Accepted to appear in the Proceedings of the
International Conference on Software Engineering 2023.
This manuscript will be revised before final camera-ready submission.

Measuring Secure Coding Practice and Culture: A Finger Pointing at the Moon is not the Moon

Ita Ryan
ADVANCE Centre for Research Training
School of Computer Science and IT
University College Cork
Cork, Ireland
ita.ryan@cs.ucc.ie

Utz Roedig
Connect Research Centre
School of Computer Science and IT
University College Cork
Cork, Ireland
u.roedig@cs.ucc.ie

Klaas-Jan Stol
Lero, the SFI Research Centre for Software
School of Computer Science and IT
University College Cork
Cork, Ireland
k.stol@ucc.ie

***Abstract*—Software security research has a core problem: it is impossible to prove the security of complex software. A low number of known defects may simply indicate that the software has not been attacked yet, or that successful attacks have not been detected. A high defect count may be the result of white-hat hacker targeting, or of a successful bug bounty program which prevented insecurities from persisting in the wild. This makes it difficult to measure the security of non-trivial software. Researchers instead usually measure effort directed towards ensuring software security. However, different researchers use their own tailored measures, usually devised from industry secure coding guidelines. Not only is there no agreed way to measure effort, there is also no agreement on what effort entails. Qualitative studies emphasise the importance of 'security culture' in an organisation. Where software security practices are introduced solely to ensure compliance with legislative or industry standards, a 'checkbox' attitude to security may result. The security culture may be weak or non-existent, making it likely that precautions not explicitly mentioned in the standards will be missed. Thus, researchers need both a way to assess software security practices and a way to measure software security culture. To assess security practice, we converted the empirically-established 12 most common software security activities into questions. To assess security culture, we devised a number of questions grounded in prior literature. We ran a secure development survey with both sets of questions, obtaining organic responses from 1,100 software coders in 59 countries. Our results show that some coders still work in environments where there is little to no attempt to ensure code security. We used proven common activities to assess security practice, and made the first attempt to quantitatively assess security culture. Our analysis found that secure coding practice is not always matched by a secure coding culture, which may lead to problems in defect prevention and sustained security effort.**

***Index Terms*—Security, secure coding, security compliance**

I. Introduction

Software security was defined in 2004 as *'the idea of engineering software so that it continues to function correctly under malicious attack [1].'* Since then, our world has become increasingly connected, and the hacker community has morphed into a global criminal industry. Insecure software is now routinely exploited for ransomware, cybercrime and cyberespionage purposes, so that software security is an increasingly urgent requirement.

Academia has not reached consensus on a way to measure the level of secure coding in an organisation or open source team. Ethnographic studies provide rich qualitative data, but by their nature [2] do not produce quantitative measurements or generalisable conclusions. Most researchers attempting quantitative secure coding research synthesise a number of practices from industry methodologies and assess participants' use of them. These synthesised secure coding practice lists tend to be used only by their authors. They are not empirically evaluated, and if fine-grained may be too rigid to be re-used over time, as software security is a quickly-evolving field. In addition, highly security-conscious individuals may not wish to answer detailed security practice questions, ironically due to security concerns.

The question of what is, or should be, measured is complicated by the existence of the 'checkbox mentality' [3], a compliance-focused attitude to security that reflects a weak security culture. Questions measuring only security practice ignore issues of culture that may affect the security of code produced. Security practice and security culture, though related, are separate phenomena; there has been no attempt to date to measure security culture in software development. This has serious implications for the study of secure software development. The secure coding practices in an environment are only the tip of the iceberg. The security culture in the environment, out of sight beneath the surface, may be a much larger determinant of long-term security success. Security culture determines whether secure coding activities are kept up-to-date, and are undertaken with the necessary conviction, enthusiasm and insight.

To explore these issues, we conducted a large-scale survey of software developers. We wished to evaluate the current state of secure coding practice. Twelve security activities were recently identified by Weir et al. [4], in an analysis of a large trove of industry secure-coding data, as being those most commonly adopted in secure coding initiatives in large organisations. Given their empirical backing, we adopted these *'common activities'* (CAs) as our core measurements for secure software coding practice.

There is no existing precedent for the quantitative measurement of security culture. Again grounding our questions firmly in prior literature, we devised a list of such questions to investigate this as-yet unmeasured phenomenon. Thus, in this paper we seek to address the following research questions:

A. Research Question 1: What are the secure-coding characteristics of our sample group?

B. Research Question 2: What are the security culture characteristics of our sample group?

C. Research Question 3: Do secure coding practice and culture correlate, and if not, what lessons can we learn to help support the development of secure coding?

This paper makes the following contributions: first, we present the '*CA Score*', a lightweight instrument for assessment of the level of software security practice in a work environment. Second, we provide an overview of the current state of security practice based on our survey results. Third, we provide a group of questions designed to assess security culture. Fourth, we assess security culture in our participants' working environments. Finally, we explore the interplay between security practice and security culture.

II. Background and Related Work

A. Measuring Secure Development

While assessing the security of software developed in controlled studies is feasible and repeatable [5]–[7], measuring secure development in organisations and open source environments is an ongoing issue in academia. As far back as 2012, Jaatun [8] discussed whether software security can be measured at all. Counts of known security defects for publicly available applications could be used; however, some applications are subject to considerably more analysis and attack than others. An absence of known vulnerabilities could simply indicate that the application has not been closely studied. An example of an attempt to demonstrate a correlation between use of assurance techniques and decreased levels of security issues in code comes from Weir et al. [9]. They surveyed 335 Android app developers on their use of security assurance techniques, downloaded a free app from each developer, and used automated analysis to detect instances of three categories of security issue in the apps. Surprisingly, app analysis showed no increased security for apps whose creators claimed to use assurance techniques, and more cryptographic security issues for apps whose creators had access to security experts. The authors concluded that issues with analysis tools and missing cryptography may have skewed the analysis. This study illustrates the complexity and difficulty inherent in determining whether secure coding efforts are correctly reported, and whether they result in secure code outcomes.

BSIMM reports on contemporary software security activities are compiled semi-annually by Synopsys, who interview contributing organisations about their security practices. For example, in 2021 BSIMM 12 collated data from 128 organisations. The contributing organisations are usually large and have active software security strategies. Jaatun [8] discussed using the BSIMM [10] approach of measuring use of software security activities for academic studies. Variations on this approach of measuring activities, synthesising BSIMM and other industry practices and guidelines such as OWASP's Software Assurance Maturity Model (SAMM) [11], Microsoft Security Development Lifecycle (MS-SDL) [12] and Software Assurance Forum for Excellence in Code (SAFECode) [13] have been widely used by the research community [14]–[19]. Each research team, however, measures different activities. New groups of activities to measure are continually defined. None are empirically evaluated. No research team reuses the groups defined by previous teams. Thus, research results are difficult to compare.

Similarly, Morrison et al. [20] found that 85% of 324 unique security metrics had been proposed and evaluated solely by their authors. They concluded that there is no convergence as yet on an accepted set of metrics in the software life cycle security metrics field.

B. Security Culture

It might be argued that engaging in security practices would naturally entail a good security culture. However, many observers have noted that organisations may follow recommend practices (perhaps for compliance) but have a '*checkbox*' attitude to security. Although mentioned in several papers [**?**], [3], and implied in others [21]–[23], this phenomenon and its implications are rarely explored.

In some environments, implementation of security activities is driven by a need to demonstrate compliance with regulations or standards such as the payment card industry's DSS [24]. Such standards may have a narrow focus [23], and security investment which focuses solely on compliance is not sufficient to ensure secure software [25]. The significance of the organisation's security posture in the area of secure coding is widely acknowledged in academia [26]–[28]. Assal et al. [29] found that most deterrents to developers coding securely relate to the absence of clear secure-coding plans, resources and priorities in the developers' working environments. Concluding that it is important for companies to build a security culture and offer learning opportunities for secure coding, they suggested that a research focus on problems in the organisational approach to security would increase understanding of software security deterrents.

