"ImmediateShortTerm3MthsAfterThatLOL": Developer Secure-Coding Sentiment, Practice and Culture in Organisations

Ita Ryan
Lero and School of Computer Science and Information Technology
University College Cork
Ireland
iryan@ucc.ie

Utz Roedig
School of Computer Science and Information Technology
University College Cork
Ireland
u.roedig@ucc.ie

Klaas-Jan Stol
Lero and School of Computer Science and Information Technology
University College Cork
Ireland
k.stol@ucc.ie

***Abstract*—As almost all areas of human endeavour undergo rapid digital transformation, secure coding is increasingly important to personal, commercial and national security. Yet studies have shown that software developers do not always prioritise or even understand security. Our large survey of organically sourced coders (n=863) examines how software developers currently experience secure coding in the workplace. We found that developers express an interest in secure coding, display basic security knowledge, and turn to their managers and teams first for help with security concerns. We found that developer secure coding sentiment and security practice do not correlate with organisational statistics such as size, but do correlate weakly with measures of security culture, indicating that organisational security support goes hand-in-hand with secure development. Most developers would look for help in-house if they had security concerns. Investigating the effects of code breaches, we found that for almost half of cases, code security does not increase, or increases only for a short time.**

I. Introduction

Software security as a concern becomes more pressing as the world becomes more connected [1]. At the heart of software development, and therefore of software security, is the developer. Yet we have little understanding of how developers currently experience the need for secure coding. We do not know if the upsurge in security concerns in recent years, the normalisation of ransomware, and the increasing newsworthiness of cybersecurity incidents have increased developers' interest in coding securely. We do not have insight into how developers experience secure coding in their workplace. We do not know whether organisations provide sufficient and appropriate secure-coding support to developers.

To address these questions, we ran a global online developer survey. We organically sourced 1,100 respondents, obtaining 863 valid responses from developers working in organisations. Our research questions are as follows:

Research Question 1*: Do developers working for organisations have a basic level of software security knowledge?* We asked three questions on basic knowledge.

Research Question 2*: How do developers working for organisations currently feel about secure coding, and about security practice, culture and support in the organisation?* This investigation explored developers' sentiments about secure coding, and their experience of secure coding in organisations.

Research Question 3*: How does a security breach affect the secure coding focus of individuals and teams in the short and long term?* The issue of whether software breaches affect secure development in organisations is undecided; we added evidence from the survey.

This paper offers an overview of developers' secure coding sentiment at this point in time. We discover that desirable secure-coding attributes such as interest in and enjoyment of security are not correlated to organisation size, team size or the number of developers in an organisation. They do, however, tend to correlate with security-focused work environment attributes such as prioritisation of software security. We find that experiencing a software breach can cause over 50% of developers and organisations to improve their security stance in the long term, but that in some cases improvements are short-term, and breaches are sometimes ignored.

The paper is organised as follows: Section II discusses background and related work. Section III outlines our survey methodology. Section IV describes the sample. Section V provides the answers to the research questions. Section VI discusses the implications and limitations of the results. Section VII concludes the work.

II. Background and Related Work

Gary McGraw began the *'Building Security In'* series in 2004 to address issues with software security [2]. Two decades later, security in software development remains a central cybersecurity concern. At its core is the developer. Discussion about software development in a security context is often referred to as Developer Centred Security (DCS), which *'aims to understand the context in which developers produce security-relevant code as well as provide tools and processes that better support both developers and secure code production'* [3]. In discussing software *'developers,'* we are concerned in this paper only with those actually writing code, and do not include

testers, managers and those in other related roles. Writing code is the one indispensable process in the business of producing software. This central role is changing; Pieczul et al. [4] suggested that developers could increasingly be thought of as aggregators who assemble various pre-existing components into new arrangements and bind them together with scripts. However, this is an oversimplification. Software development remains a challenging, inventive task.

A. Developers' Secure-Coding Perception

Researchers have used surveys to assess developers' security blockers and motivations [5, 6], their understanding of the security implications of their decisions on their applications [7], their opinions of the move to DevOps [8], their app-update motivations [9], their Stack Overflow use [10, 11], and more.

Like maintainability, software security is a non-functional requirement (NFR) and it is possible to release successful software without it. In contrast, the software's core functionality must usually be in place before the software is released. Studies have shown that the presence of NFRs in code is often taken for granted by management, who do not explicitly ask for them [3]. Therefore, developers' own attitudes to security may have a strong effect on the security of code produced. Lopez et al. found that the organisation putting secure-coding practices in place is a motivator for developers [12]. Assal and Chiasson looked at what motivates developers to code securely, finding that identifying with security importance is a strong motivator [5]. Poor tools [13, 14], lack of security knowledge [15–18], and a perception that security is scary and dull [19] all inhibit security adoption. There can be other motivators, and perverse incentives; Palombo et al. worked with developers who were inclined to leave certain vulnerabilities in place because it saved them support calls [20]. Acar et al. devised a research agenda to introduce usable security thinking to the software development space [21]. They advocated for a systematic approach to understanding developers' needs and improving the usability of their tools, arguing that this would improve the security of the entire software ecosystem.

B. The Impact of the Developer's Workplace

Management issues can be at the root of lack of attention to secure coding [22], with unclear and conflicting priorities, the absence of a coherent security stance [20], and a lack of resources all contributing to developer security inattention [5, 23]. Weir et al. found that the organisation was both the biggest motivator and very much the biggest blocker mentioned by their study participants in exit interviews [6]. This illustrates the extent to which a developer's secure coding stance can be affected by their working environment; unsurprising when management usually controls access to resources such as time and budget without which it is difficult for a developer to act [24]. Haney et al.'s interview study with multiple organisations working on cryptography solutions found that these organisations were dedicated to security, with a focus on mentoring, maturity, and *'rigorous development and testing practices'* [19]. This impressively meticulous approach was not echoed in studies of less security-focused organisations. Assal and Chiasson encountered a developer who had not seen any security practices in her work environment; when she queried this she was told *'well, that's how the company is. Security is not something we focus on right now'* [15]. Morales et al. reported an ethnographic study of a multi-year large development project with multiple subcontractors, supposedly using DevSecOps [25]. The implied emphasis on security was surface-level only. Subcontractor incentives were not aligned to security and subcontractors were forced to compete for time on the DevSecOps pipeline, which they could not replicate in-house because it was poorly implemented, badly documented, and not kept up-to-date in source control. Morales et al.'s account showed that a security-oriented development method cannot compensate for the damage caused by bad management and a failure to focus on security.

Meanwhile, solo developers such as some app developers are spared issues with bad management, but are forced to make difficult decisions with limited or no support [7]. Secure development methods tend to make assumptions about the availability of large teams and resources. There are few or no methods tailored to small organisations [26].

