) and crowdsourcing platforms (Section 5.3) to improve the recruitment process for participants with programming skills. We also dedicate a subsection to our experience with the recruitment channels as lessons learned for future researchers (Section 5.4).

5.1 Self-Reported Programming Skills

We find that participants clearly interpret "programming skills" very differently across all channels. All of the crowdsourcing participants self-identified as having programming skills in the screening survey, yet their answers on other questions shows a very wide range of experience and skill. Notably 31.5% of crowdsourcing participants could not correctly select the most fitting value for a Boolean from a multiple choice list and 33% could not select the correct description of a compiler. Even more interestingly 27.9% of crowdsourcing participants got one of those two questions wrong but not the other. These findings suggest that participants may indeed have some programming skills but their skills may be limited in scope and depth. For example, someone who only programs in an interpreted language like JavaScript might not have to know what a compiler is to write functioning code. Participants who have completed one programming course or who have done a self-directed coding project may also be indicating that they have programming skills.

Recent efforts have worked to bring programming to a wider audience by teaching it to younger people in schools and through self-learning websites aimed at general audiences. While laudable, these initiatives may also increasing the number of people exposed to programming concepts and therefore the number of people who might self-identify as having programming skills or programming experience. We therefore recommend that researchers avoid simple questions such as "Do you have programming skills?" for screening and instead ask questions that more concretely define the types of skills required by the research, for example the object-oriented question in PROGEX or the types of questions in REALCODE.

Using existing survey instruments though comes with some risks. Similar to how many of the questions proposed by Kahneman [27] for use in psychology experiments are now well known to crowdsourcing workers, over use of instruments like PROGEX or

REALCODE may lead to these groups learning and automatically responding with the "correct" answers rather than knowing them themselves. One avenue for future research is to extend these questionnaires; having a longer list of questions and randomly selecting a few of them to be presented to the participants (with a timer potentially as suggested for some questions in Danilova et al. [10]) may reduce the chances of participants using online resources to answer questions.

5.2 Recommendations for Researchers

5.2.1 Consider Using CS Students. We suggest recruiting CS students as a population for running studies that require programming skills. While they may not well represent seasoned developers working in industry, they are a good source of people with basic programming experience. If recruiting from a specific university, they will also have a more homogeneous background which can reduce unexpected confounds. Compared to crowdsourcing platforms CS students are accessible, cost-effective, and produce high quality data in terms of number of duplicates and passing attention check questions. Recruiting from multiple institutes through collaborations instead of staying with one mailing list may also help reduce biases inherent with a single-university sample.

5.2.2 Screen Crowdsourcing Participants and Account for Extra Costs. If researchers need participants with programming skills, we suggest using Prolific compared to Appen, Clickworker, and MTurk. Prolific allows screening participants without additional costs, however, the return rate could be low. Therefore, we suggest asking for a sample size with about three times more than what is the desired final sample size; or running a separate screening survey (e.g., if looking for 100 participants with programming skills, start the survey with 300, and then filter out participants who do not pass programming questions; or run a screening survey first). Running screening surveys or recruiting additional participants also causes additional costs which should be taken into account.

5.2.3 Consider Privacy and Security Di erences in Recruitment Channels. While CS students rated themselves lower in secure software development self-efficacy, they did the best on the programming questions. On the other hand, lower general security behaviors of CS students and higher privacy attitudes from them means that running studies involving privacy and security elements may result in a population with a higher privacy attitudes and lower security attitudes compared to a crowdsourcing population with programming skills. We find that even among crowdsourcing participants the privacy and security answers attitudes (e.g., Clickworkers have a higher awareness of privacy practices of online companies compared to MTurk and Prolific participants), which should be considered when recruiting participants for usable privacy and security studies. We suggest recruiting from multiple channels instead of one to minimize such effects and have a more diverse population.

5.3 Recommendations for Crowdsourcing Platforms

With the increase in human-centered studies [55], there is a demand for specific types of populations; for example in our case, a sample with programming skills. The opportunity here for crowdsourcing platforms is to offer specific and niche screening criteria and populations which may give them a competitive advantage over other platforms with basic screening criteria and a general population. Below we make suggestions to improve the situation for recruiting participants with programming skills.

Based on our findings it is clear that the question "Do you have programming skills" does not correlate with basic programming skills and knowledge. Participants on crowdsourcing platforms may over estimate their programming skills and such question is not helpful as a screening criteria in crowdsourcing platforms for recruiting participants with programming skills. We suggest crowdsourcing platforms to adopt new programming-related questions that better represent the level or type of skill to ensure that participants who pass those questions are able to participate in studies with required programming skills (e.g., questions from Danilova et al. [10]). We also believe that these questions should change and update over time so that the answers do not circulate in the participants community. One short-term and easy to adopt suggestion is to use the more specific self-reported questions such as experience in object-oriented programming (very inexperienced to very experienced [16]), years of experience in software development (numeric), and primary role in software development teams (e.g., software developer, tester, manager). While these are still self-reported, we find them overall more correlated to programming skills compared to the general yes/no question of "Do you have programming skills?"