Recent ethnographic studies have provided useful insights into the security culture in organisations. Lopez et al. [30] immersed themselves in the software development unit of a large company, examining how non-experts treat security during their daily tasks and focusing primarily on developer security motivation. They concluded that the organisation had put procedures and activities in place to ensure software security, and that the developers accepted these and attempted to implement them in good faith. This resulted in *'mostly secure'* software. By contrast, Morales et al. [22] published a devastating paper on a well-funded project using Agile and implementing DevSecOps without rigorous adherence to DevSecOps principles, and the resulting security implications.

It is a valuable reminder that no development method can compensate for bad management. Broken pipelines, adversarial subcontractor relationships, inadequate definitions of done, poor test resources, a failure to emphasise security and more all combined to produce a product with poor security assurance. On paper, this dysfunctional project adopted appropriate security practices. So where did it go wrong?

The differentiating factor between these two organisations is security culture, discussed by Haney et al. [3], who found that all of the cryptography-focused organisations they studied placed a high value on a strong security culture. Tuladhar et al. [31], embedded in an organisation adopting secure coding practices, concluded that key security culture enablers were upper management setting security as a goal, and the team applying security knowledge in context. Arizon-Peretz et al. [32] echoed the importance of upper management commitment, and found that the participants in their interview study encountered *'inconsistent and confusing cues.'*

The question of how individual developers' security enthusiasm is regarded within their work environment, an important aspect of security culture, has not been much explored by the research community. Tahaei et al. [33] examined the experience of privacy *'champions'* in software teams, finding that they play an important advocacy role. Ryan et al. [34] identified a *'hero'* software security archetype; a coder struggling to introduce secure coding practices in a security-hostile environment. Jaatun et al. [14] noted organisations' dependence on individual developers' enthusiasm. While Haney and Lutters [35] looked at how cybersecurity advocates attempt to influence cybersecurity behaviours, they only interviewed recognised security experts in senior roles. Weir et al. [36] used a series of interventions within organisations to attempt to empower developers to code securely, but as this involved prior organisational agreement, a level of management support was assured. Studies on developers' tool adoption behaviour have focused on social influences [37]. Although Witschey et al. [38] observed that many developers seek out information on security tools when needing to write secure code, they did not examine how developers can introduce such tools into their working environment, and what constraints they encounter when attempting to do so. An ethnographic study by Palombo et al. [39] shed some light on how heroes can succeed in improving security in a *'security inattentive'* [18] environment. Two researchers used and advocated a *'co-creation'* model for secure development in which they added security checks seamlessly, with no developer friction for other team members. This required considerable investigation and work, but other team members then adopted the checks, possibly because of their frictionless nature. The researchers did not attempt to introduce external security tools.

We have not found in the literature a way to measure security culture in a work environment. Although Votipka et al. [19] suggested that the secure software development self-efficacy (SSD-SES) scale could be used to measure security culture, a developer's self-efficacy could arise from previous employment or personal motivation, and cannot safely be attributed to the security culture in their current working environment.

A set of questions to provide a quantitative measure of security culture would allow us to establish where a coding environment is on the security culture spectrum. This would enable us to see beyond the visible tip of the iceberg, to the security reality beneath.

III. Methodology

We conducted a large-scale cross-sectional survey. In this section we discuss the design of the questionnaire, data collection procedures, data screening procedures, and data analysis procedures.

Initial questions concerned demographics and participants' personal relationship with secure coding. These questions are not the focus of this paper so we will not discuss them here. A complete list of questions can be found in the supplementary material.

Surveys that constrain respondents' answers to specific values can impose forced choice bias, which is inappropriate in a context where increased understanding of context is sought [40]. Therefore we allowed free text in many questions, and asked a question towards the end looking for comments on any aspect of the survey or of the respondent's secure software development experience. The free text aspect of the survey provided us with many interesting insights. As the use of *'(sic)'* can be seen as condescending, we do not use it when quoting respondents. Any quote from respondents is presented exactly as entered in the questionnaire.

A. Questions on Secure Coding Posture in Participants' Work Environments

Questions in this study arose from a review of prior literature. They are designed to measure software security and security culture via survey. An organisation's secure coding posture comprises not only its secure coding practice but also the associated security culture. We included two types of questions to assess secure coding posture in respondents' work environments. See Section V for details on how individual questions were chosen.

1) Common Activities (CAs): The data gathered for annual BSIMM reports since 2008 provides a rich source of longitudinal information on software security activities in industry. This data was explored in a 2021 study by Weir et al. [4], who identified 12 security activities as being those that are usually adopted '*early, together, and notably more often than any others*' by BSIMM contributors. We judged that determining the presence or absence of these activities in an environment would give a contemporary objective assessment of the environment's software security practice. Therefore, we asked our participants whether they were aware of each of these 12 CAs in their working environments. The text of the questions can be seen in Table III.

2) Security Culture: Haney et al. [3] define security culture as *'a subculture of an organization in which security becomes a natural aspect in the daily activities of every employee.'* As discussed, we did not find in the literature a way to measure security culture in a work environment. We therefore devised questions based on existing literature describing positive and negative security cultures. There is an overlap between practice and culture in our questions, in that our culture question on the use of tools (SCQ3) asks about an activity that may also be used to assess security practice. However, tool use is a useful proxy for the cultural value of expending effort, time and money on security.

B. Data Collection

The survey was publicised via personal contacts, a conference talk by the first author, and social media platforms such as LinkedIn, Facebook, and Twitter. We found that it was possible to generate significant interest by asking for retweets on Twitter, posting to Facebook development groups, and promoting the survey on LinkedIn. The survey was open for just over 3 months, to the end of January 2022. This long window provided time to publicise the survey, and helped build momentum. At closing time, we had received 1,100 responses.

Researchers running secure software development surveys have encountered issues with respondent quality when offering payment for completion. Danilova et al. [41] found that of 129 respondents sourced from Qualtrics, 96 did not pass programming tests. Witschey et al. [42] cleaned 313 suspiciously-speedy participants from their survey, leaving only 61 usable responses. Given the considerable interest we succeeded in generating in our survey, we did not need to offer financial incentives or use platforms such as Prolific to source participants. This removed the danger of non-developers participating for financial gain. However, it remained important to carefully screen responses to ensure high quality analysis results; we discuss the screening process next.

C. Screening Questions

One concern in sample research is that the sample is representative. We were only interested in responses from people who were coding frequently at the time of answering the questionnaire, so we began by asking respondents *'Do you write computer code frequently, either professionally or open source?'* Those who answered *'No'* were excluded from further participation.

Danilova et al. [43] assessed a number of screening questions to filter out non-developers from paid studies. Tahaei and Vaniea [44] used these questions to compare results for participants recruited from several different paid sources. The Danilova paper recommended two programming comprehension questions as the best screening choices in surveys where a time penalty is acceptable. These two questions have since been used successfully, for example by Kaur et al. [45]. We also used these two questions, adapting question wording slightly since the answers documented by Danilova et al. are in the public domain. We had some interesting findings on them, discussed in Sec. III-D.

Since our questionnaire was relatively long with 62 questions, we wanted to ensure that participants were paying attention all the way through. Rather than ask complex attention questions with opposing answers, we borrowed a simple strategy from Murphy-Hill et al. [46], simply stating for example *'This is an attention question. Please select a little.'* We had two such questions, designed to blend with adjacent questions so that participants who were simply clicking blindly would not notice them. This strategy caused some amusement; for example, one participant noted as a comment at the end of the survey *'The attention questions are funny ^^.'*

D. Screening Process

Since only the questions on programming expertise were mandatory, and some people think more quickly than others, we did not reject participants who completed the questionnaire quickly.