C. Evaluation of Developers' Security Skills

A series of studies by Naiakshina et al. examined whether developers coded securely when coding passwords, both when primed to do so and when not primed [27–29]. They found strong evidence that most developers will not consider security unless asked. These studies were laboratory experiments [30]. They took place in a controlled environment, using the known best solution to a specific short coding task to evaluate the security of developers' code. However, most useful software is complex. This measurement technique is not scalable to larger projects. The evaluation is labour-intensive, and when complexity is introduced it becomes difficult to verify that no security vulnerability is present and that the coding solution is optimal. The innovative *'Build It, Break It, Fix It'* project crowd-sourced the security checks, asking study participants to attack one another's code [31]. Unfortunately, not all study participants participated in the breaking or fixing phases.

A different procedure is called for when attempting to evaluate developers' security expertise at scale, for example as part of a survey. Votipka et al., observing that self-efficacy has been shown to correlate with skill in other contexts, introduced the secure software-development self-efficacy scale (SSD-SES) [32]. They suggested that this scale, based on 15 questions evaluating self-efficacy, could be used in a before-and-after sense to assess the value of an intervention, or as a covariate allowing researchers to control for secure development skill. The scale was thoroughly assessed and evaluated, however, it did not detect consistent self-efficacy improvement during the *'Build It, Break It, Fix It'* series of studies.

D. Security Tools and APIs

Static analysis and other tools are key to harnessing security expertise and making it available to developers in a routine way.

Sadowski et al. described how an enterprise-wide static analysis tool was introduced in Google, allowing teams to define rules for static analysis which could be local or shared [33]. Getting tooling right is not a simple task and may involve trial and error. Once a suitable model has been adopted, however, the benefits are enormous. The authors listed a number of high-profile issues that would have been avoided if the enterprises involved had been using the Google static analysis tools. Haney et al. found that all of the 21 highly security-focused organisations they studied had formal security processes for the software development lifecycle including threat modelling, pen testing, and so on [19]. However, the industry-wide move to DevOps means that such processes, if not available in automated form, may be left behind [8, 34]. Thus tool information [16], adoption [35], and usability [36] are increasingly important to secure software development. Some researchers have used gamification to increase tool usability [37].

As with tools, the design of libraries is important. Green and Smith gave ten rules for designing usable cryptography APIs [13]. Gorski et al. integrated security warnings directly into cryptography APIs and found that 73% of study participants who received the advice fixed the code [38]. Warnings in the IDE have also received attention; context is key and the ability to customise warnings is highly valued [39]. Studies have consistently found that APIs are badly designed [40] and lack vital information in documentation [41, 42].

III. Methods

We undertook a large cross-sectional survey to investigate current software security practice and culture. The survey included questions on demographics and on participants' personal relationship with secure coding. It followed with questions about secure coding practices and culture in participants' working environments. In prior work based on the survey, we explored measurements of secure coding practice and culture, and the relationship of secure coding to security culture [43]. We found that in many coding environments, few or no secure coding practices are undertaken. Security culture can appear to be poor even in environments with high levels of secure coding practice. We made secure-coding training recommendations and assessed training practice based on the survey's training data [44]. We explored secure-coding practice further for developers coding in non-standard environments, focusing on levels of practice among isolated developers, and those coding outside organisations [45].

The focus of this paper is specifically on the *organisational* working environment. It uses the survey data to assess the secure coding sentiments of developers who work in organisations, and how developer sentiment correlates with measurements of software security practice and culture in the organisation. Therefore, we filtered the 962 valid survey responses to include only those from people working in organisations with more than a single employee. This gave a sample of 863.

We further develop analysis of the security culture measurement questions introduced in prior work, using factor analysis to distil twelve questions to two distinct factors and explore their relationship to developer sentiment.

A. Survey Process

The survey was approved by our institution's ethics review board. After two stages of piloting it was made available online. We initially promoted it via personal contacts, receiving few responses. We therefore worked daily for three months to secure responses, promoting the survey with developer communities on social media. This resulted in 1,100 entries. The screening process, agreed before data collection, excluded non-coders and screened for coding aptitude and for attention. There were 962 valid respondents. Further details of the data collection process, the screening process, and the ethics procedure are available in prior work [43, 44].

In order to avoid the forced choice bias imposed by limiting responses to set values [46], we allowed free text entries at many points during the survey. These free text spaces were used by respondents to provide us with context, to explain why their answers might appear contradictory, and to comment on the survey. Respondents also used them to elaborate on what was excellent about the security posture in their working environment. Where security procedures were far from excellent, respondents described them with gusto and humour. The result was that the free text comments were both informative and enjoyable to read. We have included many of them in this paper, to provide readers with a flavour of the rich body of data in the survey results. Respondents' entries are quoted exactly as they were entered by the respondents. We do not use '*sic*.'

B. Data Analysis

R Statistical Software (v4.4.1) was used for statistical analysis [47]. The *'likert'* package was used for data visualisation [48], the *'psych'* package was used for factor analysis [49], and the *'corrr'* package was used for statistical analysis [50]. Spearman's rho was used for correlations. A total of 32 correlations were tested, implying a Bonferroni-corrected maximum p-value of 0.0016 ($0.05\backslash 32$) to attain statistical significance. For added caution, we checked all p-values against a lower value of 0.001 ($0.05\backslash 50$). We found that all correlations that were significant at the Bonferroni level were also significant at this lower level. A replication package for R is available in the supplementary material [51].

Since many questions allowed free text answers, a small amount of coding was done on qualitative data by the first author. In most cases the length of the data analysed was a sentence or fragment. Therefore, coding by a single author was considered sufficient.

IV. Sample Description

In order to study developers' relationship with their work environment, we included in the sample only developers who work in organisations with more than one employee (n=863). The full dataset is described in detail in prior work [43]. Below we give figures for the sample used in this paper.

TABLE I
AGE DISTRIBUTION OF THE SAMPLE. DUE TO ROUNDING, TOTAL % IS LESS THAN 100. SINCE QUESTIONS WERE NOT COMPULSORY, 'N' VARIES BETWEEN QUESTIONS. HERE ALL RESPONDENTS ANSWERED: N=863.

Age	Frequency	Percent
18-24	104	12.1
25-34	335	38.8
35-44	280	32.4
45-54	116	13.4
55-64	26	3.0
65+	2	0.2

TABLE II
GENDER DISTRIBUTION OF THE SAMPLE. (N=861)

Gender	Frequency	Percent
Man	721	83.7
Woman	70	8.1
Non-binary	47	5.5
Prefer not to say	23	2.7

TABLE III
LOCATION DISTRIBUTION SHOWING THE TOP TEN COUNTRIES IN THE SAMPLE. WE RECEIVED ENTRIES FROM ALL CONTINENTS EXCEPT ANTARCTICA. (N=846)

Country	Frequency	Percent
United States	387	45.7
United Kingdom	89	10.5
Germany	73	8.6
Canada	56	6.6
France	28	3.3
Australia	24	2.8
Netherlands	20	2.4
Ireland	16	1.9
Sweden	11	1.3
Switzerland	10	1.2
Other	132	15.6

The age of respondents varied from 18 to over 65 (see Table I). 83.7% of respondents identified most with the gender *'Man'* and 8.1% with '*Woman*,' close to the numbers found in industry (see Table II). We asked developers what country they live in, allowing them to add their country if it was missing (see Table III). A complete breakdown of respondents' countries can be found in the supplementary material [51].