5.4 Researcher Experience of Recruitment

While conducting this research, we gained a good deal of experience with the various platforms used to recruit participants. In this section, we reflect on our experience with these platform so future researchers can learn from our experiences. However, we should note that these are anecdotal observations, no formal usability evaluation of the platforms was done.

Several key challenges came up across all the platforms, though they handled them quite differently. The first was how to do handoff between the crowdsourcing platform and our Qualtrics survey software. Typically this is done by providing a link to the Qualtrics survey on the crowdsourcing platform, collecting the participant's platform-generated ID, and then after the survey is completed providing a completion code (either randomized or a static reused code) at the end of the survey which the participant then enters into the crowdsourcing platform to demonstrate completion of the job. We then compare the IDs and codes entered into the crowdsourcing platform against those recorded in Qualtrics and provide payment to those that are valid. Including/excluding participants was another challenge due to two reasons (1) including participants who passed our screening criteria in Appen, Clickworker, and MTurk, and (2) excluding participants who already took our survey once because we started our recruitment with smaller population sizes to catch any errors before going for a larger population size (for all crowdsourcing platforms). However, each platform had some amount of variation on these processes. Below, we discuss these challenges for each recruitment channel:

5.4.1 Appen. Using Appen required us to have an institute-wide account first, which we thankfully had, but it may mean extra time and money to setup for those without. The language used

on features is focused on running labeling tasks (e.g., rows, agreement, judgment). Support for running surveys is limited and the completion validation is limited to a static reused code (not randomized per participant) that also appears in the in an HTML tag (i.e., `data-validates-regex`). Including participants from the screening survey was done through a graphical interface that enabled us to import a CSV file with the selected participants' IDs of those who passed the screening criteria (excluding participants who already took the survey was done in this way as well). We could not find a way to include participant's ID in the link to Qualtrics as an extra parameter, so we also had to ask participants to manually enter their ID. We could also export additional information about participants from Appen itself such as location and start time.

5.4.2 Clickworker. Account creation did not require our institute to be involved. When creating a task, we had to select which countries we want our task to be advertised, where we had to add countries one by one from a list instead of adding countries all at once. To reduce the risks of having errors in the tasks, we ran tasks in smaller sample sizes first. However, because there was no option to increase the number of participants in a task, we had to run multiple tasks excluding participants from a previous task. Though, this was easy as we could select previous tasks to be excluded. The completion code was a static reused code that was the same for all participants. Excluding and including participants (e.g., for those who passed the screening criteria) is done through a "Team" feature which does not support uploading CSV files; meaning that to advertise the main survey to the selected participants, we had to add all selected participants' IDs manually one by one. There were no extra information provided about participants. Including participant's ID in the URL was not consistent and sometimes it did not include the IDs; we asked for IDs from participants in the survey to be consistent.

5.4.3 MTurk. We used MTurk's basic service and interface without the API. We were able to create an individual account. Including and excluding participants (e.g., for those who passed the screening criteria) required creating qualifications which were not the most intuitive method; however, it was possible to upload CSV files with a list of participants' IDs to be excluded/included which also means that excluding participants from previous tasks required us to create a new list to be excluded (again, we ran multiple smaller tasks first to reduce the chances of errors). MTurk also charges extra for adding qualifications to a task depending on their fees, qualification type, and number of participants. We were able to export a few extra information about participants such as start date and approval rate. We had to collect the participants' IDs from the users in the survey instead of passing them in the URL. A randomized completion per participant code was used.

5.4.4 Prolific. We were able to create an individual account. Including and excluding participants from previous tasks is done through the screening interface similar to other screening criteria by copy pasting a list of participants' IDs or completely excluding participants from a task. However, we did not use this feature because we did not run a screening survey on Prolific, and for increasing the number of participants we could increase the population size without worrying about excluding participants who already took the survey. It is possible to add multiple screening criteria such as programming skills, language, and level of education without extra cost. It is also possible to pass extra parameters including participants' ID in the URL and they will be captured by Qualtrics. Extra information about participants such as time taken, location, gender, and language can be exported as well. A static reusable completion code was used.

5.4.5 CS Students. We had to ask for permission to send out our email from our mailing list moderators. Other than the permission, we had no issues. We chose to not run a screening survey with this channel, therefore, we had no challenges with excluding/including participants. Compensation was easy to manage by randomly selecting participants and sending an email with their gift card.

