Training Developers to Code Securely: Theory and Practice

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

Utz Roedig
Connect, the SFI Research Centre for Future Networks and Communications
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 is essential. Flaws in software design and coding produce vulnerabilities that can be exploited by hostile actors, resulting in ransomware, espionage and the hacking of critical infrastructure. Meanwhile, DevOps and continuous integration introduce speed imperatives, often bypassing traditional security gates. Software security responsibility is increasingly shifting to software developers. Industry insiders advise that this extra responsibility should be accompanied by developer training. But is it? Analysis of what constitutes good software security training is sparse, and there is little information on the amount and quality of training actually offered to developers in industry. We analyse recent literature to find the positive features of effective secure development training. We examine training information from a large developer survey (n=962) to assess how training in the field matches key positive features. We find that while some developers experience excellent secure-coding training, others receive inadequate training, and the majority receive none.

CCS CONCEPTS

• **Security and privacy → Software and application security**; **Human and societal aspects of security and privacy**.

KEYWORDS

secure software development, security, secure coding training

ACM Reference Format:
Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2024. Training Developers to Code Securely: Theory and Practice. In *2024 ACM/IEEE 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS) and 2024 IEEE/ACM Second International Workshop on Software Vulnerability (EnCyCriS/SVM '24), April 15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3643662.3643956

1 INTRODUCTION

Software security as a concern becomes more pressing as the world becomes more connected [41]. At the heart of software development, and therefore of software security, is the developer, who is most directly responsible for vulnerabilities in software design, and mistakes in implementation. Security-conscious workplaces may have software security teams, but their role as gatekeepers has declined with the rise of DevSecOps [26, 48]. Security auditors are often outnumbered by factors of more than a hundred to one [48]. Yet we have little understanding of the secure-coding training that developers receive. What features are important for secure coding training, and are they present in contemporary training courses? Are there security-training barriers that may reduce effectiveness, such as time pressure or lack of focus? What type of training is effective, and is it currently being offered in the real world?

The importance of developer secure-coding training is emphasised in the literature [5, 6, 47–49, 51, 54]. Tomas et al. discussed developer security training as a metric that can usefully be monitored as part of DevSecOps [49]. By correlating with the number of mistakes in different security categories, this metric could be used to guide training programs. For this measurement to be effective it should distinguish between vastly different training. An annual video does not compare, for example, with an immersive one-week instructor-led training course. More exploration of the nature of secure-coding training is needed [27].

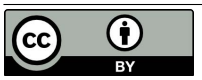
Secure-coding training is used and recommended in industry. For example, the Building Security In Maturity Model, an industry secure-coding model, suggests that organisations provide software security awareness training, including individualised training and training during onboarding [9]. However, availability of training materials does not guarantee that they will be accessed. Potential barriers to security training, such as time pressure and lack of relevance, should be investigated and removed. Management buy-in is essential to having training time allocated for developers.

As part of a systematic literature review [40], we explored the features of secure-coding training. In this paper we analyse successful training and identify the features that appear to affect training quality. As part of a large developer survey [42], we enquired about the secure-coding training the participants received. We analysed the answers and compared them to the features we had identified. Our research questions are as follows:

RQ1: What are the features of effective secure-coding training?

RQ2: Are developers offered secure coding training, and if so, does it have the positive features we have identified?

Our contribution is twofold: we identify the features necessary for successful secure-coding training according to academic literature, and we describe the current state of secure coding practice, including whether those features are present.



EnCyCriS/SVM '24, April 15, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0565-6/24/04.
https://doi.org/10.1145/3643662.3643956

The paper is organised as follows: Section 2 discusses background and related work. Section 3 outlines the methodology. Section 4 uncovers the features of successful secure-coding training. Section 5 analyses the survey data on current practice, to see if key features are present. In Section 6, we discuss our findings, their implications and limitations. Section 7 concludes the paper.

2 BACKGROUND AND RELATED WORK

2.1 Training Availability

Developers may be motivated to code securely but ignorant of how to do it. Researchers have long called for more software security training in computer science curricula [5, 10, 14, 32, 48]. Even the most reputable universities still conduct a surprisingly low amount of secure-coding training [4]. In fact, lecturers routinely use insecure functions in code examples; thus they are actively training future developers to code insecurely [4]. Many teachers would like to adjust training materials to reflect code security concerns, but lack the time to update their content [11].