Fifty-two respondents answered *'No'* to the initial screening question on frequent coding, and were brought directly to the end of the survey, leaving 1,048 respondents. We then proceeded to the two mandatory programming expertise questions. We had intended to reject all participants who got either of the programming questions wrong. However, a tranche of six or seven survey entries was obtained immediately after the first author gave a software security talk to an audience comprised of SQL programmers. In this tranche, respondents included SQL in the list of programming languages they used, and some got the second programming question wrong. Upon reviewing the questions, we realised that these included pseudocode that would be typical of C++ or Java interview questions but is meaningless in SQL. The questions' limitations were confirmed later, when a survey participant communicated with the first author, commenting that Python programmers could have difficulty with these questions. Fifty-five respondents answered one or both of the programming screening questions incorrectly.

Similarly, we had planned to remove any respondent from the data set who got an attention question wrong. This seemed like an uncontroversial position, but one participant, answering Question 60 which gave an opportunity to comment on the survey, responded *"Ididint understand q56. Attention? No idea what you are wanting from this poor question."* Thirty-seven respondents failed one or both of the attention questions.

Having considered the feedback on our screening questions, we decided to consider two groups of respondents during analysis. Group 1 ('all valid participants') includes all respondents who confirmed that they write computer code frequently, giving a total of 1,048. Group 2 ('all correct participants') additionally omits respondents who got programming and/or attention questions wrong (but retains the respondent who explicitly told us that they didn't understand the attention questions), giving a total of 962. Statistical results reported in the paper are based on Group 2; however, tests were also run on Group 1. No significant difference was found between the two groups.

E. Ethics

Our questionnaire involved research on human subjects, and therefore we obtained approval for the questionnaire from our

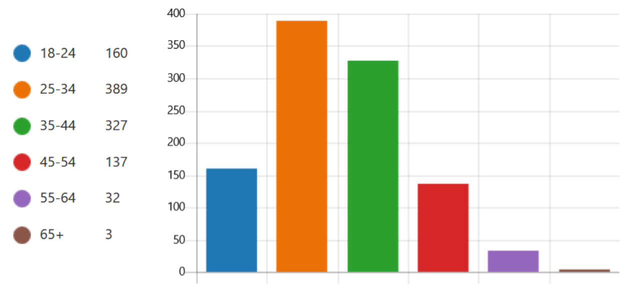

Fig. 1. Participant age

TABLE I
GENDER DISTRIBUTION OF THE SAMPLE

Gender	Frequency	Percent
Man	852	81
Woman	94	9
Non-binary	63	6
Prefer not to say	34	3
Other	4	0.4

institution's Social Research and Ethics Committee. Participants were advised that taking the questionnaire was not obligatory, and were asked to consent to taking the questionnaire. All submissions were anonymous.

F. Data Analysis

Statistical analysis was done using R [47]. Positive answers to the 12 CA questions were summed to create a CA score between 1 and 12, representing the secure coding level in an environment. The Pearson correlation coefficient was used to assess correlations between this score and security culture indicators, measuring correlation of increasing scores with increasing levels of these indicators [48]. A replication package is available in the online materials.

Qualitative data from open-ended survey questions was used to provide context for our quantitative results. However, qualitative data analysis was not done for this paper.

IV. Sample Description

A. Demographics

Respondents ranged in age from 18 to over 65 (see Fig. 1). We asked respondents what gender they most identified with. The majority of respondents answered *'Man.'* The numbers seemed to correspond with approximate gender balance numbers in the industry (see Table I). We asked developers what country they live in, allowing them to add their country if it was missing. We obtained responses from 59 different countries across all continents except Antarctica. The highest numbers of respondents came from the United States (453), United Kingdom (110) and Germany (92).

B. Team and Organisation Size

itaTodo: add the breakdown here. Developers working alone were advised to select '*Not applicable*' in relevant survey responses. Solo developers were excluded from statistical analysis in the Results section, since group dynamics are not relevant for them.

C. Programming Languages and Tools

We provided developers with a list of programming languages and other technologies and asked which they used frequently, also allowing free text entries. We ended with a list of 73 programming languages, of which JavaScript (454), Python (447) and HTML/CSS (332) were the most popular. Some minority languages, such as Chicken Scheme and Moonscript, were entirely new to us. We asked about other technologies; 129 were entered. Git (n=966) was by far the most popular, followed by Docker (n=470), PostgreSQL (n=349) and Amazon Web Services (AWS) (n=336). Sixty-eight unusual technologies appeared only once, including Cassandra, ScyllaDB, and Godot, which is apparently *'The game engine you waited for'*.

The field of tools used to enhance code security is rapidly evolving. We wanted to get a sense of which are in general use. Asked which security tools they use, developers entered over 100 different tools. Clang Analyzer and SonarQube were by far the most popular (see Fig. 2), but the large range indicates little convergence in the tools market.

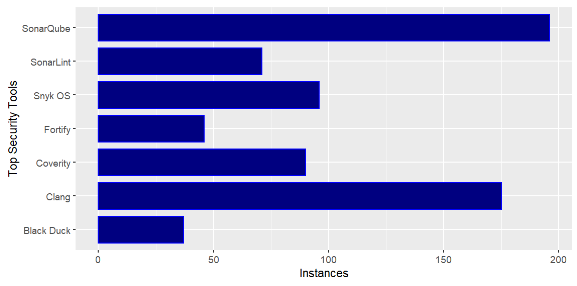

Fig. 2. Security Tools Use

D. Adoption of Secure Coding Standards

We asked developers what secure coding initiatives or standards they are aware of in their organisations or teams. We provided a list of well-known initiatives, and allowed developers to add their own. By far the most common response was *'None'* (n=304), followed by *'Not applicable'* (n=212). PCI-DSS, used in the payments industry, followed at 78, with FIPS 140-2 (n=43), Common Criteria (n=37) and US-CERT's Top 10 Secure Coding Practices (n=33) also fairly common. Industry secure-coding standards used in academia fared badly; the BISMM got only three mentions, OWASP's SAMM two, and SAFECode five, and of the commonly-referenced general approaches only MS-SDL hit double digits at 29.

The free text section provided further insights. MISRA, standards set by the Motor Industry Software Reliability Association, appeared seven times. Mention was also made of government standards from France, Germany, the US and the UK. We had inadvertently omitted an *'I don't know'* option; 18 people added this in free text. We received other thoughtful entries such as *'Standards are artificial, we try to focus on* *actual security (incidentally we probably apply some of them)'* and some less thoughtful ones: *'Yes there is some I don't care.'*

E. Adoption of Development Methods

We asked participants about the development methods in their organisation, providing 15 common options (allowing selection of multiple options), and a free text option. We found that Agile (n=650), DevOps (n=445), Scrum (n=358) and Kanban (n=296) were the most common responses. Some of the 28 participants who used the free text option were positive, with comments such as *'Good practices but no dogmatic application of any of these.'* The majority of respondents delighted in letting off steam, for example: *'...They claim it's scrum / agile. It is not, it's waterfall with extra meetings,' 'Fake cargo cult agile,'* and *'RDD (Resume-Driven Development).'*

F. Secure Coding Policy

Asked whether their organisation or team has a written secure coding policy, less than a fifth of our participants (n=184) answered *'Yes.'* Other pre-set options were *'No'* (n=537), *'I'm not sure'* (n=238) and *'Not applicable'* (n=65). Testament to the positive impact of security surveys, one free text respondent wrote: *'I'm going to write one today.'*

G. Security Priorities

We asked people what aspects of software security are important. Data protection (n=978), Preventing vulnerabilities (n=954), Customer privacy (n=877) and Customer confidentiality (n=829) were widely chosen. Most other aspects were selected by at least 30% of respondents. Only two respondents chose *"I do not think that software security is important."* There were 49 free text entries comprising a wide range of priorities and experience, usefully summed up by the single entry *'It Depends™.'*

H. Training

We asked participants whether they had ever been offered security or privacy training, and the majority answered *'No'* (see Table II). We then asked for training details, and received over 300 individual free text entries. They ranged from negative comments such as *'Boring'* to more positive feedback: *'Nothing formal. It happens ad hoc (usually directed by me) as a part of code review.'* A reminder of the reluctance of individuals who take security seriously to part with confidential information, several answers were inscrutable, e.g. *"Confidential," "Information refused,"* and *"Internal training."*

TABLE II
SECURITY OR PRIVACY TRAINING

Response	Frequency	Percent
Yes	416	40
No	525	51
Not applicable	98	9

V. RESULTS

One of our survey participants, asked about the importance of security activities, responded that *'External checks are a "finger not the moon" problem, but a useful diagnostic.'* This is a reference to a Buddhist saying on dogma, *'A finger pointing at the moon is not the moon,'* suggesting that it is easy to confuse descriptions of a thing for the thing itself. Given the difficulty in ascertaining whether software is secure, and the comparative simplicity of evaluating lists of software security activities, it is easy to confuse the activities with the goal. Evaluating security culture alongside security activities may get us a little closer to the moon.