6 CONCLUSION

We recruited 539 participants from four crowdsourcing platforms who claimed to have a programming experience and also 74 CS students from our local institute's CS mailing list (in total 613 participants). We found that overall CS students are a good source for recruiting participants with programming skills and Prolific, as a recruitment channel, in the tested crowdsourcing platforms results in a cost-effective sample with a higher number of participants with basic programming knowledge.

We recommend that researchers working in the human factors of software engineering account for the extra costs that will come with screening participants and also consider using CS mailing lists; potentially collaborating with other researchers in other institutes to reduce validity issues. We also suggest that developer-centered privacy and security researchers consider the differences in privacy and security attitudes of participants from various recruitment channels and aim for multiple channels to reduce bias against one recruitment channel which may have higher or lower privacy and security attitudes compared to other channels.

ACKNOWLEDGMENTS

We thank Chris Lucas for extensive advice on statistics as well as the reviewers for their constructive feedback helped improve the paper greatly. Mohammad did this work while he was at the University of Edinburgh. This work was sponsored in part by Microsoft Research through its Ph.D. Scholarship Program as well as the University of Edinburgh's Institute for Language, Cognition and Computation.

REFERENCES

[1] Julio Angulo and Martin Ortlieb. 2015. "WTH..!?!" Experiences, Reactions, and Expectations Related to Online Privacy Panic Situations. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*. USENIX Association, Ottawa, 19–38. https://www.usenix.org/conference/soups2015/proceedings/presentation/angulo

[2] Appen. 2021. *Confidence to Deploy AI with World-Class Training Data.* Retrieved August 2021 from https://appen.com

[3] Antonio A. Arechar and David G. Rand. 2021. Turking in the time of COVID. *Behavior Research Methods* (May 2021). https://doi.org/10.3758/s13428-021-01588-4

[4] 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*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300519

[5] Steven Bellman, Eric J. Johnson, Stephen J. Kobrin, and Gerald L. Lohse. 2004. International Differences in Information Privacy Concerns: A Global Survey of

Consumers. *The Information Society* 20, 5 (2004), 313–324. https://doi.org/10.1080/01972240490507956

[6] Karoline Busse, Mohammad Tahaei, Katharina Krombholz, Emanuel von Zezschwitz, Matthew Smith, Jing Tian, and Wenyuan Xu. 2020. Cash, Cards or Cryptocurrencies? A Study of Payment Culture in Four Countries. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. 200–209. https://doi.org/10.1109/EuroSPW51379.2020.00035

[7] James Carifio and Rocco Perla. 2008. Resolving the 50-year debate around using and misusing Likert scales. *Medical Education* 42, 12 (Dec. 2008), 1150–1152. https://doi.org/10.1111/j.1365-2923.2008.03172.x

[8] Technische Universität Chemnitz. 2021. *Measuring Programming Experience.* Retrieved September 2021 from https://www.tu-chemnitz.de/informatik/ST/research/material/PE/

[9] Clickworker. 2021. *AI Training Data and other Data Management Services.* Retrieved August 2021 from https://clickworker.com

[10] 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 Computer Society, Los Alamitos, CA, USA, 537–548. https://doi.org/10.1109/ICSE43902.2021.00057

[11] 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)*. Association for Computing Machinery, 13 pages. https://doi.org/10.1145/3377811.3380387

[12] Times Higher Education. 2021. *World University Rankings 2021 by subject: computer science.* Retrieved September 2021 from https://www.timeshighereducation.com/world-university-rankings/2021/subject-ranking/computer-science

[13] Serge Egelman, Marian Harbach, and Eyal Peer. 2016. *Behavior Ever Follows Intention? A Validation of the Security Behavior Intentions Scale (SeBIS).* Association for Computing Machinery, New York, NY, USA, 5257–5261. https://doi.org/10.1145/2858036.2858265

[14] Serge Egelman and Eyal Peer. 2015. Scaling the Security Wall: Developing a Security Behavior Intentions Scale (SeBIS). In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2873–2882. https://doi.org/10.1145/2702123.2702249

[15] Davide Falessi, Natalia Juristo, Claes Wohlin, Burak Turhan, Jürgen Münch, Andreas Jedlitschka, and Markku Oivo. 2018. Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineering* 23, 1 (Feb. 2018), 452–489. https://doi.org/10.1007/s10664-017-9523-3

[16] Janet Feigenspan, Christian Kästner, Jörg Liebig, Sven Apel, and Stefan Hanenberg. 2012. Measuring programming experience. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)*. IEEE, Passau, Germany, 73–82. https://doi.org/10.1109/ICPC.2012.6240511