With little to no secure coding education before employment, developers are dependent on the organisations they work for to provide this training. Arizon-Peretz et al., in interviews with 27 practitioners, found that '*many of the participants' statements indicated that they did not feel that they had received sufficient training with regard to security and privacy*' [6].

There is little literature quantitatively evaluating whether developers are generally offered software security training in the workplace, and whether they are satisfied with the training offered. Pikulin et al. ran a small survey, concluding that developers' training preferences vary with experience and personality [36].

2.2 Training Effect

We did not find any previous work that focused primarily on correlation or causation between secure coding training and actual code security. We did identify studies which considered secure training effectiveness as a secondary concern. For example, Witschey et al., studying security tool adoption, found a correlation between secure coding training and adoption of security tools [57].

In studies that considered training and security correlation as a secondary concern, researchers disagreed and sometimes contradicted themselves. A laboratory study by Oliveira et al. examining '*blind spots in developer's heuristic-based decision-making processes*' stated that security training had a significant effect on security outcomes [30]. Developers were asked to modify and comment on vulnerable code snippets. The researchers checked for correlations between total scores and occupation, level of education, and so on. They stated that developers' total score and whether or not they had previously taken security classes were '*highly correlated.*' However, the Pearson's correlation value given in the paper's Table 5 is −0.313. This suggests a weak negative correlation, with the developers who had taken security classes actually producing slightly *less* secure code than others. It is unclear whether it is the positive assertion in the text, or the negative value in the table, that is correct, or whether the apparent discrepancy is due to a misunderstanding. Also, the content, relevance and duration of the security classes are not described. It appears that the evidence from this paper is inconclusive.

A 2016 coding competition and study called *Build It, Break It, Fix it (BIBIFI)* required teams to code software to specifications, and then attempt to break each others' solutions using hacking techniques [39]. The solutions and vulnerabilities were monitored and evaluated. A 2020 paper that quantitatively analysed the BIBIFI study data stated that there was no statistically significant effect on secure coding from having taken the related Massive Open Online Course (MOOC) [34]. This was not the whole story, however; the original 2016 BIBIFI paper stated that '*the effect size is quite large [though not statistically significant]; it is possible that the MOOC security training played a role*' [39]. This original paper's Table 5 reported a p-value of 0.086, so that the effect from MOOC participation may have narrowly escaped conventional statistical significance. If some non-MOOC participants had taken secure-coding classes elsewhere, this might have improved their secure-coding ability, and reduced the statistical significance of the difference between the two populations. Considering '*logic, background knowledge and experimental design*' alongside statistical significance [29], we argue that the finding that *MOOC participation* had no statistically significant effect on the secure coding outcome does not imply that *secure coding training* does not have an effect on secure coding. This is important because a paper describing qualitative analysis of the same BIBIFI data, published in 2020 and also reporting no statistically significant effect of MOOC completion on secure coding, concluded that '*it is likely that increased development experience and security training have, at most, a small impact*' [54]. Note that MOOC completion was conflated with training here; participants may have taken different training.

This confusion from the BIBIFI papers seems to have muddied the research waters. Mokhberi and Beznosov, citing the qualitative 2020 paper [54] in a recent literature review [27], said:

> 'The appearance of different types of security vulnerabilities did not seem to correlate significantly with [...] completed security training'

Later, they concluded that [27]:

> 'as some research suggests, security training is not a significant factor in preventing vulnerabilities [...] more investigation is necessary to understand why education is not working and how developers' education can be adjusted or even reformed'

We agree that more investigation is necessary. This paper is a first step towards exploring this issue further.

2.3 Software Security Learning

Researchers have examined the learning tools available to developers to help them to code securely, identifying security gaps in the resource ecosystem [3]. Qualitative studies have suggested that developers should use Stack Overflow (SO) to learn about software security [25], although laboratory studies have indicated that use of SO can result in insecure code [1, 15, 45]. Imaginative approaches to training such as those based on serious games are described in the literature [13, 18]. These studies have mixed results.