Asked about the development process, participants mentioned a total of 73 programming languages and 128 other development tools. Asked if they used secure-coding tools, only 418 participants chose *'Yes'*; however, 494 reported one or more security tools that were present in their environment. Free text answers mentioned in-house tools and also at least one unscalable solution: '*Experienced programmer's brain*.' One respondent was too security-conscious to answer the question, replying instead '*Information refused*.' Table IV lists the most frequently mentioned programming languages, other technologies, and secure-coding tools.[1]

TABLE IV
PROGRAMMING LANGUAGES, TOOLS AND SECURE-CODING AIDS. N VARIES BY QUESTION AND IS IN THE HEADINGS BELOW.

Language, tool or aid	Frequency	Percentage of n
Programming Languages (n=862)		
JavaScript	373	43.3
Python	360	41.8
HTML/CSS	270	31.3
Bash/Shell	269	31.2
SQL	259	30.0
TypeScript	210	24.4
C#	176	20.4
Java	167	19.4
C	163	18.9
C++	151	17.5
Node.js	138	16.0
PHP	82	9.5
Other Technologies (n=851)		
Git	804	94.5
Docker	470	55.2
AWS	295	34.7
PostgreSQL	293	34.4
MySQL	178	20.9
React.js	187	22.0
SQLite	169	19.9
.NET Core / .NET 5	143	16.8
.NET Framework	122	14.3
SQL Server	118	13.9
Kubernetes	115	13.5
Microsoft Azure	112	13.2
Yarn	112	13.2
NumPy	108	12.7
jQuery	98	11.5
Pandas	85	10
DigitalOcean	81	9.5
Express	71	8.3
Google Cloud Platform	65	7.6
Secure-Coding Tools and Aids (n=494)		
SonarQube	174	35.2
Clang	135	27.3
Snyk OS	87	17.6
Coverity	73	14.8
SonarLint	62	12.6
Fortify	40	8.1
Black Duck	34	6.9
Checkmarx	30	6.1
CodeSonar	25	5.1
Veracode	22	4.5
CodeQL	8	1.6

We asked about development methods and got 757 responses (see Table V). The free text option was used by 22 participants, who added only three documented methods not included in the survey list; *'Shape Up'*, which had two mentions, and *'Git Flow (sadly)'* and *'mob programming,'* which had one mention each. Actually, documented development method *'RDD (Resume-Driven Development)'* also had a mention, but with values such

[1]Respondents could choose multiple items when answering questions analysed for this and other tables in this study, so percentages may sum to more than 100.

TABLE V
SECURE CODING METHODS, STANDARDS AND IMPORTANCE. N VARIES BY QUESTION AND IS IN THE HEADINGS BELOW.

Language, tool or aid	Frequency	%
Development Methods (n=757)		
Agile	584	77.1
DevOps	388	51.3
Scrum	327	43.2
Kanban	262	34.6
TDD (Test-Driven Development)	187	24.7
Waterfall	112	14.8
FDD (Feature-Driven Development)	103	13.6
DevSecOps	67	8.9
Prototype	55	7.3
I don't know	46	6.1
XP	29	3.8
Lean	26	3.4
RAD	7	0.9
Joint application development	6	0.8
RUP	0	0.0
Secure Coding Standards (n=594)		
None	272	45.8
Not Applicable	141	23.7
PCI-DSS	68	11.4
FIPS 140-2	36	6.1
Common Criteria (ISO/IEC 15408)	30	5.1
US-CERT's Top 10 Secure Coding Practices	28	4.7
Microsoft SDL	23	3.9
NIST 800-64 / SP 800-160	21	3.5
I don't know	18	3.0
MISRA	7	1.2
DIACAP	2	0.3
SAFECode	2	0.3
BSIMM	1	0.2
Regulation (EU) 2017/745	1	0.2
Most Important Aspects of Security (n=863)		
Data protection	714	82.7
Preventing vulnerabilities	693	80.3
Customer privacy	639	74.0
Customer confidentiality	607	70.3
Preventing reputational damage to organisation	345	40.0
Passing external penetration tests	274	31.7
Passing external compliance checks	226	26.2
Having a good personal security reputation	217	25.1
I do not think that software security is important	2	0.2

as *'Hiring buzzwords over proven track records'*[2] its inclusion seemed somewhat tongue-in-cheek, as did that of *'JustTryIt.'* Other participants indicated that their working environments used non-standard methods; for example, *'Our own thing, it doesn't have a fancy name.'* Analysis of the 22 free text entries showed that while 32% of free text contributors seemed happy with the development methods used ('*we're all full stack, manage our own aws stuff, aim for dynamic incremental planning, sometimes pair program, trunk based development*'), 50% did not. This was made abundantly clear in their responses, for example: *'It's a dog's breakfast. Everything from poorly-done scrum run by people who don't understand how to run scrum, to waterfall followed by agile-ish stuff to fix all the parts that didn't get planned,'* and *'Lip service paid to Agile.'*

[2]https://rdd.io/

Asked about secure coding standards at work, 594 participants responded (see Table V). There were 67 free text entries added. Some were predictable options inadvertently omitted from the survey list, such as '*MISRA*' (Motor Industry Software Reliability Association) (7) and '*I don't know*' (18), but there were also thoughtful responses such as '*None of these look familiar to me, but we do try to use the Rust programming language more often (and for new projects) nowadays to increase security of our products*.' Based on the secure-development literature, SAMM, CLASP and Synopsys Touchpoints were included in the survey options [17, 52, 53]. They are now considered old-fashioned [6] and interestingly, they had no users in this sample.

Asked what aspects of software security they feel are most important, all 863 respondents replied. See bottom of Table V for the ordering of the aspects supplied in the survey. There were also 49 free text entries giving a varied range of concerns with little in common, such as '*Customer safety from physical harm*,' '*Forward security: protecting against the future (e.g. post quantum cryptography)*,' '*Lateral Movement / Priv Escalation*,' and '*I don't work at a position that's concerned with security, but I personally think it's important*.' The wide range of concerns reflects the large and varied sample.