[17] Robert Feldt, Thomas Zimmermann, Gunnar R. Bergersen, Davide Falessi, Andreas Jedlitschka, Natalia Juristo, Jürgen Münch, Markku Oivo, Per Runeson, Martin Shepperd, Dag I. K. Sjøberg, and Burak Turhan. 2018. Four commentaries on the use of students and professionals in empirical software engineering experiments. *Empirical Software Engineering* 23, 6 (Dec. 2018), 3801–3820. https://doi.org/10.1007/s10664-018-9655-0

[18] The R Foundation. 2021. *Fitting Generalized Linear Models.* Retrieved August 2021 from https://search.r-project.org/R/refmans/stats/html/glm.html

[19] Sandra Gabriele and Sonia Chiasson. 2020. Understanding Fitness Tracker Users' Security and Privacy Knowledge, Attitudes and Behaviours. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376651

[20] GitHub. 2021. *GitHub Acceptable Use Policies - GitHub Docs.* Retrieved November 2021 from https://docs.github.com/en/github/site-policy/github-acceptable-use-policies#6-information-usage-restrictions

[21] GitHub. 2021. *GitHub Privacy Statement - GitHub Docs.* Retrieved November 2021 from https://docs.github.com/en/github/site-policy/github-privacy-statement#public-information-on-github

[22] 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 2018)*. USENIX Association, Baltimore, MD, 265–281. https://www.usenix.org/conference/soups2018/presentation/gorski

[23] Margaret Gratian, Sruthi Bandi, Michel Cukier, Josiah Dykstra, and Amy Ginther. 2018. Correlating human traits and cyber security behavior intentions. *Computers & Security* 73 (2018), 345–358. https://doi.org/10.1016/j.cose.2017.11.015

[24] Lee Hadlington. 2017. Human factors in cybersecurity; examining the link between Internet addiction, impulsivity, attitudes towards cybersecurity, and risky cybersecurity behaviours. *Heliyon* 3, 7 (2017), e00346. https://doi.org/10.1016/j.heliyon.2017.e00346

[25] 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 *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 486–498. https://doi.org/10.1109/ICSE43902.2021.00053

[26] Roberto Hoyle, Srijita Das, Apu Kapadia, Adam J. Lee, and Kami Vaniea. 2017. *Was My Message Read? Privacy and Signaling on Facebook Messenger.* Association for Computing Machinery, New York, NY, USA, 3838–3842. https://doi.org/10.1145/3025453.3025925

[27] Daniel Kahneman. 2011. *Thinking, fast and slow.* Farrar, Straus and Giroux, New York.

[28] Ruogu Kang, Stephanie Brown, Laura Dabbish, and Sara Kiesler. 2014. Privacy Attitudes of Mechanical Turk Workers and the U.S. Public. In *Proceedings of the Tenth USENIX Conference on Usable Privacy and Security* (Menlo Park, CA) *(SOUPS '14)*. USENIX Association, USA, 37–49. https://www.usenix.org/conference/soups2014/proceedings/presentation/kang

[29] Maurits Clemens Kaptein, Clifford Nass, and Panos Markopoulos. 2010. Powerful and Consistent Analysis of Likert-Type Ratingscales. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2391–2394. https://doi.org/10.1145/1753326.1753686

[30] Amy J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. 2011. The State of the Art in End-User Software Engineering. *Comput. Surveys* 43, 3, Article 21 (April 2011), 44 pages. https://doi.org/10.1145/1922649.1922658

[31] Amy J. Ko, Thomas D. LaToza, and Margaret M. Burnett. 2015. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering* 20, 1 (Feb. 2015), 110–141. https://doi.org/10.1007/s10664-013-9279-3

[32] Amy J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. 2006. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. *IEEE Transactions on Software Engineering* 32, 12 (2006), 971–987. https://doi.org/10.1109/TSE.2006.116

[33] Oksana Kulyk, Benjamin Reinheimer, Lukas Aldag, Peter Mayer, Nina Gerber, and Melanie Volkamer. 2020. Security and Privacy Awareness in Smart Environments – A Cross-Country Investigation. In *Financial Cryptography and Data Security*, Matthew Bernhard, Andrea Bracciali, L. Jean Camp, Shin'ichiro Matsuo, Alana Maurushat, Peter B. Rønne, and Massimiliano Sala (Eds.). Springer International Publishing, Cham, 84–101. https://doi.org/10.1007/978-3-030-54455-3_7

[34] Naresh K. Malhotra, Sung S. Kim, and James Agarwal. 2004. Internet Users' Information Privacy Concerns (IUIPC): The Construct, the Scale, and a Causal Model. *Information Systems Research* 15, 4 (2004), 336–355. https://doi.org/10.1287/isre.1040.0032

[35] Winter Mason and Siddharth Suri. 2012. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior Research Methods* 44, 1 (March 2012), 1–23. https://doi.org/10.3758/s13428-011-0124-6