2.4 Training in the Workplace

Writing in the first decade of this century, van Wyk and Steven recommended that organisations should adopt interactive, engaging secure-coding training, tailored for different job roles and experience levels [52]. The academic literature on software security training in industry usually discusses a single implementation. For example, at Dropbox, participating in an internal capture-the-flag competition appeared to reduce security overconfidence in developers, made them more aware of a range of potential vulnerabilities, and improved their comfort-levels with documentation and the security team [55]. Considering security in cyber-physical systems, Crowther and Rust [12] advocated continuous training including gamification and other interactive training. The presence of training materials does not guarantee that they will be used, and researchers warn against optional training, or dull training [12, 46]. Annual security compliance training was described by a participant in a Haney et al. study as '*a layer of Dante's Hell*' [20].

Interviews with software security auditors suggest that organisations often tailor software-security training to the defects and challenges that occur in-house [38, 48]. Ethnographic researchers have observed that, within organisations, events such as audits, code reviews and the receipt of static analysis reports can create secure-coding learning opportunities [24]. On-the-job training is recommended by secure-coding experts [56], and is usually done by mentoring, embedding security champions within teams, or having security experts available for ongoing learning. Security champions themselves may benefit from on-the-job training, leveraging expertise from security groups [22]. Tailored and on-the-job training or mentoring occur in studies describing organisations with a positive security culture [21, 50]. Studying the effect of a one-off security consultancy and training event, Poller et al. observed that the security momentum dissipated over time and was not facilitated by the organisational structure [37]. An illuminating contrast was provided by Tuladhar et al., who documented the introduction of a software security drive by upper management in a software company [50]. They observed that lessons learned from formal software security training were initially difficult for software developers to apply to their daily work. Improvements in software security practice came from a positive security culture, with daily software security discussions, and situated learning.

Surprisingly, we could not find any previous work that discussed software security training in industry, identifying and evaluating the training currently available to developers within the workplace. We leveraged the knowledge gained from our literature review, and the information on training in practice gleaned from our developer survey, to begin to fill this gap.

3 METHODOLOGY

3.1 Generation of Positive Secure-Coding Training Features

The first research question asks how secure development training should be structured and delivered. We generated a list of desirable features of secure-coding training from a systematic literature review. This review considered papers related to developer-centred security from a large range of venues, and analysed them for activities that aided in secure development. The review procedures are discussed in more detail in a previous paper [40], and the in-depth review protocol is available online [43]. In the course of the review we encountered multiple papers that mentioned secure-coding training, as discussed in Section 2. However, little exploration of training quality and adequacy was done.

We generated a list of desirable software security training features from relevant literature review papers. Each paper was read carefully, and recommendations related to secure-coding training extracted. See Section 4 for further details.

3.2 Observation of Secure-Coding Training Features in Industry

The second research question concerns how secure coding training is structured and delivered in practice. While investigating factors that affect secure coding, we created a developer survey with questions about secure coding procedures in the working environment [42, 44]. The questionnaire was pilot tested in two phases. It was then publicised via personal contacts, a conference talk by the first author, and social media platforms such as Twitter, LinkedIn and Facebook. The survey was designed to be engaging. The first author publicised it daily over three months, generating 1,100 organically-sourced responses. Screening removed respondents who did not code frequently, either professionally or in an open source context. We also screened out non-programmers, and respondents who were answering at random, ending with 962 valid responses. Prior work gives a more detailed description of screening, and of the process of devising, testing and distributing the questionnaire [42].

In prior work, we analysed questionnaire answers to questions on secure coding practice and culture. We found that secure coding practice in the participants' environments ranged from non-existent to thorough [42, 44]. We also found evidence that some organisations with thorough secure coding practice appeared to be focused primarily on compliance, lacking a good secure coding culture [42].

The survey also included two questions on secure-coding training, which are the focus of this paper. Respondents were asked whether they had been offered secure-coding training. If they had, they were asked to describe it. 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 [16]. Therefore, the description was requested in free text. The free text comments were both informative and enjoyable to read. We have included several of them in this paper, to provide readers with a flavour of the richness of the responses. This qualitative data was coded by the first author to identify the features of secure-coding training offered, and any other themes. A description of the coding process can be found in Section 5.

We obtained approval for the questionnaire from our institution's Ethics Review Board. Correlation analyses were done in R using Pearson's correlation. Any quote from respondents is presented exactly as entered in the questionnaire.