V. RESULTS

A. Prior work: Security Practice and Security Culture

As part of the developer survey described in Section III, we devised a simple empirically-based measurement instrument for secure coding practice in organisations. The CA Score, documented in prior work [43], measures an organisation's use of the twelve secure-coding practices used most often in organisations [54]. The range of CA Score values amongst developers responding to the survey was normally distributed, with some developer environments getting a score of zero, some twelve, and most a score in between the two. Fig. 1 shows the range of CA Scores in the sample used for this paper.

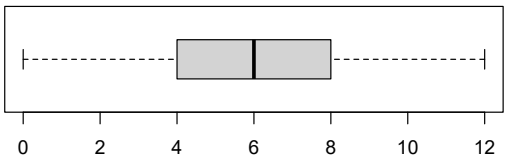

Fig. 1. Security CA Score for respondents' organisations. The score reflects the number of secure-coding common activities undertaken by the organisation.

Research suggests that introducing security **practice** to an organisation without an attendant good security **culture** can create a '*checkbox*,' compliance-focused security environment, which can have adverse effects on secure coding in organisations. A respondent described some of the issues that arise with practice when culture is deficient: '*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.' Good security culture was described by Haney et al. as *'a subculture of an organization in which security becomes a natural aspect in the daily activities of every employee'* [19]. Our survey asked respondents questions about the security culture of their working environment. Responses to twelve of these culture questions (which are listed in Table VI) were evaluated in the same prior work that assessed practice [43]. We found that some work environments with high CA scores appeared to have a poor security culture. We explore security culture further in section V-C3.

TABLE VI
SECURITY CULTURE QUESTIONS. THESE QUESTIONS ARE DESIGNED TO ASSESS THE SECURITY CULTURE IN AN ORGANISATION.

ID	Question
Q1	How much support do you feel that you get from your employer or open source team to code securely?
Q2	If you had security concerns about your current project, how likely would you be to raise them with someone?
Q3	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?
Q4	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?
Q5	If you needed permission to use the tool, how likely would you be to ask for permission?
Q6	If you asked, how likely do you think it is that you would get permission?
Q7	If you asked, how likely do you think it is that you would get funding for a **paid** tool?
Q8	Have you ever heard people say that there is a software security culture in your working environment?
Q9	Would you agree that there is a security culture in your working environment?
Q10	How highly do you think your team prioritises software security?
Q11	How often is software security mentioned in team communications?
Q12	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.

B. Research Question 1: Do developers working for organisations have a basic level of software security knowledge?

As discussed in Section II, it is difficult to assess developers' secure coding expertise at scale. We asked two multiple-choice questions designed to tell us whether respondents had a basic level of secure coding knowledge. Both questions were answered correctly by a large majority of the respondents. Asked which of a number of languages was most likely to cause buffer overflow, all 863 participants answered the question, and 806 (93.4%) correctly chose *'C++.'* The second security question asked participants to choose a possible security issue from a list; 861 participants answered, with 802 (92.9%) correctly choosing *'SQL injection.'*

We devised the first question because buffer overflow is a very common vulnerability with severe security implications. U.S. non-profit Mitre's influential '*CWE Top 25 Most Dangerous Software Weaknesses*' currently includes three items associated with buffer overflow [55]. According to Mitre at time of writing, one of these, '*2: Out-of-bounds Write*', has 18 corresponding entries in the Known Exploited Vulnerabilities (KEV) database maintained by the U.S. government's CISA agency [56]. This is 11 more than any other member of the Top 25.

We devised the second question because SQL injection is also very common and widely exploited. '*Injection*' was at the top of the '*OWASP Top Ten Web Application Security Risks*' list until 2021, and is now at number three [57]. The associated weakness (*'Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')'* is currently at number three on the Mitre list, with four corresponding entries in CISA's KEV. Introductory software security education for developers would typically reference buffer overflow and SQL injection [58].

We asked a third question on secure-coding knowledge. The Open Web Application Security Project (OWASP) is an online secure-coding community which maintains the OWASP Top Ten and numerous secure-coding tools, projects, and cheat-sheets. It is web focused, but a great deal of software-security issues are web related. We asked respondents whether they had ever heard of OWASP (see Fig. 2). Although all respondents answered this question, the chart excludes the 12 free text answers, giving a total of 851.

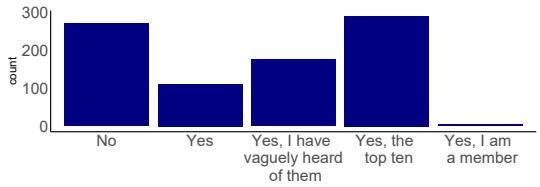

Fig. 2. Before you read this question, had you ever heard of OWASP? (n=851)

As part of our attempts to maximise the number and honesty of survey responses, we tried to make survey questions engaging and not too repetitive. In the case of the OWASP question the non-standard suggested responses make analysis difficult. However, if broken down into a simple yes/no binary, the results are encouraging. Of those choosing a standard response, 271 people (31.4%) chose '*No*.' Of the 12 free text responses, three were interpreted as '*No*,' seven began with '*Yes*,' another was positive, and the last was '*I wrote the first OWASP guide*,' which we judged to be a '*Yes*.' This gives a total of 274 (31.7%) negative responses, and 589 (68.3%) positive responses. In other words, approximately two-thirds of our respondents had heard of OWASP. Overall, the results of these three questions indicated that most respondents in our sample had a basic level of secure-coding knowledge.

C. Research Question 2: How do developers working for organisations currently feel about secure coding, and about security practice, culture and support in the organisation?

1) Developer Sentiment: In prior work we reported an analysis of survey responses for lone and isolated developers, and those working outside organisations [45]. In this work we focus on developers working within organisations. In recent years, widespread shifts to DevOps and DevSecOps have removed some of the traditional security gates, and put increased pressure on developers to be security champions in their working environments [59]. Yet it is unclear how developers experience secure coding. We asked if they are interested in software security? Do they feel competent to code securely? Do they enjoy trying to code securely? We chose these questions on developers' secure-coding interest, knowledge and enjoyment based on a systematic literature review [60], which informed our understanding of the requirements and constraints involved in secure coding for individuals. For clarity, we will refer to developer attitudes as expressed in answers to these three questions as '*developer secure-coding sentiment*' throughout this paper.

Fig. 3 shows the level of interest developers have in coding securely, and Fig. 4 shows the level of knowledge developers feel they have about secure coding. While 65% of developers have '*A lot*' or '*A great deal*' of interest in secure coding, only 22% of developers would say that they have the corresponding level of knowledge. Unsurprisingly, answers to these two questions correlate, with a medium correlation of .46 ($p<0.001$).[3] We can visualise the correlation in the stacked Likert chart in Fig. 5. Of developers who have only a little interest in secure development, 66% have little or no secure development knowledge; while only 14% of those with a great deal of interest have similarly low knowledge levels.