[36] Peter Mayer, Nina Gerber, Benjamin Reinheimer, Philipp Rack, Kristoffer Braun, and Melanie Volkamer. 2019. *I (Don't) See What You Typed There! Shoulder-Surfing Resistant Password Entry on Gamepads.* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300779

[37] Hendrik Meutzner, Santosh Gupta, and Dorothea Kolossa. 2015. Constructing Secure Audio CAPTCHAs by Exploiting Differences between Humans and Machines. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2335–2338. https://doi.org/10.1145/2702123.2702127

[38] Jenny Morales, Cristian Rusu, Federico Botella, and Daniela Quiñones. 2019. Programmer eXperience: A Systematic Literature Review. *IEEE Access* 7 (2019), 71079–71094. https://doi.org/10.1109/ACCESS.2019.2920124

[39] Brad A. Myers, Amy J. Ko, Thomas D. LaToza, and YoungSeok Yoon. 2016. Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools. *Computer* 49, 7 (2016), 44–52. https://doi.org/10.1109/MC.2016.200

[40] 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

[41] 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 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300370

[42] 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)*. Association for Computing Machinery, New York, NY, USA, 311–328. https:

//doi.org/10.1145/3133956.3134082
[43] 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 2018)*. USENIX Association, Baltimore, MD, 297–313. https://www.usenix.org/conference/soups2018/presentation/naiakshina
[44] Eyal Peer, Laura Brandimarte, Sonam Samat, and Alessandro Acquisti. 2017. Beyond the Turk: Alternative platforms for crowdsourcing behavioral research. *Journal of Experimental Social Psychology* 70 (2017), 153–163. https://doi.org/10.1016/j.jesp.2017.01.006
[45] Eyal Peer, Joachim Vosgerau, and Alessandro Acquisti. 2014. Reputation as a sufficient condition for data quality on Amazon Mechanical Turk. *Behavior Research Methods* 46, 4 (Dec. 2014), 1023–1031. https://doi.org/10.3758/s13428-013-0434-y
[46] Prolific. 2021. *Online participant recruitment for surveys and market research.* Retrieved August 2021 from https://www.prolific.co
[47] Qualtrics. 2021. *Fraud Detection.* Retrieved September 2021 from https://www.qualtrics.com/support/survey-platform/survey-module/survey-checker/fraud-detection/#RelevantID
[48] Qualtrics. 2021. *Qualtrics XM - The Leading Experience Management Software.* Retrieved September 2021 from https://www.qualtrics.com
[49] Jonathan Robinson, Cheskie Rosenzweig, Aaron J. Moss, and Leib Litman. 2019. Tapped out or barely tapped? Recommendations for how to harness the vast and largely unused potential of the Mechanical Turk participant pool. *PLOS ONE* 14, 12 (12 2019), 1–29. https://doi.org/10.1371/journal.pone.0226394
[50] Manaswi Saha, Michael Saugstad, Hanuma Teja Maddali, Aileen Zeng, Ryan Holland, Steven Bower, Aditya Dash, Sage Chen, Anthony Li, Kotaro Hara, and Jon Froehlich. 2019. *Project Sidewalk: A Web-Based Crowdsourcing Tool for Collecting Sidewalk Accessibility Data At Scale.* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300292
[51] Iflaah Salman, Ayse Tosun Misirli, and Natalia Juristo. 2015. Are Students Representatives of Professionals in Software Engineering Experiments?. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 666–676. https://doi.org/10.1109/ICSE.2015.82
[52] Shruti Sannon and Dan Cosley. 2019. *Privacy, Power, and Invisible Labor on Amazon Mechanical Turk.* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300512
[53] Yukiko Sawaya, Mahmood Sharif, Nicolas Christin, Ayumu Kubota, Akihiro Nakarai, and Akira Yamada. 2017. Self-Confidence Trumps Knowledge: A Cross-Cultural Study of Security Behavior. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 2202–2214. https://doi.org/10.1145/3025453.3025926
[54] Yukiko Sawaya, Mahmood Sharif, Nicolas Christin, Ayumu Kubota, Akihiro Nakarai, and Akira Yamada. 2017. Self-Confidence Trumps Knowledge: A Cross-Cultural Study of Security Behavior. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2202–2214. https://doi.org/10.1145/3025453.3025926
[55] Ben Shneiderman. 2017. Revisiting the Astonishing Growth of Human-Computer Interaction Research. *Computer* 50, 10 (Oct 2017), 8–11. https://doi.org/10.1109/MC.2017.3641625
[56] Janet Siegmund, Norbert Siegmund, and Sven Apel. 2015. Views on Internal and External Validity in Empirical Software Engineering. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 9–19. https://doi.org/10.1109/ICSE.2015.24
[57] Edward Smith, Robert Loftin, Emerson Murphy-Hill, Christian Bird, and Thomas Zimmermann. 2013. Improving developer participation rates in surveys. In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 89–92. https://doi.org/10.1109/CHASE.2013.6614738