4 SECURE-CODING TRAINING IN THEORY

In this section we attempt to answer Research Question 1: *What are the features of effective secure-coding training?* We base our answer

on the studies encountered in our systematic literature review, and on broader related research. Since we did not find any research explicitly focused on measuring the effectiveness of secure-coding training, we are unable to discuss statistical effect sizes. There is no statistical data available to suggest which training types are most effective, or which training features result in the most secure code. This area of research is in its infancy, and we hope that this paper will serve as a foundation to encourage others to explore these questions. The studies we encountered that discussed secure-coding training were ethnographic and interview studies. They describe the secure-coding training features that are most valued by developers and organisations. Table 1 provides a list of the recommended features and their sources, in four groups. In the following sections we discuss these four feature groups.

4.1 Relevance

As illustrated in Table 1, every paper we analysed for this work discussed the importance of making training **relevant** to developers and their context. While decontextualised learning can be effective in some situations [19], when developing with specific computer languages and frameworks, secure coding depends on intimate knowledge of the security qualities of the framework or language [35]. As stated by Larios-Vargas et al., '*developers perceive generic security training as too abstract, time-consuming, and ineffective*'.

The importance of training relevance was echoed during coding of respondent feedback, where an explicit criticism found in some free text comments was that training content was off-topic.

Related to relevance is the necessity for training to be **actionable**, and this was also mentioned in all the work we reviewed. Developers sometimes find it difficult to put secure-coding training into practice; this problem is ameliorated where the training is immediately relevant to the developer's work [50]. Ideally, developers work through examples in training which they can then apply directly when coding.

Furthermore, to be relevant, training should be **up-to-date**. Most of the papers that we analysed either suggested or implied that training should be **iteratively improved**, a process likely to keep the content fresh and topical. This is very much the opposite of designating an annual video for developers to watch, and then ticking the 'training' box. For example, a security auditor interviewed by Thomas et al. [48] said that:

> 'What we often do is periodically assess the category of vulnerabilities, what the kinds of volume and mistakes the developers are making and provide a very small part of training just on those things.'

This is an example of iterative improvement that makes the training relevant, actionable and up-to-date. It also illustrates how **errors-based** training, recommended by several of the papers we reviewed, can be implemented. The auditors interviewed by Thomas et al., conscious of the high proportion of developers to security experts, suggested that secure-coding tools could double as **training tools** if they accompanied error notifications with information about defects, secure coding and learning opportunities [48]. A common suggestion for content was that it should **include risks**.

	Crowther and Rust [12]	Arizon-Peretz et al. [5]	Larios-Vargas et al. [23]	Tuladhar et al. [50]	Thomas et al. [48]	Haney et al. [21]	Assal and Chiasson [7]
Relevance							
Relevant	●	●	●	●	●	●	●
Actionable	●	●	●	●	●	●	●
Up-to-date	◐	●	●	●	●	●	●
Iteratively improved	●	●	●	●	◐	○	◐
Errors-based	●	○	○	○	●	●	●
Include risks	●	○	●	●	○	○	●
Training tools	○	○	○	○	●	○	○
Time allocated							
Continuous	●	◐	●	●	○	●	○
Ongoing mentoring	○	○	○	●	○	●	○
Include time	●	○	●	○	○	●	○
Mandatory	●	○	○	○	◐	◐	◐
Engagement							
Engaging	●	○	○	○	●	○	○
Varied	●	○	●	●	○	○	●
Interactive	●	○	○	●	○	○	○
Gamified	●	○	○	○	●	○	○
Additional training features							
Available expert	○	○	○	●	●	●	○
Peer review	○	○	○	●	◐	●	○
All hierarchy	○	○	○	○	○	○	●
Non-tech skills	○	○	●	○	○	○	○

Table 1: Desirable Features for Secure Coding Training. ● = explicitly mentions the training feature. ◐ = implied only, but not explicitly mentioned. ○ = no discussion of this feature.