A. Research Question 1: What are the secure-coding characteristics of our sample group?

As discussed in our 'Background and Related Work' section, there is no agreed measurement of software security practice in academia. Different groups of academics roll their own based on BSIMM, SAMM, MS-SDL and other practices. BSIMM currently includes a total of 122 activities, with correspondingly large numbers of similar activities in the other methods. This is cumbersome, therefore a synthesised subset is usually generated. Measurement tools are thus not empirically validated, are not easily comparable between studies, and tend not to be reused across studies. The CA Score is motivated by Weir et al. [4], which analysed the BSIMM assessments dataset to study how security activities were introduced in organisational settings over a period of 12 years. Weir et al. found that there were 12 activities that were adopted 'most often, together and first' by a majority of organisations, with at least half of them found in 92% of the assessments. There was 'a marked jump' of almost 20 percent point between the frequency of use of the other developer activities studied, of which the highest frequency was 47% (i.e. less than half), and these 12 activities, which had frequencies from 65% (CAQ5) to 89% (CAQ10).

As described in Sec. III-A1, we asked respondents about the presence of the known 12 most common software security activities [4] in their environment. See Table III for the exact text of each of the 12 questions. We evaluated the number of these CAs that they identified as in use in their working environments (see Fig. 3). We assigned a score of 1-12 to work environments by counting the number of CAs they were undertaking from those 12 we asked about in the survey. We call this the *'CA score.'* This score is empirically justifiable, practical, repeatable, and thus fills a gap in secure coding measurement.

While a small number of participants stated that all 12 common activities were used, there was a concerning number of participants who stated that none or very few of the activities were undertaken in their work context.

B. Research Question 2: What are the security culture characteristics of our sample group?

We asked a series of questions about security culture which can be found in Table IV. The reasoning behind each question can be found in this section.

1) Support for Secure Coding (SCQ1): Support from the organisation has been found to be a significant aspect of security culture in multiple papers [3], [18], [29], [31]. We asked to what extent participants felt supported to code securely (SCQ1). Fig. 4 presents the answers to this question. As can be observed, answers are fairly evenly divided between those who feel supported, those who do not feel supported, and neutral responses. Answers to this question had a weak to moderate correlation of 0.4625 with CA scores, with p<.001.

2) Raising Security Concerns (SCQ2): We asked participants whether, if they had a security concern at work, they would raise that concern. This question is part of our investigation into how security-motivated developers can influence their working environment. Fig. 5 shows that the vast majority of participants would be somewhat or very likely to raise the concern. Those who would not may work in organisations or teams where raising security concerns is actively discouraged, a phenomenon that has previously been observed [18]. This question can be a useful way to detect extremely security unfavourable coding environments. Answers to this question had a correlation of 0.2448 with CA scores, with p<.001.

3) Whether there are Security Tools in the Working Environment (SCQ3): Security tool use is a fundamental aspect of secure coding, mentioned in most of the relevant literature on the subject [49]–[51]. Therefore, we asked whether security tools are present in the developer's environment. See Fig. 6, which shows that 33% of respondents had no security tools present in their working environment. Answers to this question had a low correlation of 0.3815 with CA scores, with p<.001.

4) Introducing Free Tools (SCQ4): Previous research on reasons why developers adopt security tools focused on their motivations and inspirations to do so. In this survey we wanted to explore the level of support for finding and integrating such tools that developers are likely to find within their environment. Xiao et al. [37] noted that several participants in their widely-cited interview study on how developers adopt security tools were self-motivated, seeking out tools because they needed them, and finding them online. Those developers would then need to introduce the tools to their work environment. This process of introduction was not a focus of the Xiao et al. or related Witschey et al. [38] paper, and remained unexplored.

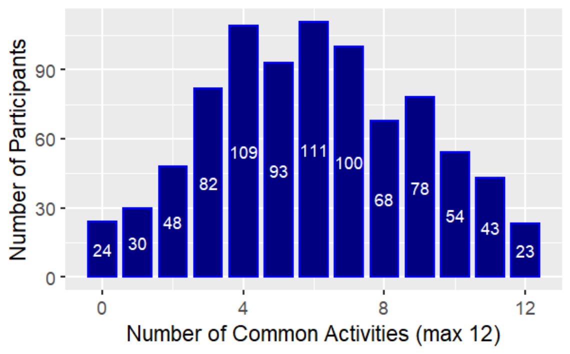

Fig. 3. Number of Security Common Activities in Participants' Environments

TABLE III
QUESTIONS ON TWELVE COMMON ACTIVITIES

No. (Survey No)	Question
CAQ1 (Q47)	Static analysis tools are brought into the code review process to make the review more efficient and consistent.
CAQ2 (Q48)	Compliance constraints are translated into software requirements for individual projects and are communicated to the engineering teams.
CAQ3 (Q49)	QA efforts go beyond functional testing to perform basic adversarial tests and probe simple edge cases and boundary conditions, with no particular attacker skills required.
CAQ4 (Q50)	QA targets declarative security mechanisms with tests derived from requirements and security features. A test could try to access administrative functionality as an unprivileged user, for example, or verify that a user account becomes locked after some number of failed authentication attempts.
CAQ5 (Q51)	Penetration test tools are used internally.
CAQ6 (Q52)	Emergency codebase response can be done. The organisation or team can make quick code and configuration changes when software (e.g., application, API, microservice,CAQ infrastructure) is under attack.
CAQ7 (Q53)	Defects found in operations are entered into established defect management systems and tracked through the fix process.
CAQ8 (Q54)	Bugs found in operations monitoring are fed back to development, and may change developer behaviour. For example, viewing production logs may reveal a need for increased logging.
CAQ9 (Q55)	Penetration testing results are fed back to engineering through established defect management or mitigation channels, with development and operations responding via a defect management and release process.
CAQ10 (Q56)	Host and network security basics are in place across any data centers and networks and remain in place during new releases.
CAQ11 (Q57)	Security-aware reviewers identify the security features in an application and its deployment configuration (authentication, access control, use of cryptography, etc.), and then inspect the design and runtime parameters for problems that would cause these features to fail at their purpose or otherwise prove insufficient.
- (Q58)	(an attention question).
CAQ12 (Q59)	External penetration testers are used to identify security problems.

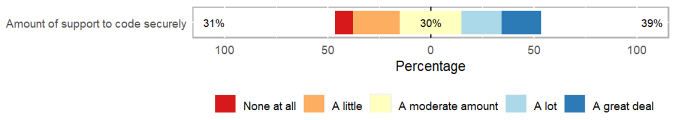

Fig. 4. SCQ1. How much support do you feel that you get from your employer or open source team to code securely?