[58] Statista. 2020. *Software developer gender distribution worldwide.* Retrieved September 2021 from https://www.statista.com/statistics/1126823/worldwide-developer-gender/
[59] Mohammad Tahaei, Alisa Frik, and Kami Vaniea. 2021. Deciding on Personalized Ads: Nudging Developers About User Privacy. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. USENIX Association, 573–596. https://www.usenix.org/conference/soups2021/presentation/tahaei
[60] Mohammad Tahaei, Alisa Frik, and Kami Vaniea. 2021. Privacy Champions in Software Teams: Understanding Their Motivations, Strategies, and Challenges. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, 1–15. https://doi.org/10.1145/3411764.3445768
[61] Mohammad Tahaei, Adam Jenkins, Kami Vaniea, and Maria K. Wolters. 2021. "I Don't Know Too Much About It": On the Security Mindsets of Computer Science Students. In *Socio-Technical Aspects in Security and Trust* (first edition ed.), Thomas Groß and Tryfonas Theo (Eds.). Springer International Publishing. https://doi.org/10.1007/978-3-030-55958-8
[62] Mohammad Tahaei, Tianshi Li, and Kami Vaniea. 2022. Understanding Privacy-Related Advice on Stack Overflow. In *Proceedings on Privacy Enhancing Technologies*. 1–18.
[63] Mohammad Tahaei and Kami Vaniea. 2019. A Survey on Developer-Centred Security. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 129–138. https://doi.org/10.1109/EuroSPW.2019.00021
[64] Mohammad Tahaei and Kami Vaniea. 2021. "Developers Are Responsible": What Ad Networks Tell Developers About Privacy. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI '21 Extended Abstracts)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3411763.3451805
[65] Mohammad Tahaei, Kami Vaniea, Beznosov Konstantin, and Maria K. Wolters. 2021. Security Notifications in Static Analysis Tools: Developers' Attitudes, Comprehension, and Ability to Act on Them. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3411764.3445616
[66] Mohammad Tahaei, Kami Vaniea, and Naomi Saphra. 2020. Understanding Privacy-Related Questions on Stack Overflow. 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, 1–14. https://doi.org/10.1145/3313831.3376768
[67] Phillip Merlin Uesbeck, Andreas Stefik, Stefan Hanenberg, Jan Pedersen, and Patrick Daleiden. 2016. An Empirical Study on the Impact of C++ Lambdas and Programmer Experience. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 760–771. https://doi.org/10.1145/2884781.2884849
[68] Daniel Votipka, Desiree Abrokwa, and Michelle L. Mazurek. 2020. Building and Validating a Scale for Secure Software Development Self-Efficacy. 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, 1–20. https://doi.org/10.1145/3313831.3376754
[69] Jing Xie, Bill Chu, Heather Richter Lipford, and John T. Melton. 2011. ASIDE: IDE Support for Web Application Security. In *Proceedings of the 27th Annual Computer Security Applications Conference* (Orlando, Florida, USA) *(ACSAC '11)*. Association for Computing Machinery, New York, NY, USA, 267–276. https://doi.org/10.1145/2076732.2076770
[70] Aiko Yamashita and Leon Moonen. 2013. Surveying developer knowledge and interest in code smells through online freelance marketplaces. In *2013 2nd International Workshop on User Evaluations for Software Engineering Researchers (USER)*. 5–8. https://doi.org/10.1109/USER.2013.6603077
[71] Yixin Zou, Kevin Roundy, Acar Tamersoy, Saurabh Shintre, Johann Roturier, and Florian Schaub. 2020. Examining the Adoption and Abandonment of Security, Privacy, and Identity Theft Protection Practices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3313831.3376570

A PARTICIPANTS' DEMOGRAPHICS

Table 4 shows a summary of participants in the main survey.

B ANSWERS TO ALL SURVEY INSTRUMENTS

Table 5 shows mean and standard deviation of each survey instrument per recruitment channel.

C ANSWERS TO REALCODE QUESTIONS

Table 6 shows the five instrument questions with the percentage of participants in each recruitment channel that answered the question correctly. We also include "Programmer" and "Non-Programmer" columns from the original REALCODE paper [10] for comparison to prior work.

D ANSWERS TO PROGEX QUESTIONS

Table 7 shows the mean and standard deviation for three recommended questions by Feigenspan et al. [16]. We also include a column to present the data from the original PROGEX paper for comparison with prior work.

Table 4: Summary of demographics. μ: mean, σ: standard deviation. The denominator for percentages is the respective "population size."