4.2 Time Allocated

As can be seen in Table 1, several of the papers we read emphasised the importance of **continuous** training for ongoing learning. Analysing how situated learning contributed to security culture in a software company, Tuladhar et al. found that ongoing discussion of security requirements and application of training knowledge in context is important. **Ongoing mentoring** is a useful way of delivering continuous training, and was mentioned in those papers that focused on having a secure-coding culture in the workplace [21, 50]. Several papers also mentioned that a secure-coding training regime should **include time** for developers to engage in self-directed learning, exploring secure-coding issues in depth. Several papers imply that **mandatory** training is essential, while Crowther and Rust, writing about their industry experience coding cyberphysical systems, emphasise that '*effective training must be mandatory*' [12].

4.3 Engagement

Two studies explicitly advise that training should be **engaging** to retain developer interest [12, 48]. This is easier when **varied** training materials and delivery vehicles are available to developers [23]. Training material that is **interactive** engages the user. **Gamified** training takes interactivity up a notch, providing the trainee with game-like rewards as they progress. A well-known general example is Duolingo, which gamifies natural language learning. Crowther and Rust note that '*a gamified delivery has increased the level of engagement, with the average user doing more training than the requirement*' [12].

4.4 Additional Training Features

There were four additional recommendations on secure training delivery that are somewhat related. Several qualitative studies of successful secure-coding environments note that they benefit when there is an **available expert** whom the developers can consult and from whom they can learn. Security **peer review** allows more expert developers to draw their peers' attention to security issues in code, leading to discussion and a learning opportunity. Assal and Chiasson advise that software security affects all of an organisation, and there should therefore be tailored, relevant training available throughout the hierarchy, from management to developers [7], a requirement defined in Table 1 as '**All hierarchy**.' This suggestion gains credence from a study comparing two organisations by Oyetoyan et al., who assessed the specific secure code training needs of developers, architects and testers, finding that they differed [32]. Developers may also benefit from some training on **non-tech skills**, more typically targeted at management. Larios-Vargas et al. [23] were the lone voice suggesting this, saying:

> 'most security challenges are indirectly related to conflict management, negotiation skills, communication ability, and empathy [...] we encourage organizations to include topics related to non-technical skills as part of any security training program, which is helpful in the context of security and boosts collaboration and productivity in the company.'

4.5 Indicators for a Poor Training Environment

Having absorbed all the advice discussed above, we considered indicators for a poor secure-coding training environment. Indicators for a poor training environment would be that the training is irrelevant, off-topic, dull, or infrequent, is not adjusted over time based on feedback and industry events, or is skipped due to tight deadlines.

5 SECURE-CODING TRAINING IN PRACTICE

In this section we assess the survey respondents' feedback on training in the light of our research questions. Since we only had two questions on secure-coding training, we consider this preliminary work. Future research in this area should consider the list of desirable features from Section 4 and include them in investigations.

5.1 Demographics

A detailed description of the sample can be found in prior work [42]. Respondents ranged in age from 18 to over 65. When asked what gender they most identified with, 82% of respondents answered *'Man,'* which is in line with other industry surveys [31]. Respondents selected or included 59 different countries as their working location.

5.2 Breakdown of Training Answers

To assess secure-coding training in practice, we evaluated answers from two questions in our secure-coding survey. Developers were asked '*Have you ever been offered training relevant to software security or privacy in your environment?*' with possible answers '*Not applicable*,' '*Yes*,' or '*No*.' Table 2 reports the results. Only 381 respondents, 39.6% of the total, answered 'Yes.' The next question was: '*If you answered yes to the previous question, please describe the training here.*' Input to this answer was in free text, as we did not want to constrain responses.

Of the possible 381 free-text descriptions of secure-coding training, 75 were left blank. The remaining 306 were coded by the first author. Each entry was initially evaluated and flagged as either relevant to secure coding training or not relevant. If not relevant, it was assigned to one of the codes in Table 3. These codes were generated as analysis proceeded, appearing in the survey data. Some respondents did not comment, or listed off-topic training; this included standard IT training such as anti-phishing, with no software development component.

Once all of the irrelevant comments had been flagged, only 220 comments remained that actually described secure-coding training. In the next section, we analyse these comments to attempt to answer Research Question 2.

Table 2: Whether the respondent states that they have been offered training relevant to software security or privacy in their environment. Only 39.6% of respondents replied 'Yes' to this question.