The three questions on secure-coding knowledge assessed in section V-B reflected a reasonable degree of basic understanding by respondents. However, secure coding is a complex issue and respondent sentiment on knowledge shows that developers grasp how much understanding is needed. Training could provide it; unfortunately, in prior work we discovered that only 40% of respondents to this survey had been offered secure coding training in their work environment [44]. Of those who had been offered training, 23% indicated that it was subpar.

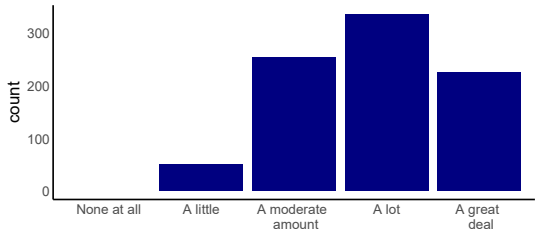

Fig. 3. What level of interest do you have in developing secure software? (n=862)

[3]We use correlation effect-size definitions from Cohen [61].

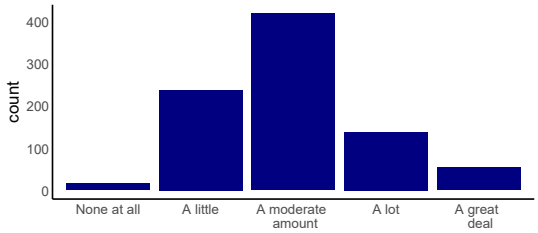

Fig. 4. What level of software development security knowledge would you say you have? (n=863)

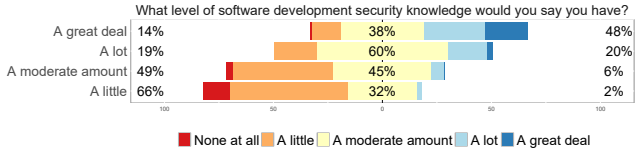

Fig. 5. Individuals' knowledge about secure software development, grouped by interest. The less interest a developer has in secure development, the less they know about it. In the survey, no member of the sample selected '*None at all*' for level of interest. (n=862)

We next asked respondents whether they enjoy coding securely. We used a unique scale for this question to encourage engagement and honest responses (see Fig. 6). Only five valid respondents chose *'I hate it.'* Only 10% of respondents claimed to hate secure coding or stated that they did not like it much, while 54% stated that they enjoy it or that they love it.

2) Exploring relationships between developer sentiment and practice and organisational characteristics: The developer-centred security literature has reported varying results on the effect of organisation size on secure coding. Some studies have found that organisation size does not affect secure coding [7, 15, 19], while others have observed that it does [5, 62]. To contribute to this open question, we checked whether answers on developer secure-coding sentiment and the CA Score correlate with organisation size, team size or number of developers in an organisation (see Table VII).

Prioritising security and spending time on it can have a positive effect on security culture in a working environment [19, 23, 63]. This may be more relevant than statistics such as company size to assessing developer secure-coding sentiment. We expected that responses on developer secure-coding sentiment would be more strongly correlated to measurements of security practice and culture than to organisational metrics such as size, team size, or number of developers.

Analysis showed that while security knowledge had a small correlation to team size, developer secure-coding sentiment had no other significant correlation with organisational metrics. Interest had a small and knowledge a medium correlation with the CA Score. The CA Score also correlated weakly with organisation and team size, and moderately with number of developers in an organisation, suggesting that software-security practices are more common in organisations with large developer groups.

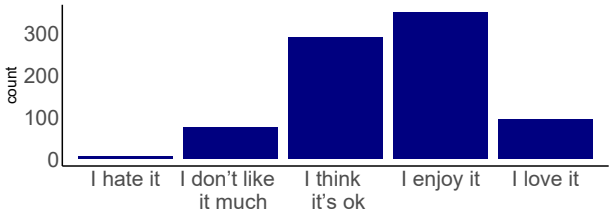

Fig. 6. How much do you enjoy the challenge of attempting to code securely? Only 10% hated it or did not like it much. (n=814)

TABLE VII
CORRELATIONS BETWEEN SOFTWARE SECURITY INTEREST, KNOWLEDGE AND ENJOYMENT, AND THE CA SCORE IN THE SAMPLE. WE EXPLORED POSSIBLE CORRELATIONS WITH ORGANISATION SIZE, NUMBER OF DEVELOPERS AND TEAM SIZE. ONLY RELATIONSHIPS WITH P-VALUES <0.001 ARE CONSIDERED STATISTICALLY SIGNIFICANT. THESE ARE IN BOLD. N VARIES BY QUESTION BUT IS 863 OR LESS.

	Interest	Knowledge	Enjoyment	CA Score
Interest	1.00			
Knowledge	**0.46**	1.00		
Enjoyment	**0.44**	**0.25**	1.00	
CA Score	**0.20**	**0.31**	0.11	1.00
Org Size	0.03	0.08	–0.03	**0.28**
Num Developers	0.03	0.09	0.01	**0.32**
Team Size	0.02	**0.12**	0.01	**0.29**

3) Developers and security culture: As discussed, in prior work we investigated the answers to twelve questions on software security culture in the organisation (see Table VI). These twelve security **culture** questions, derived from analysis of the literature, differ from the twelve empirically-based CA Score questions, which measure **practice**. The detailed rationale for each of these culture questions has been omitted here for space reasons, but can be found, with further discussion, in prior work [43].

In order to assess the benefits of security culture we wished to explore these questions further. We suspect that the correlations between developer secure-coding sentiment and organisational security culture will be greater than those between developer secure-coding sentiment and the CA Score. This would indicate that developers tend to have better developer secure-coding sentiment in environments where there is a good secure-coding culture. This would provide additional incentive for organisations to improve their software security culture.

It is difficult to evaluate correlations on twelve questions separately, and furthermore there is likely to be redundancy among these initial questions which can be reduced using factor analysis. We therefore conducted an exploratory factor analysis on answers to these twelve questions. We found that only 539 participants had answered all 12 security culture questions. The KMO value for this sample is 0.83, making it '*meritorious*' and suitable for factor analysis. A scree plot and eigenvalues suggested that two factors would best reflect the data, and on undertaking exploratory factor analysis we found that two factors provided the most explanatory value. Fig. 7 shows a visualisation of the two factors. We interpret the first factor (with seven items) as comprising items which indicate support for secure coding in the developer's working environment. These include items such as software security prioritisation by the team, and time available to work on software security during the week. We named this factor '*Environment*.' The second factor includes items such as whether the developer would raise software security concerns if they had them, and whether the developer believes that they would get permission from management to introduce new software-security tools if they asked for it. We interpret these items as relating to the developer's feeling of secure-coding empowerment, and therefore named this factor '*Empowerment*.'