In a follow-up survey by Witschey et al. [42], the authors expressed surprise that although statements expressing *'security concern'* were predictors for security tool use, they had weak effects. We suspected that this could relate to a difficulty for developers in introducing security tools. In security-conscious environments, there could be a prolonged approval process or

TABLE IV
SECURITY CULTURE QUESTIONS

No. (Survey No)	Question
SCQ1 (Q23)	How much support do you feel that you get from your employer or open source team to code securely?
SCQ2 (Q24)	If you had security concerns about your current project, how likely would you be to raise them with someone?
SCQ3 (Q26)	Security tools are tools that help to ensure source code is free from security vulnerabilities. For example, they may analyse code for known issues or check configurations for problems. Some examples are Coverity, CodeSonar and SonarQube. Some security tools like SonarLint integrate with the IDE. Are there security tools available in your environment?
SCQ4 (Q28)	If you saw a **free** tool that you felt would help you to code more securely, how likely would you be to install and use it **without** asking anyone for permission?
SCQ5 (Q29)	If you needed permission to use the tool, how likely would you be to ask for permission?
SCQ6 (Q30)	If you asked, how likely do you think it is that you would get permission?
SCQ7 (Q31)	If you asked, how likely do you think it is that you would get funding for a **paid** tool?
SCQ8 (Q33)	Have you ever heard people say that there is a software security culture in your working environment?
SCQ9 (Q34)	Would you agree that there is a security culture in your working environment?
SCQ10 (Q35)	How highly do you think your team prioritises software security?
SCQ11 (Q36)	How often is software security mentioned in team communications?
SCQ12 (Q43)	Roughly how much time do you spend on software security in an average week? This would include any tasks aimed at making or keeping a product secure, such as fixing vulnerabilities that could cause a breach, keeping third-party components updated and assessing code for security issues.

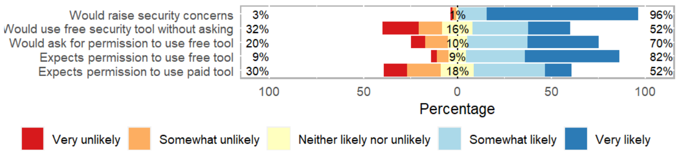

Fig. 5. Results from Security Culture questions SCQ2-SCQ6, concerned with the interplay between the security-conscious individual and their work environment.

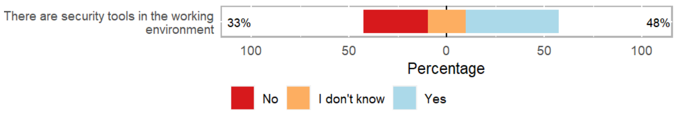

Fig. 6. Security tools are present in the developer's environment (SCQ3)

outright prohibition on new practices and tools. In security-hostile environments, they could be seen as a waste of time. The interaction of security-motivated developers with their environment influences their security activities [41], [52], yet the degree of support or resistance they are likely to encounter has not been investigated. Therefore we asked several questions in our survey about attempting to introduce new security tools. Questions SCQ4-SCQ7 explore how developers would go about introducing such tools, and what reaction they anticipate they would get within their environment. Developers' subjective expectations are important, since a perception that tools would be rejected could result in their not seeking out or proposing them [32]. SCQ4 asked whether the respondent would use a free security tool without asking for permission. See Fig. 5 for the results; 52% of those surveyed would use such a tool, while 32% would not. Note that while it might be expected that security-aware organisations would encourage experimentation with security tools, use of a free tool could itself introduce security vulnerabilities. Ironically, many security tools run with elevated privileges and can be sources of risk. Answers to this question did not correlate at all with CA scores, with a value of -0.0624 and p = 0.067.

5) Asking for Tool Permissions (SCQ5-SCQ7): In SCQ5, we asked participants how likely it was that they would ask for permission to use a free tool (see Fig. 5). Those selecting *'Not Applicable'* were discounted. 70% of respondents would ask, 10% were neutral and 20% were unlikely or very unlikely to ask. Thus, 20% of respondents apparently have a particularly low interest in security, particularly low expectations from their management team, or both. (Answers to this question did not correlate with CA scores, with a value of 0.1405 and p<.001.)

However, SCQ6 indicated that a large 82% of respondents thought that it was somewhat or very likely that, if they did ask, they would get permission for a **free** tool. The figure for a **paid** tool (SCQ7) was much lower at 52%. It can be argued that paid security tools are often better supported than free tools, so that they should be preferred. The only negative difference between a free and paid tool is financial outlay. The contrast between the answers to these two questions may indicate a poor security culture in the relevant participants' organisations. Haney et al. [3] included spending money on security in their list of attributes needed for a positive security culture in a development environment. The signals and cues received by employees influence their understanding of the real priorities in their work environments [29], [32]. In this case, participants have concluded that security is not a management priority.

Answers to SCQ6 did not correlate at all with CA scores, with a value of 0.1202 and p<.001, but SCQ7 answers showed a weak correlation of 0.2718 (p<.001), indicating that paid tools are slightly more likely to be considered if other security precautions are in place.

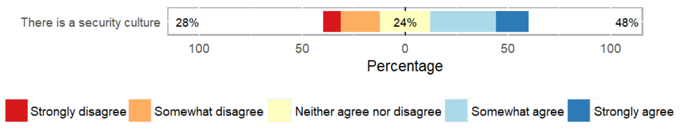

Fig. 7. SCQ9 Whether participant agrees that there is a security culture in their working environment.

6) Perceptions of Security Culture (SCQ8-SCQ9): Although the importance of security culture is emphasised in numerous papers on organisations and teams with secure coding processes [21], [31], [53], [54], there can be a disconnect between management and developers when it comes to security [21]. Some researchers have found that developers can be cynical about management drives for a security culture [54], perhaps perceiving them as lip service. Therefore we first asked whether the developer is aware of a claim to security culture in the environment, and then asked whether the developer would agree with this claim. Asked whether they had heard it said that there was a security culture in their working environment, only 261 respondents said *'Yes'*; 633 selected *'No,'* with 133 selecting *'Not applicable.'* We allowed free text contributions; one was: *'no real people say stuff like that,'* an appropriate reminder that the term *'security culture'* can seem like just more management speak or propaganda [54] if its use is not accompanied by concrete steps towards prioritising security. Another contribution was *'We're a small firm, shipping working software at all is the priority.'* Answers to this question correlated weakly with CA scores, with a value of 0.4029 and p<.001.

The following question (SCQ9) was whether participants would agree that there was a security culture in their work environment. As can be seen in Fig. 7, 48% of participants agreed or strongly agreed with this statement. 28% disagreed or strongly disagreed, while 24% were neutral. This question correlated with the CA score, with a weak to moderate correlation value of 0.4537 (p<.001).

7) How Highly the Team Prioritises Security (SCQ10): Security is often treated as an NFR (non-functional requirement) in the software development process, and is not explicitly prioritised by management or teams [49], [55]. Continuing to probe the security culture in the developers' working environment, we asked them how highly they thought their team prioritised software security. See Fig. 8; while 39% felt that their team prioritised it a lot or a great deal, 25% felt that they prioritised it only a little, or not at all. Correlation between answers to this question and the CA score is low to moderate with a score of 0.4962 (p<.001).

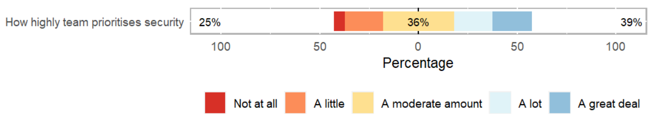

Fig. 8. SCQ10 How Highly the Team Prioritises Security

8) How Often Security is Mentioned in Team Communications (SCQ11): A developer may perceive their team's prioritisation of software security inaccurately, or their reporting may be subject to social desirability bias. Haney et al. [3] discussed how a security culture involves a *'commitment to address security'* and the *'perpetuation of a security mindset.'* Communication of priorities is essential within working environments [31], [56]. Furthermore, if indications of concern for secure coding are missing, that in itself allows developers to infer that secure coding is not considered important on the team [32]. In order to explore security prioritisation further, we asked participants approximately how often software security is mentioned in team communications. The answers can be seen in Fig. 9, and differ markedly from some of the previous answers. 21% of respondents said that security was mentioned on the team about once a week or more often, and 21% said it was mentioned a few times a month. However, a large 58% said it was mentioned about once a month or less, with 9% stating that it is never mentioned. Correlation between answers to this question and the CA score is low to moderate with a score of 0.4488 (p<.001).