	Appen	Clickworker	MTurk	Prolific	CS Students	Total
Population size	9	38	167	325	74	613
Self-reported as a CS student*	3 (33.3%)	10 (26.3%)	90 (53.9%)	47 (14.5%)	72 (97.3%)	222 (36.2%)
Gender						
Male	8 (88.9%)	31 (81.6%)	121 (72.5%)	240 (73.8%)	40 (54.1%)	440 (71.8%)
Female	1 (11.1%)	6 (15.8%)	46 (27.5%)	79 (24.3%)	28 (37.8%)	160 (26.1%)
Other		1 (2.6%)		6 (1.8%)	6 (8.1%)	13 (2.1%)
Number of employees						
1-9 employees	2 (22.2%)	10 (26.3%)	4 (2.4%)	100 (30.8%)	29 (39.2%)	145 (23.7%)
10-99 employees	1 (11.1%)	10 (26.3%)	52 (31.1%)	85 (26.2%)	15 (20.3%)	163 (26.6%)
100-999 employees	4 (44.4%)	9 (23.7%)	87 (52.1%)	64 (19.7%)	11 (14.9%)	175 (28.5%)
1,000-9,999 employees	2 (22.2%)	6 (15.8%)	18 (10.8%)	45 (13.8%)	5 (6.8%)	76 (12.4%)
10,000 or more employees		3 (7.9%)	6 (3.6%)	31 (9.5%)	14 (18.9%)	54 (8.8%)
Primary role in software teams						
Developer	5 (55.6%)	21 (55.3%)	71 (42.5%)	96 (29.5%)	27 (36.5%)	220 (35.9%)
Manager	2 (22.2%)	6 (15.8%)	49 (29.3%)	49 (15.1%)	1 (1.4%)	107 (17.5%)
Designer		2 (5.3%)	14 (8.4%)	21 (6.5%)	3 (4.1%)	40 (6.5%)
Privacy/Security			1 (0.6%)	8 (2.5%)	2 (2.7%)	11 (1.8%)
Tester		2 (5.3%)	18 (10.8%)	22 (6.8%)	2 (2.7%)	44 (7.2%)
Not CS-related	2 (22.2%)	7 (18.4%)	12 (7.2%)	92 (28.3%)	6 (8.1%)	119 (19.4%)
Unemployed				13 (4%)	29 (39.2%)	42 (6.9%)
Other			2 (1.2%)	24 (7.4%)	4 (5.4%)	30 (4.9%)
Latest employment status						
Full-time	8 (88.9%)	25 (65.8%)	146 (87.4%)	203 (62.5%)	10 (13.5%)	392 (63.9%)
Part-time		3 (7.9%)	11 (6.6%)	22 (6.8%)	9 (12.2%)	45 (7.3%)
Consultant/Freelance	1 (11.1%)	6 (15.8%)	7 (4.2%)	39 (12%)		53 (8.6%)
Student		3 (7.9%)	2 (1.2%)	33 (10.2%)	44 (59.5%)	82 (13.4%)
Furloughed/On leave		1 (2.6%)		3 (0.9%)		4 (0.7%)
Unemployed				14 (4.3%)	4 (5.4%)	18 (2.9%)
Other			1 (0.6%)	11 (3.4%)	7 (9.5%)	19 (3.1%)
Current continent						
Africa	2 (22.2%)	5 (13.2%)		25 (7.7%)		32 (5.2%)
Asia	5 (55.6%)	10 (26.3%)	39 (23.4%)	3 (0.9%)	9 (12.2%)	66 (10.8%)
Europe	1 (11.1%)	13 (34.2%)	3 (1.8%)	254 (78.2%)	65 (87.8%)	336 (54.8%)
North America	1 (11.1%)	8 (21.1%)	123 (73.7%)	39 (12%)		170 (27.7%)
Oceania		2 (5.3%)		4 (1.2%)		6 (1%)
South America			2 (1.2%)			2 (0.3%)
Other			1 (0.6%)			1 (0.2%)
Age (μ,σ)	40.22, 5.78	34.97, 9.07	33.92, 8.21	34.07, 8.75	22.91, 5.29	32.62, 9.09
Years of software dev (μ,σ)	11.78, 6.83	8.95, 8.84	5.77, 5.22	5.44, 6.70	2.94, 3.59	5.48, 6.31
Team members (μ,σ)	13, 8.51	12.74, 25.58	35.81, 88.51	8.34, 21.29	5.24, 5.26	18.37, 58.82

*Despite our screening for not being a student in the crowdsourcing platforms, we got many CS students from them which might come from some participants working and also being a CS student.

Table 5: Mean (μ) and standard deviation (σ) of all five survey instruments across channels. Numbers in the parentheses shows the range.