Response	Frequency	Percent
Yes	381	39.6
No	495	51.5
Not applicable	86	8.9

Table 3: Types of training description by the 381 respondents who stated that they had been offered training. 'Blank' indicates that no comment was made. 'IT Security' means that the respondent described standard IT security training, with no secure coding component. 'Privacy' means the description involved privacy only. 'OT Comment' indicates that the respondent did not actually discuss training when answering this question. 'Description' indicates that text compatible with secure-coding training was supplied. The examples are complete verbatim quotes; many responses were brief.

Category	n	%	Example response text
Blank	75	19.7	*NA*
IT Security	40	10.5	'*Mostly how to avoid phishing attack vectors*'
Privacy	27	7.1	'*Video lessons about the Brazilian GDPR (LGPD)*'
OT comment	19	5	'*If by offered, you mean sales offerings.*'
Description	220	57.7	'*internal, often language or platform specific*'

5.3 Research Question 2: Are developers offered secure coding training, and if so, does it have the positive features we have identified?

In this section we explore the descriptions of secure coding training given by participants in our survey, and whether they correspond with theoretical features of good training described in Section 4. The descriptions were coded by the first author. Key features identified from the literature review were assessed. Several additional features were found in survey responses. Two respondents' security training was so good that they refused to divulge any details, saying simply *'Confidential.'*

5.3.1 Relevance. On analysing the 220 comments that described secure-coding training content, we discovered that 160 indicated that the content was slightly or very relevant (see Table 4). Inevitably, some comments were negative, such as *'Boring'* and *'Dry training courses around secure development in conjunction with sales from SAST providers.'* Others were very positive, and these tended to describe relevant training. For example:

> 'My team offers a security champion program at our workplace :), other than that we have an internal, mandatory security training course for all employees, and optionally free courses on udemy / generous budget if necessary for further learning.'

An example of training that lacked relevance:

> ' "Secure Code Warrior" – an online platform that shows sample insecure code, and asks you to identify "vulnerabilities" and pick the correct remediation from multiple choice. Very broad and unfocused, targets niches that are not applicable to our team, mostly a waste of time.'

It is interesting that Secure Code Warrior was mentioned in positive terms by six participants but negatively here. In this case it seems that context was not considered when the training was chosen, which, if it is happening generally, is cause for concern.

Table 4: Whether training was relevant and interactive.

Feature	No	Not really	Unknown	Slightly	Very
Relevant	6	8	46	69	91
Interactive	33	5	149	9	24

5.3.2 Time allocated. In our list of positive features of good secure-coding training we included several related to time, including 'Continuous.' When coding responses, we found that a 'Frequency' variable was needed (see Table 5), because durations varied widely. There were 35 participants whose training seemed to be continuous, while 31 were given annual or one-off training. Annual, or even less frequent, training suggests a compliance approach that does not add value.

Table 5: Training frequency. Frequency for 133 participants was unknown. 'Old' means the training took place years ago.

Continuous	Frequent	Monthly	Regular	Annual	One-off	Old
35	8	2	7	19	12	4

Two other features relevant to time are whether training involves ongoing mentoring, and whether training is mandatory. Our findings on these values can be seen in Table 6. Ongoing mentoring allows the developer to obtain timely and relevant help. We only identified eleven free-text entries that had explicit or implicit references to mentoring, while in 39 of the entries it was clear that no mentoring was taking place.

When training is not mandatory it can be skipped due to deadline pressures. If this is allowed it conveys the message that secure coding is not important [6]. We identified six instances where training was not mandatory, and 32 where it was. Five participants explicitly told us that they did not or could not attend security training. One participant said *'I can't [describe the training], I ended up not going to it due to project pressures.*'

Table 6: Whether training included mentoring, whether it was mandatory.

Feature	No	Unknown	Yes
Ongoing mentoring	39	170	11
Mandatory	6	182	32

5.3.3 Engagement. Due to the brevity of most of the answers, the easiest engagement feature to assess was interactivity (see Table 4). While 38 participants seemed to receive training that was not interactive, training for 33 others was slightly or very interactive. This shows the large range of effectiveness in the training reported.

5.4 Features Added from Survey Data

A number of additional features were found in the survey data. Some were positive, some less so. We describe the top three here.

'**Timely**' indicates that help is available when needed. For example, one participant said '*We can schedule security reviews with a dedicated security team*,' which is an indicator that on-the-job training is available. Seven comments had the 'timely' feature.