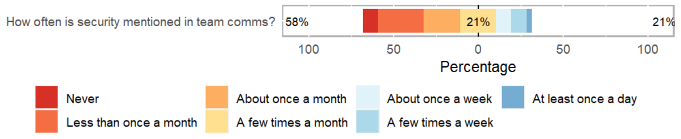

Fig. 9. How Often Team Discusses Security (SCQ11

9) Time per Week Spent on Secure Coding (SCQ12): Time is a scarce resource, and time spent should give further insight into the real emphasis on security in the work environment [32], [57]. We asked participants how much time they spent on security during the week, supplying options from *'None,' 'Less than half an hour'*, etc., to *'A week'*, and also allowing free text replies. This question can be criticised on the basis that security should come with quality. For example, one respondent replied: *'impossible to quantify this way; all work that leads to higher quality software also usually leads to more secure software'*. However, developing secure software entails many security-specific activities; threat modeling is considered an essential component of secure coding [58]–[60]. Security conscious respondents could be expected to have an approximate mental model of how much time they spend on concerns that are primarily security motivated. Therefore, answers to this question are of interest, giving us additional insight into the security culture in the developer's working environment.

Even bearing in mind the qualification above, the answers to this question are not encouraging. See Figure 10, which is dominated by red, orange and yellow blocks. Less than two hours a week is spent on security by 55% of respondents, with nearly 20% spending none. One free text response was: *'Fluctuates on whether my project is blocked by security review. Usually not at all, sometimes a few hours.'* Another participant added *'less than half an hour a month.'*

Correlation between answers to this question and the CA score is low with a score of 0.3961 (p<.001).

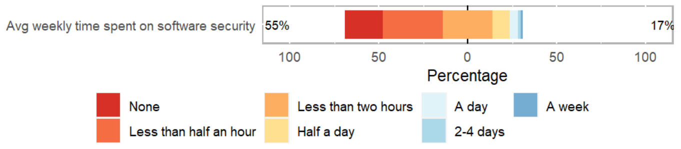

Fig. 10. SCQ12 Time Spent on Security per Week

C. Research Question 3: Do secure coding practice and culture correlate, and if not, what lessons can we learn to help support development of secure coding?

As we examined the answers to our security culture questions we noted the correlation score of each question's answers to the overall CA score for software security practice. In many cases, there was no correlation. Where it did exist it was weak to moderate. By contrast, if good security practice entailed good security culture we would expect to find high correlations between culture answers and practice findings. In an attempt to attain a greater understanding of our results, we broke down some of the answers by CA score. Fig. 11 shows the degree to which the participant thinks their team prioritises security, broken down by CA score. There is clearly a correlation, and for high CA scores we are led to believe that security is very highly prioritised by the team.

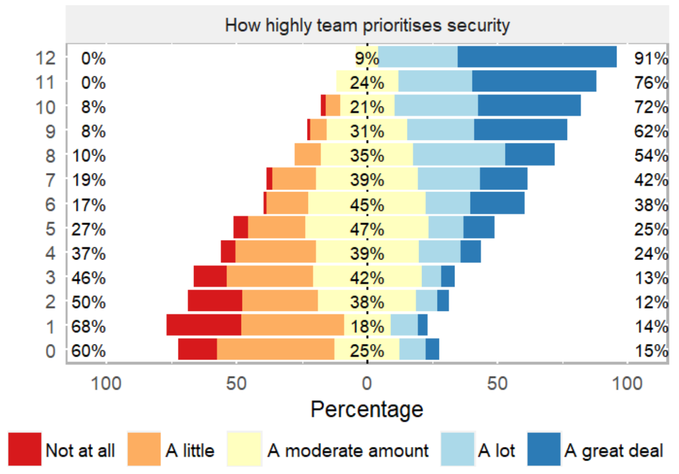

Fig. 11. SCQ10 How Highly Participant Thinks Team Prioritises Security. This graph shows the value broken down by the number of common security activities undertaken in the environment. The correlation is visible.

However, when we look at how frequently the team communicates about security, a prioritisation of security is not at all clear. See Fig. 12. Even in the teams with the highest security practices, security is mentioned within the team relatively infrequently. At the very highest level of twelve CAs, security is mentioned less than once a week in 36% of teams. Low levels of communication about a topic is an indicator that the topic is low-priority for a team [32]. We suggest that this question gives some insight into environments where, although security is apparently high-priority, the security climate is unfavourable. The minimum work necessary for security compliance is done.

Similarly, when we look at the amount of time spent on security per week it is surprisingly low at all CA levels. At the zero-CA level, 95% of respondents spend less than half an hour a week on security-related activities, with two thirds of these choosing *'None.'* Even at the highest level, 59% of participants spend an average of less than two hours per week on security activities. Fig. 13 suggests that this question could be useful when attempting to identify a compliance-focused mentality.

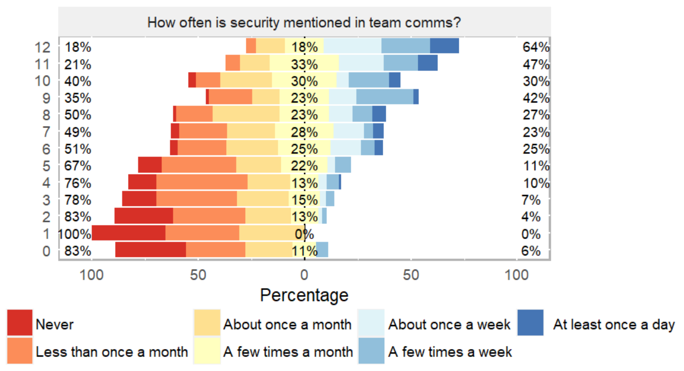

Fig. 12. SCQ11 How Often Security is Mentioned in Team Communications. This graph shows the value broken down by the number of common security activities undertaken in the environment. Even at the highest CA scores, team security communication is relatively infrequent.

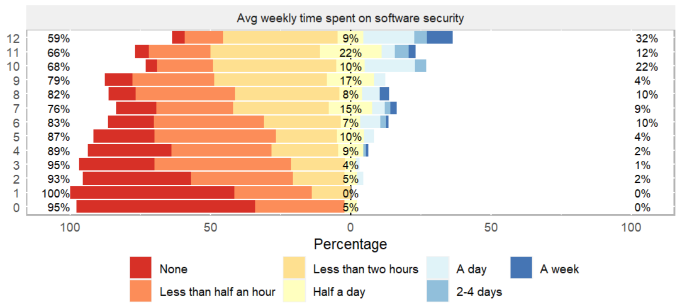

Fig. 13. SCQ12 Time Spent on Security per Avg Week. Even at the highest level of common security activities, many participants spend less than two hours a week on security.

VI. Discussion

Asked to comment on the survey, one participant said of the 12 CAs: *'A lot of the questions about bugs or security issues being tracked are kind of missing the point; the system exists, but often times fails. Bugs and bug fixes are in the correct system, but there is not enough time allocated for reporters to verify or validate fixes or for regression.'*

This is an excellent illustration of the importance of assessing security culture alongside security activities. If the organisation's employees are going through the motions, producing paperwork and audit trails for compliance purposes, but are not allowed time to do the job properly, a good security culture is missing. In this environment it is inevitable that security issues will slip through the cracks, adversely affecting the organisation or open source team, impacting their clients, and sometimes posing a systemic threat to their very existence.

A. Threats to Validity

The questions we used to objectively assess environmental secure coding practice are based on the secure coding activities identified by analysis of BSIMM results as those adopted first and most frequently by security-aware organisations. These activities might not be suited or appropriate to all coding environments. However, the BSIMM data is the largest trove of such data available at this time. It is the closest approach we have identified to finding an empirically-established objective measure of secure coding in environments developing complex applications.

The questions used to provide insight into participants' understanding of the security culture of their working environment were based on ideas that can also be found in organisational climate theory. The importance of the topics that we asked about can be determined by their prevalence in academic software security literature. It is possible that other wordings or additional questions exist which would further enhance our understanding of security culture. Finding and evaluating such questions should be the subject of further research.

As with all questionnaire-based studies, another risk to validity is social desirability bias – the danger that developers will tell us what they think we want to hear. Our respondents were self-selecting, and the data cannot be verified. We attempted to mitigate this risk by emphasising in recruitment materials and in the survey consent form that participants did not need to be interested in security and did not need to know anything about it.