	Appen (N=9)		**Clickworker** (N=38)		**MTurk** (N=167)		**Prolific** (N=325)		**CS Students** (N=74)		**Total** (N=613)	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
PROGEX												
Programming experience (0–10)	7.89	1.45	6.34	2.04	7.19	2.06	5.20	2.37	5.74	1.55	6.00	2.39
Programming experience compared to class/work mates (0–5)	4.11	0.33	3.63	0.97	3.92	0.80	3.41	0.98	3.61	0.92	3.59	0.97
Experience in object-oriented programming (0–5)	4.00	0.87	3.92	1.05	3.91	0.92	3.41	1.22	3.89	0.92	3.62	1.13
REALCODE												
Count of correct answers (0–5)	2.67	0.87	4.34	1.17	2.13	1.51	3.23	1.67	4.84	0.50	2.98	1.77
SSDSES (0–5)												
Security communication	3.51	1.33	3.34	0.91	3.68	0.67	3.05	1.07	2.61	0.96	3.23	1.01
Vulnerability identification and mitigation	3.33	1.21	2.93	0.98	3.65	0.70	2.64	1.05	2.04	0.83	1.08	1.28
SEBIS (0–5)												
Proactive awareness	3.53	0.83	3.74	0.82	2.91	0.69	3.74	0.67	3.45	0.65	3.40	0.79
Device securement	4.08	1.35	4.41	0.91	3.92	0.71	4.30	0.80	4.20	0.66	4.13	0.81
Password generation	3.75	0.89	3.59	0.78	3.36	0.61	3.56	0.82	3.32	0.74	3.44	0.76
Updating	4.19	1.03	3.85	0.83	3.76	0.69	3.76	0.81	3.33	0.71	3.72	0.78
IUIPC (0–7)												
Awareness	5.35	1.39	5.45	1.28	5.41	0.92	5.66	0.90	5.87	0.83	5.94	0.90
Collection	4.92	1.91	5.49	1.35	5.43	0.99	5.67	1.06	5.63	1.23	5.56	1.12
Control	5.35	1.39	5.45	1.28	5.41	0.92	5.66	0.90	5.87	0.83	5.58	0.96

Table 6: Percentage of participants from each channel that correctly answered each of the REALCODE questions. "Programmer" and "Non-Programmer" columns are from Danilova et al. [10] for comparison with ground-truth (non-)programmers.

	Collected data in this paper						**Danilova et al. [10]**	
	Appen	**Clickworker**	**MTurk**	**Prolific**	**CS Students**	**Total**	**Non-Programmer**	**Programmer**
Population size	9	38	167	325	74	613	100	50
Frequent website as aid for programming	5 (55.6%)	33 (86.8%)	55 (32.9%)	200 (61.5%)	74 (100%)	367 (59.9%)	6 (6%)	50 (100%)
Description of a compiler's function	7 (77.8%)	33 (86.8%)	76 (45.5%)	246 (75.7%)	72 (97.3%)	434 (70.8%)	33 (33%)	50 (100%)
Definition of a recursive function	2 (22.2%)	33 (86.8%)	69 (41.3%)	199 (61.2%)	70 (94.6%)	373 (60.8%)	30 (30%)	50 (100%)
Value of a Boolean variable	8 (88.9%)	34 (89.5%)	81 (48.5%)	248 (76.3%)	73 (98.6%)	444 (72.4%)	25 (25%)	50 (100%)
Parameter of the function in a sample code	2 (22.2%)	32 (84.2%)	74 (44.3%)	157 (48.3%)	69 (93.2%)	334 (54.5%)	13 (13%)	50 (100%)
All answers correct	0 (0%)	24 (63.2%)	14 (8.4%)	108 (33.2%)	66 (89.2%)	212 (34.6%)	2 (2%)	50 (100%)

Table 7: Programming experience, programming experience comparing to classmates, and experience with object-oriented programming for our data as well as data from Feigenspan et al. [16]. The columns with REALCODE split the data into participants who passed all five REALCODE programming questions vs. those who did not.

	Collected data in this paper						**Feigenspan et al. [16]**
	Individual recruitment channels				**Regardless of recruitment channel**		
	Prolific	**Clickworker**	**MTurk**	**CS students**	**Passed REALCODE**	**Failed REALCODE**	
Programming experience							
μ	5.20	6.34	7.19	5.74	6.38	5.62	4.63
σ	2.37	2.04	2.06	1.55	1.85	2.53	1.81
Programming experience compared to class/work mates							
μ	3.41	3.63	3.92	3.61	3.78	3.48	2.05
σ	0.98	0.97	0.80	0.92	0.83	0.99	0.84
Experience with object-oriented programming							
μ	3.41	3.92	3.91	3.89	4.04	3.42	3.59
σ	1.22	1.05	0.92	0.92	0.89	1.17	0.92