TABLE VIII
CORRELATIONS BETWEEN THE TWO SECURITY CULTURE FACTORS AND OTHER VALUES IN THE SAMPLE. STATISTICALLY SIGNIFICANT RELATIONSHIPS ARE IN BOLD. (N=539).

	Interest	Knowledge	Enjoyment	CA Score
Environment	**0.28**	**0.36**	**0.18**	**0.63**
Empowerment	0.12	0.13	0.12	**0.25**

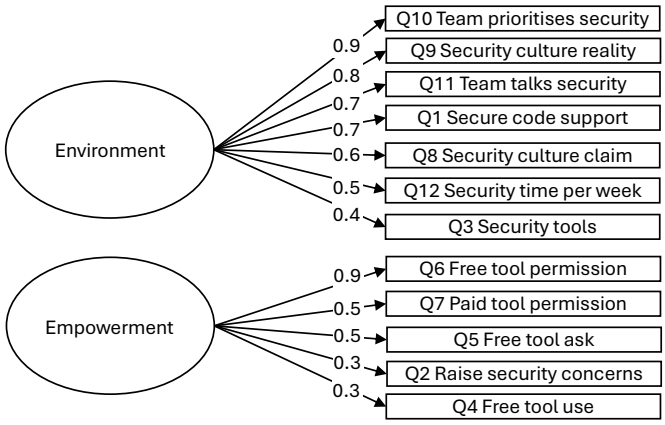

Fig. 7. Diagram of exploratory factor analysis for the 12 security-culture questions in the survey (see Table VI). (n=539, KMO=0.83, principal axis factoring with promax rotation)

In order to assess the relationship of developer secure-coding sentiment to software security culture, we ran correlation analysis between the three sentiment values and the two factors. The results can be seen in Table VIII. We were interested to see that the first factor, environment, correlated weakly to moderately with developer secure-coding sentiment: interest, knowledge and enjoyment. This factor correlated strongly with the CA Score. This is good news as it indicates a strong connection between good software security practice and good software security culture. The second factor, empowerment, had a weak correlation with the CA Score but no statistically significant correlation with developer secure-coding sentiment.

4) Developers and software security concerns: We asked developers whether, if they had code security concerns, they would raise them with someone (see Fig. 8). A very large

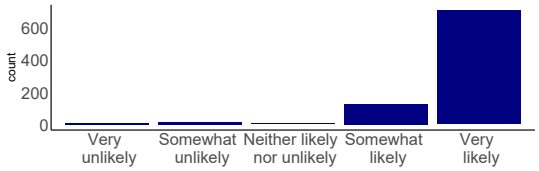

Fig. 8. If you had security concerns about your current project, how likely would you be to raise them with someone? (n=863).

majority of 96% would be somewhat or very likely to raise the concerns with someone.

The follow-up question asked who they would raise the concerns with. We were interested to see whether developers would be more likely to approach someone within their coding environment, or to look for outside assistance; for example, consulting a friend or industry expert. We offered five options and an '*other*' option with free text and received 853 answers. There were 46 free text answers, with the longest being 13 words. These were coded into appropriate groups, resulting in eight groups in total (see Table IX). Of the three people entering '*myself*,' one added that they were the CTO, and another that they were the R&D software manager.

The person who respondents would be most likely to tell about a security concern is a manager, at 78.7%. This is followed by one or more members of the team at 58.9%. With internal security groups coming third at 45.1%, the numbers suggest that most respondents would be likely to raise security concerns with multiple people within their work environment. This is a practical illustration that most respondents feel a level of support for secure-coding issues in their coding environment.

TABLE IX
IF YOU DID RAISE SECURITY CONCERNS WITH SOMEONE, WHO WOULD IT BE? (N=853)

Description	Number	%
Manager	671	78.7
Team	502	58.9
Internal security group	385	45.1
Friend	69	8.1
External Expert	26	3.0
Colleagues	8	0.9
Myself	3	0.4
Those Affected	1	0.1

D. Research Question 3: How does a security breach affect the secure coding focus of individuals and teams in the short and long term?

Assal and Chiasson [15] and Lopez et al. [64] found that experiencing a software security breach can cause individuals and teams to become more security aware, while Tomas et al. [65] encountered a developer whose organisation essentially ignored a security compromise. We were intrigued by these varied findings, because intuitively the discovery of a vulnerability causing a breach should enhance software security practice. In order to investigate further and contribute to the literature, we asked four questions on this topic in our survey. We asked participants whether they *themselves* had experienced a breach in code they had written, and if so, whether it had affected their attitude to security in the short or long term. We also asked whether *their organisation or team* had experienced a breach, and if so, whether it had affected its attitude to security in the short or long term. Table X presents the answers to the questions on whether they had experienced a breach. Fig. 9 shows the responses when participants were asked whether the security breach in their own code had affected their personal secure coding. Fig. 10 shows the responses when asked whether the security breach in their work environment had improved secure coding there. For almost half of cases in both demographics, the security breach caused either a short-term effect or no effect at all. There were eleven free text responses to the question on the response to a personal breach. They spanned the range of possible security reactions, from *'Our general level is pretty good, but obviously remedial work took place immediately'* to *'Wasn't allowed to take the time to prioritise.'* A gap between engineering and management appeared in one case: *'This gets management's attention, but engineers already know what's important, and are usually deprioritised.'*

TABLE X
BREACH HISTORY FOR RESPONDENTS' OWN CODE AND THEIR WORK ENVIRONMENTS. N VARIES BY QUESTION AND CAN BE FOUND BELOW.

Context	n	Had Breach		No Breach		Don't Know	
		n	%	n	%	n	%
Respondent's Code	863	260	30.1%	603	69.9%	0	0
Work Environment	848	259	30.5%	358	42.2%	231	27.2%

There were five free text responses to the question on the organisation or team's response to a breach. Two respondents did not know whether security had improved, one noted that security was already highly prioritised, one said there had been an improvement, and the final respondent said *'ImmediateShortTerm3MthsAfterThatLOL.'*

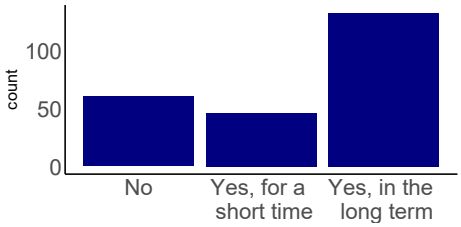

Fig. 9. If you answered yes to the previous question, did the breach affect how much you prioritise security? (n=238)

VI. DISCUSSION

There is cause for optimism in some of the results we have found. For example, it is reassuring to know that most developers will turn first to their managers and colleagues with security concerns. Respondents showed a good grasp of secure coding basics, with 542 (63%) of the sample answering the