Our interpretations of the answers to our questions could be mistaken. This is a consequence of the fact that it is not possible to statistically assess the actual security of software, for reasons discussed previously. In a discussion of assessing software security in the field, it is necessary to bear in mind that the direction of causation cannot be determined. However, correlations can be observed and can be instructive, which is the approach we have taken here.

VII. Conclusion

In this study we add to the existing body of knowledge on how to identify environments where secure coding is prioritised and interest in code security is valued.

We wish to ensure that the correct thing is being measured. In our research, we want to focus on the moon, and not on the finger pointing at the moon.

We adopted a list of 12 secure coding activities which have been empirically established as being those adopted first and most often in software security drives. We found that while many of our respondents identified some of these activities in their work environments, in some workplaces none of them were present.

It is well known that security activities can be undertaken for compliance reasons. Researchers into secure coding have established that to achieve secure coding success, a security culture should be established. Based on a thorough literature review, we asked a number of questions designed to evaluate security culture. For example, we asked how much time the respondent spends on secure coding, and how often, on average, secure coding is mentioned in team communications in a week. In this paper we discuss the answers to 12 such questions. We find indications of poor security culture at all levels of security practice. Even where participants state that all 12 common security practices are undertaken in their working environments, communication about security and time spent on security can be very low.

We expect that our results will have an impact in several areas.

Academia: Research on software security practice does not always attempt to measure organisational security [39], and even when this is measured, culture is not quantified. When studying developer behaviour, for example in ethnographic studies, it is important to also consider the security processes and culture in the environment. Otherwise the research may miss fundamental influences on the developer or team under study. The CA Score provides a simple but empirically validated measure for practice, and our culture questions introduce a way to also assess culture.

Organisations: For organisations that are serious about encouraging secure coding, it is not enough to introduce practices and follow processes. In addition, time, budget and space for security must be provided.

Industry: Changes to regulation, law and industry standards must look beyond the checkbox approach [23] and consider the holistic background.

The results of this survey show that when evaluating security posture it is not enough merely to measure activities. Security culture must also be evaluated. Understanding the time, money and support available for secure coding activities is crucial to assessing how thoroughly they will be implemented.

A. Data Availability

We have supplied the following in the supplementary material: a spreadsheet containing the text of all 62 questions in our survey, a spreadsheet giving the answers to the questions relevant to this paper, some R scripts, and a PDF containing an appendix.

References

[1] G. McGraw, "Software security," *IEEE Security and Privacy*, vol. 2, no. 5, p. 80–83, 2004.

[2] K.-J. Stol and B. Fitzgerald, "The ABC of software engineering research," *ACM Transactions on Software Engineering and Methodology*, vol. 27, no. 3, p. 1–51, Sep 2018.

[3] J. Haney, M. Theofanos, Y. Acar, and S. Spickard Prettyman, "'We make it a big deal in the company': Security mindsets in organizations that develop cryptographic products," in *Proceedings of the Fourteenth Symposium on Usable Privacy and Security*. USENIX Association, 2018, p. 357–373. [Online]. Available: https://csrc.nist.gov/publications/detail/conference-paper/2018/08/12/security-mindsets-in-organizations-that-develop-crypto-products

[4] C. Weir, S. Migues, M. Ware, and L. Williams, "Infiltrating security into development: exploring the world's largest software security study," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2021, p. 1326–1336. [Online]. Available: https://dl.acm.org/doi/10.1145/3468264.3473926

[5] A. Naiakshina, A. Danilova, C. Tiefenau, M. Herzog, S. Dechand, and M. Smith, "Why do developers get password storage wrong? A qualitative usability study," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. Association for Computing Machinery, 2017, p. 311–328. [Online]. Available: http://doi.org/10.1145/3133956.3134082

[6] A. Naiakshina, A. Danilova, E. Gerlitz, E. von Zezschwitz, and M. Smith, "'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*, ser. CHI '19. Association for Computing Machinery, May 2019, p. 1–12. [Online]. Available: http://doi.org/10.1145/3290605.3300370

[7] A. Naiakshina, A. Danilova, E. Gerlitz, and M. Smith, "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*, ser. CHI '20. Association for Computing Machinery, 2020, p. 1–13. [Online]. Available: https://doi-org.ucc.idm.oclc.org/10.1145/3313831.3376791

[8] M. G. Jaatun, "Hunting for aardvarks: Can software security be measured?" in *Multidisciplinary Research and Practice for Information Systems*, ser. Lecture Notes in Computer Science, G. Quirchmayr, J. Basl, I. You, L. Xu, and E. Weippl, Eds. Springer, 2012, p. 85–92.

[9] C. Weir and B. Hermann, "From needs to actions to secure apps? The effect of requirements and developer practices on app security," in *29th USENIX Security Symposium, August 12, 2020 - August 14, 2020*, ser. Proceedings of the 29th USENIX Security Symposium. USENIX Association, 2020, p. 17.

[10] M. W. Sammy Migues, John Steven, "BSIMM," https://www.bsimm.com/, 2022.

[11] OWASP, "SAMM," https://www.opensamm.org/, 2021, [Online; accessed 13-April-2021].

[12] Microsoft, "Microsoft Security Development Lifecycle," https://www.microsoft.com/en-us/securityengineering/sdl/, 2022, [Online; accessed 01/09/2022].

[13] SAFECode, "SAFECode Homepage," https://safecode.org/, 2022, [Online; accessed 01/09/2022].

[14] M. G. Jaatun, D. S. Cruzes, K. Bernsmed, I. A. Tøndel, and L. Røstad, "Software security maturity in public organisations," in *Information Security*, ser. Lecture Notes in Computer Science, J. Lopez and C. J. Mitchell, Eds. Springer International Publishing, 2015, p. 120–138.

[15] P. Morrison, "A security practices evaluation framework," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ser. ICSE '15. IEEE Press, 2015, p. 935–938.

[16] T. D. Oyetoyan, D. S. Cruzes, and M. G. Jaatun, "An empirical study on the relationship between software security skills, usage and training needs in agile settings," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 2016, p. 548–555.

[17] K. Rindell, J. Ruohonen, and S. Hyrynsalmi, "Surveying secure software development practices in Finland," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ser. ARES 2018. Association for Computing Machinery, 2018, event-place: Hamburg, Germany.

[18] H. Assal and S. Chiasson, "Security in the software development lifecycle," in *Proceedings of the Fourteenth USENIX Conference on Usable Privacy and Security*, ser. SOUPS '18. USA: USENIX Association, 2018, p. 281–296.

[19] D. Votipka, D. Abrokwa, and M. L. Mazurek, "Building and validating a scale for secure software development self-efficacy," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20. Association for Computing Machinery, 2020, p. 1–20, event-place: Honolulu, HI, USA.

[20] P. Morrison, D. Moye, R. Pandita, and L. Williams, "Mapping the field of software life cycle security metrics," *Information and Software Technology*, vol. 102, no. May, p. 146–159, 2018.

[21] D. Ashenden and D. Lawrence, "Security dialogues: Building better relationships between security and business," *IEEE Security Privacy*, vol. 14, no. 3, pp. 82–87, 2016.

[22] J. A. Morales, T. P. Scanlon, A. Volkmann, J. Yankel, and H. Yasar, "Security impacts of sub-optimal devsecops implementations in a highly regulated environment," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20. Association for Computing Machinery, 2020, p. 8.

[23] S. Rahaman, G. Wang, and D. D. Yao, "Security certification in payment card industry: Testbeds, measurements, and recommendations," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, Nov 2019, p. 481–498. [Online]. Available: http://doi.org/10.1145/3319535.3363195

[24] "PCI DSS," https://www.pcisecuritystandards.org/document_library/?category=pcidss&document=pci_dss, [Online; accessed 26 July 2022].