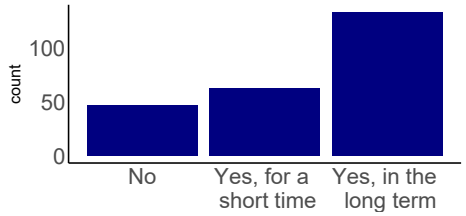

Fig. 10. If you answered yes to the previous question, did the breach affect how much the organisation or team prioritises security? (n=242)

two secure-coding questions correctly and indicating familiarity with OWASP. They expressed an interest in coding securely. These results reflect findings made by others, for example Assal and Chiasson found that developers on the whole were self-motivated to consider software security [5]. Other results, however, do not allow for complacency. The finding that approximately 45% of software breaches cause only short term changes, or no changes, in the security stance of relevant individuals and organisations is worrying. The degree of cynicism displayed in free text entries, while often amusing, is also of concern to those who see software security as one of the major issues currently facing the developer community, and who wish to see developers supported. Here is an example: *'in general, working in Big Tech, security is deprioritized, underfunded, and usually seen as an obstacle to shipping that stands in the way of promotion incentives up and down the line of almost all disciplines (prod, eng, etc.)- this is not universally true (some companies better than others, and some teams w/i companies better than other teams in same company) but is generally true on "product" teams and companies.'*

Many of the improvements required to enhance software security have been discussed for a long time; improved tools [66], improved documentation [67], better web resources [68] and more. However, the role of the organisation or other work environment in enhancing security is vital, since they control budget and resources [15, 23, 24]. To take an example from one respondent: *'Generally I think my organization takes security fairly seriously, partially from a legal "CYA" perspective. The organization could do better. In recent years, we have almost no QA, with developers doing their own test coding. I think the lack of formal QA (much less security-aware QA) is an ongoing issue.'*

These research findings have implications for software security in practice. The large correlation we have observed between good software security practice and culture suggests a real relationship between them. Software security research provides pointers for how organisations can enhance software-security culture to avoid issues and shortcomings in software-security practice. Haney et al. found that security-focused organisations make extensive use of mentoring and deep expertise, and incorporate security activities into the daily life of the organisation [19]. Having software security experts available to foster a culture of discussion and action was a key component of the introduction of a security culture in an organisation studied by Tuladhar et al. [63]. Studies that observe a positive security culture among developers tend to observe that support from upper management is key. Senior managers who wish to enhance software security in their organisations should provide overt and explicit support.

A. Limitations

As in all survey studies, participants may have been motivated to take the survey by an interest in the topic. Survey responses may also be affected by social desirability bias. To mitigate these dangers, we emphasised in recruitment materials that respondents did not need to know anything about software security, and that the survey was anonymous.

Answers for the security culture questions subject to factor analysis do not all use the same scales. However, Norman has established that the Pearson correlation is '*extremely robust with respect to violations of assumptions*,' and argues that it can be safely used with non-normal, skewed and ordinal data [69]. Correlations found outside factor analysis use Spearman's rho because most of the data is ordinal. In some cases we also ran a Pearson's correlation and obtained nearly identical results. In this paper correlation findings are used to uncover interesting research directions, not to indicate causation.

VII. Conclusion

Our comprehensive survey results provide insight into the current secure coding stance of developers. We found that developers are generally interested in security and enjoy the challenge of coding securely. Basic secure-coding knowledge was displayed by 63% of respondents, but they are not always confident in their secure coding knowledge, with 29% stating that they only have a little knowledge or none at all, and only 22% claiming to have a lot or a great deal of knowledge. We also found that developer interest in and knowledge of security tend to correlate with good security practice and culture in the work environment. Software security practice in the work environment correlates strongly with indicators of software security culture, showing that good software security practice and culture go hand-in-hand.

If concerned about a security issue, 96% of our respondents said that they would be somewhat or very likely to raise the issue with someone. Of these, 78.7% would raise it with a manager, and many would also bring it to the attention of others within their work environment. This shows that most developers have confidence that security issues can be dealt with in-house.

We found that experiencing a software breach changed how much over 50% of developers and organisations prioritised security in the long term, although a sizeable minority were inattentive to security.

Acknowledgment

This publication was financially supported by Science Foundation Ireland under Grant numbers 13/RC/2094-P2 and 18/CRT/6222. For the purpose of Open Access, the authors have applied a CC-BY public copyright license to any Author Accepted Manuscript version arising from this submission.

References

[1] I. Ryan, U. Roedig, and K.-J. Stol, "Insecure software on a fragmenting Internet," in *Proceedings of the 2022 Cyber Research Conference - Ireland (Cyber-RCI)*, 2022.

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

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

[4] 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*. ACM Press, 2017, p. 46–56.

[5] 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*. ACM, 2019, p. 1–13.

[6] 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.

[7] D. van der Linden, P. Anthonysamy, B. Nuseibeh, T. T. Tun, M. Petre, M. Levine, J. Towse, and A. Rashid, "Schrödinger's security: Opening the box on app developers' security rationale," in *Proceedings of the 42nd International Conference on Software Engineering*, 2020, p. 12.

[8] A. A. Ur Rahman and L. Williams, "Security practices in DevOps," in *Proceedings of the Symposium and Bootcamp on the Science of Security*. ACM, 2016, p. 109–111.

[9] E. Derr, S. Bugiel, S. Fahl, Y. Acar, and M. Backes, "Keep me updated: An empirical study of third-party library updatability on Android," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM Press, 2017, p. 2187–2200.

[10] W. Bai, O. Akgul, and M. L. Mazurek, "A qualitative investigation of insecure code propagation from online forums," in *2019 IEEE Cybersecurity Development (SecDev)*. IEEE, 2019, p. 34–48.

[11] T. Lopez, T. Tun, A. Bandara, M. Levine, B. Nuseibeh, and H. Sharp, "An anatomy of security conversations in Stack Overflow," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Society*. IEEE Press, 2019, p. 31–40.

[12] 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*. IEEE Press, 2019, p. 61–68.

[13] M. Green and M. Smith, "Developers are not the enemy! The need for usable security APIs," *IEEE Security and Privacy*, vol. 14, no. 5, p. 40–46, 2016.

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

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

[16] 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 *IEEE Cybersecurity Development (SecDev)*. IEEE, 2017, p. 22–26.

[17] 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 *Proceedings of the 11th International Conference on Availability, Reliability and Security (ARES)*, 2016, p. 548–555.

[18] J. Hallett, N. Patnaik, B. Shreeve, and A. Rashid, ""Do this! Do that!, And nothing will happen" Do specifications lead to securely stored passwords?" in *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering*. IEEE, 2021, p. 486–498.

[19] 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 14th Symposium on Usable Privacy and Security*. USENIX Association, 2018, p. 357–373.

[20] 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 *Proceedings of the 16th Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, 2020, pp. 205–220.

[21] Y. Acar, S. Fahl, and M. L. Mazurek, "You are not your developer, either: A research agenda for usable security and privacy research beyond end users," in *Proceedings of the IEEE Cybersecurity Development (SecDev)*. IEEE, 2016, p. 3–8.

[22] 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*, vol. 31, no. 1, pp. 9:1–9:52, 2021.

[23] 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.

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

[25] 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*. ACM, 2020, p. 8.

[26] T. Bender, R. Huesmann, and A. Heinemann, "Software development processes for ADs, SMCs and OSCs supporting usability, security, and privacy goals - an overview," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*. ACM, 2021, p. 1–6.

[27] 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 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, p. 311–328.

[28] 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*. ACM, 2019, p. 1–12.

[29] 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*. ACM, 2020, p. 1–13.

[30] 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, 2018.

[31] A. Ruef, M. Hicks, J. Parker, D. Levin, M. L. Mazurek, and P. Mardziel, "Build It, Break It, Fix It: Contesting secure development," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, p. 690–703.

[32] 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*. ACM, 2020, p. 1–20.

[33] C. Sadowski, E. Aftandilian, A. Eagle, L. Miller-Cushon, and C. Jaspan, "Lessons from building static analysis tools at Google," *Communications of the ACM*, vol. 61, no. 4, p. 58–66, 2018.

[34] V. Mohan and L. B. Othmane, "SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps," in *Proceedings of the 11th International Conference on Availability, Reliability and Security (ARES)*, 2016, p. 542–547.

[35] S. Xiao, J. Witschey, and E. Murphy-Hill, "Social influences on secure development tool adoption: Why security tools spread," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, 2014, p. 1095–1106.

[36] C. Wijayarathna and N. A. G. Arachchilage, "Why Johnny can't develop a secure application? A usability analysis of Java Secure Socket Extension API," *Computers and Security*, vol. 80, p. 54–73, 2019.

[37] T. Espinha Gasiba, I. Andrei-Cristian, U. Lechner, and M. Pinto-Albuquerque, "Raising security awareness of cloud deployments using infrastructure as code through cybersecurity challenges," in *The 16th International Conference on Availability, Reliability and Security*. ACM, 2021, p. 1–8.

[38] P. L. Gorski, L. L. Iacono, D. Wermke, C. Stransky, S. Moeller, Y. Acar, and S. Fahl, "Developers deserve security warnings, too: On the effect of integrated security advice on cryptographic API misuse," in *Proceedings of the 14th Symposium on Usable Privacy and Security (SOUPS 2018)*, 2018, p. 265–281.

[39] 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*. ACM, 2020, p. 136–148.

[40] L. Lo Iacono and P. L. Gorski, "I do and I understand. Not yet true for security APIs. So sad," in *Proceedings 2nd European Workshop on Usable Security*. Internet Society, 2017, p. 11.

[41] D. Votipka, K. R. Fulton, J. Parker, M. Hou, M. L. Mazurek, and M. Hicks, "Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It," in *Proceedings of the 29th USENIX Security Symposium*. USENIX Association, 2020, p. 109–126.

[42] S. Nadi, S. Krüger, M. Mezini, and E. Bodden, "Jumping through hoops: Why do Java developers struggle with cryptography APIs?" in *Proceedings of the 38th International Conference on Software Engineering*. ACM Press, 2016, p. 935–946.

[43] I. Ryan, U. Roedig, and K.-J. Stol, "Measuring secure coding practice and culture: A finger pointing at the moon is not the moon," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023.

[44] ——, "Training developers to code securely: Theory and practice," in *2024 IEEE/ACM 5th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*, 2024.

[45] I. Ryan, K.-J. Stol, and U. Roedig, "The state of secure coding practice: Small organisations and 'lone, rogue coders'," in *2023 IEEE/ACM 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*, 2023, p. 37–44.

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

[47] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, 2024. [Online]. Available: https://www.R-project.org/

[48] J. Bryer and K. Speerschneider, *likert: Analysis and Visualization Likert Items*, 2016, r package version 1.3.5, http://github.com/jbryer/likert. [Online]. Available: http://jason.bryer.org/likert

[49] William Revelle, *psych: Procedures for Psychological, Psychometric, and Personality Research*, Northwestern University, Evanston, Illinois, 2024, r package version 2.4.6. [Online]. Available: https://CRAN.R-project.org/package=psych

[50] M. Kuhn, S. Jackson, and J. Cimentada, *corrr: Correlations in R*, 2022, r package version 0.4.4, https://corrr.tidymodels.org. [Online]. Available: https://github.com/tidymodels/corrr

[51] I. Ryan, U. Roedig, and K.-J. Stol, "Data and scripts for "'immediateshortterm3mthsafterthatlol": Developer secure-coding sentiment, practice and culture in organisations'," 2025. [Online]. Available: https://osf.io/tfxp9/?view_only=5fb840be1d6b43d19c54ebade77419de

[52] P. Morrison, B. H. Smith, and L. Williams, "Surveying security practice adherence in software development," in *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp*. ACM, 2017, p. 85–94.

[53] 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*. ACM, 2018.

[54] 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.

[55] "2024 CWE Top 25 most dangerous software weaknesses," https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html.

[56] U.S. Cybersecurity and Infrastructure Security Agency (CISA), "Known exploited vulnerabilities catalog," https://www.cisa.gov/known-exploited-vulnerabilities-catalog.

[57] Open Worldwide Application Security Project (OWASP), "OWASP Top Ten," https://owasp.org/www-project-top-ten/.

[58] "Becoming a secure coder," https://www.codecademy.com/learn/becoming-a-secure-coder.

[59] H. Myrbakken and R. Colomo-Palacios, *DevSecOps: A Multivocal Literature Review*. Springer International Publishing, 2017, vol. 770, p. 17–29.

[60] I. Ryan, U. Roedig, and K.-J. Stol, "Unhelpful assumptions in software security research," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2023.

[61] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Elsevier, 1977.

[62] 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*. ACM, 2020, p. 1513–1531.

[63] 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 *Proceedings of the Seventeenth Symposium on Usable Privacy and Security*, 2021, p. 617–631.

[64] T. Lopez, H. Sharp, T. Tun, A. Bandara, M. Levine, and B. Nuseibeh, "Talking about security with professional developers," in *Proceedings of the Joint 7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice*. IEEE Press, 2019, p. 34–40.

[65] 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)*, 2019, p. 1–8.

[66] G. Wurster and P. C. van Oorschot, "The developer is the enemy," in *Proceedings of the 2008 New Security Paradigms Workshop*. ACM, 2008, p. 89–97.

[67] 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.

[68] T. Lopez, T. T. Tun, A. Bandara, M. Levine, B. Nuseibeh, and H. Sharp, "An investigation of security conversations in Stack Overflow: Perceptions of security and community involvement," in *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment*. ACM, 2018, p. 26–32.

[69] 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.