[25] C. Heitzenrater and A. Simpson, "A case for the economics of secure software development," in *Proceedings of the 2016 New Security Paradigms Workshop*, ser. NSPW '16. New York, NY, USA: Association for Computing Machinery, Sep 2016, p. 92–105. [Online]. Available: http://doi.org/10.1145/3011883.3011884

[26] I. Rauf, M. Petre, T. Tun, T. Lopez, P. Lunn, D. Van der Linden, J. Towse, H. Sharp, M. Levine, A. Rashid, and et al., "The case for adaptive security interventions," *ACM Transactions on Software Engineering and Methodology*, p. (In Press), 2021.

[27] I. Pashchenko, D.-L. Vu, and F. Massacci, "A qualitative study of dependency management and its security implications," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. Association for Computing Machinery, Oct 2020, p. 1513–1531. [Online]. Available: http://doi.org/10.1145/3372297.3417232

[28] I. A. Tøndel, D. S. Cruzes, M. G. Jaatun, and K. Rindell, "The security intention meeting series as a way to increase visibility of software security decisions in agile development projects," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi-org.ucc.idm.oclc.org/10.1145/3339252.3340337

[29] H. Assal and S. Chiasson, "'Think secure from the beginning': A survey with software developers," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. Association for Computing Machinery, 2019, p. 1–13. [Online]. Available: http://doi.org/10.1145/3290605.3300519

[30] T. Lopez, H. Sharp, T. Tun, A. K. Bandara, M. Levine, and B. Nuseibeh, "'Hopefully we are mostly secure': Views on secure code in professional practice," in *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '19. IEEE Press, 2019, p. 61–68.

[31] A. Tuladhar, D. Lende, J. Ligatti, and X. Ou, "An analysis of the role of situated learning in starting a security culture in a software company," in *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, 2021, p. 617–632. [Online]. Available: https://www.usenix.org/conference/soups2021/presentation/tuladhar

[32] R. Arizon-Peretz, I. Hadar, G. Luria, and S. Sherman, "Understanding developers privacy and security mindsets via climate theory," *Empirical Software Engineering*, vol. 26, no. 6, p. 34, 2021.

[33] M. Tahaei, A. Frik, and K. Vaniea, "Privacy champions in software teams: Understanding their motivations, strategies, and challenges," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021, p. 16. [Online]. Available: http://doi.org/10.1145/3411764.3445768

[34] I. Ryan, U. Roedig, and K.-J. Stol, "Understanding developer security archetypes," in *International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. Association of Computer Machinery, Aug. 2021.

[35] J. M. Haney and W. G. Lutters, "'It's scary…it's confusing…it's dull': How cybersecurity advocates overcome negative perceptions of security," in *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, 2018, p. 411–425. [Online]. Available: https://www.usenix.org/conference/soups2018/presentation/haney-perceptions

[36] C. Weir, I. Becker, and L. Blair, "A passion for security: Intervening to help software developers," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2021, p. 21–30.

[37] S. Xiao, J. Witschey, and E. Murphy-Hill, "Social influences on secure development tool adoption: Why security tools spread," in *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, ser. CSCW '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 1095–1106.

[38] J. Witschey, S. Xiao, and E. Murphy-Hill, "Technical and personal factors influencing developers' adoption of security tools," in *ACM Workshop on Security Information Workers*. ACM Press, 2014, p. 23–26.

[39] H. Palombo, A. Z. Tabari, D. Lende, J. Ligatti, and X. Ou, "An ethnographic understanding of software (in)security and a co-creation model to improve secure software development," in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, Aug. 2020, pp. 205–220. [Online]. Available: https://www.usenix.org/conference/soups2020/presentation/palombo

[40] W. Foddy and W. H. Foddy, *Constructing Questions for Interviews and Questionnaires: Theory and Practice in Social Research*. Cambridge University Press, 1994.

[41] A. Danilova, A. Naiakshina, and M. Smith, "One size does not fit all: a grounded theory and online survey study of developer preferences for security warning types," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20.

Association for Computing Machinery, Jun 2020, p. 136–148. [Online]. Available: http://doi.org/10.1145/3377811.3380387

[42] J. Witschey, O. Zielinska, A. Welk, E. Murphy-Hill, C. Mayhorn, and T. Zimmermann, "Quantifying developers' adoption of security tools," in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, 2015, p. 260–271.

[43] A. Danilova, A. Naiakshina, S. Horstmann, and M. Smith, "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)*, May 2021, p. 537–548.

[44] M. Tahaei and K. Vaniea, "Recruiting participants with programming skills: A comparison of four crowdsourcing platforms and a CS student mailing list," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–15. [Online]. Available: http://doi.org/10.1145/3491102.3501957

[45] H. Kaur, S. Amft, D. Votipka, Y. Acar, and S. Fahl, "Where to recruit for security development studies from: Comparing six software developer samples," in *31st USENIX Security Symposium (USENIX Security 22)*, Aug 2022, p. 24. [Online]. Available: https://teamusec.de/publications/conf-usenix-kaur22/

[46] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde, "What predicts software developers' productivity?" *IEEE Transactions on Software Engineering*, p. 1–23, 2019.

[47] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2022. [Online]. Available: https://www.R-project.org/

[48] G. Norman, "Likert scales, levels of measurement and the "laws" of statistics," *Advances in Health Sciences Education*, vol. 15, no. 5, p. 625–632, Dec 2010.

[49] J. Xie, H. R. Lipford, and B. Chu, "Why do programmers make security errors?" in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2011, p. 161–164.

[50] O. Pieczul, S. Foley, and M. E. Zurko, "Developer-centered security and the symmetry of ignorance," in *Proceedings of the 2017 New Security Paradigms Workshop on ZZZ - NSPW 2017*. ACM Press, 2017, p. 46–56. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3171533.3171539

[51] T. W. Thomas, M. Tabassum, B. Chu, and H. Lipford, "Security during application development: An application security expert perspective," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–12.

[52] I. Rauf, D. van der Linden, M. Levine, J. Towse, B. Nuseibeh, and A. Rashid, "Security but not for security's sake: The impact of social considerations on app developers' choices," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. Association for Computing Machinery, 2020, p. 141–144, event-place: Seoul, Republic of Korea. [Online]. Available: https://doi-org.ucc.idm.oclc.org/10.1145/3387940.3392230

[53] M. Tahaei, A. Jenkins, K. Vaniea, and M. Wolters, "'I don't know too much about it': On the security mindsets of computer science students," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019, pp. 350–350.

[54] N. Tomas, J. Li, and H. Huang, "An empirical study on culture, automation, measurement, and sharing of DevSecOps," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Jun 2019, p. 1–8.

[55] M. Tahaei and K. Vaniea, "A survey on developer-centred security," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE. IEEE, 2019, pp. 129–138.

[56] A. Poller, L. Kocksch, K. Kinder-Kurlanda, and F. A. Epp, "First-time security audits as a turning point?: Challenges for security practices in an industry software development team," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '16. Association for Computing Machinery, 2016, p. 1288–1294, event-place: San Jose, California, USA. [Online]. Available: https://doi-org.ucc.idm.oclc.org/10.1145/2851581.2892392

[57] D. Oliveira, M. Rosenthal, N. Morin, K.-C. Yeh, J. Cappos, and Y. Zhuang, "It's the psychology stupid: how heuristics explain software vulnerabilities and how priming can illuminate developer's blind spots," in *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM Press, 2014, p. 296–305. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2664243.2664254

[58] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek, and S. Fahl, "Developers need support, too: A survey of security advice for software developers," in *2017 IEEE Cybersecurity Development (SecDev)*. IEEE, Sep 2017, p. 22–26. [Online]. Available: http://ieeexplore.ieee.org/document/8077802/

[59] C. Weir, I. Becker, J. Noble, L. Blair, M. A. Sasse, and A. Rashid, "Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers," *Software - Practice and Experience*, vol. 50, no. 3, p. 275–298, 2020.

[60] J. Whitmore and W. Tobin, "Improving attention to security in software design with analytics and cognitive techniques," in *2017 IEEE Cybersecurity Development (SecDev)*, 2017, p. 16–21.