'**Compliance focus**' indicates that the training may be a box-ticking exercise. There were 13 comments that appeared to indicate a compliance focus, for example:

> 'None of it was practical – it was all "*Read this document that says we are security focused, and then sign off that you read the document. Your security training is now complete*". '

> 'The training was fairly pro forma security training that, frankly, didn't teach me anything I didn't already know, but let my company tell clients that it is an annual training all of us do.'

'**Certification**' indicates that the training included aspects aimed at certification. There were nine such entries. An example is '*Pentester lab licenses, regular CTF events.*'

Table 7: Developer tone discussing secure-coding training.

Very Negative	Negative	Neutral	Positive	Very Positive
5	17	164	31	3

5.4.1 Developer Tone. The tone of developers' text entries varied widely, so we analysed it; Table 7 reports the results. Some training was considered terrible by the participant: *'It was honestly awful – unbelievably bad. Less than useless.'* Other participants were very positive: '*Many options by internal security teams and external trainers including web technology focused trainings, tool/language best practices, secure coding puzzles, etc.*'

Overall, we found that while most of the participants' descriptions were neutral, 17 were negative and 5 very negative. Positive

comments comprised 31 of the responses, with 3 very positive entries. Developer sentiment correlated with the relevance of training, with a correlation of .497 and a p-value less than 0.001.

6 DISCUSSION

Training in the workplace has symbolic significance, signalling management priorities, as highlighted by Arizon-Peretz et al. [6]:

> 'A lack of effective training for handling security and privacy concerns may convey two messages to employees: (a) that the employee is not really expected to handle them, and (b) that privacy and security are not considered main goals, even if they are declared to be important.'

Aside from signalling intent, training should also impart knowledge and improve skills in the targeted area. The question of whether secure coding training improves developers' secure coding performance is a vexed one. We found very little work that considers this question. Where the effect of secure-coding training is discussed, it is as a secondary concern in studies whose main focus is elsewhere (see Section 2.2). It is somewhat surprising that secure-coding training has not yet been formally evaluated by the academic community. The explanation may lie partly in the use of an analogy with computer security for general computer users made by Wurster and von Oorschot in 2008 [58], and more recently by Acar et al. [2]. Wurster and von Oorschot [58] argued that:

> 'While it is generally accepted that "more user training" is not a viable solution to some security problems, apparently many in the security community continue to believe that developer education will solve the [software security] problem...we argue the best approach is to develop and enforce technical solutions instead of assuming that developers can be educated to *do the right thing* [emphasis theirs]'

We agree that it is important to create technical solutions [58] and usable security tools [2]. However, it is also essential to educate software developers about secure coding in general, and about the secure-coding traits of the platforms and development tools they are using in particular [56]. Software that does not itself process sensitive data may be used as a stepping-stone to other parts of a system [8]. Software security should be a central concern for most software developers, in a context where much of the software they write will be under continuous attack [17, 53]. Software security is a broad and far-reaching problem. There is no silver bullet. Like cybersecurity in general, an approach of defence in depth is necessary. Programmers cannot be left in ignorance of what is needed.

Our research questions are addressed in Sections 4 and 5. To answer RQ1 we analysed several qualitative studies which described the training features valued by developers and security experts. These features may appear to be trivial, or just common sense, but evaluating RQ2 we discovered that most survey respondents were not even offered secure-coding training. When it was offered, basic attributes from RQ1 such as relevance, frequency and timeliness were often missing. Quality varied widely, as did the tone of training descriptions. Some had a neutral tone. Negative comments tended to describe the training as irrelevant, while positive comments described varied, easily available and targeted training. This work provides a grounding for future research, defining some of the differentiating features which should be assessed when evaluating training. Future work could explore grey literature on this subject, and practitioners' assessments of secure-training needs.

Developers have been observed not to apply their security knowledge to a task that obviously requires security, storing passwords in a database, unless asked [28]. In ethnographic research, developers tolerated known security issues in code because they reduced support calls [33]. Such findings suggest that leadership in organisations should overtly prioritise good security culture, including ongoing learning and mentoring [21, 50, 56]. More research is needed to ensure that training budgets are wisely spent, and well-resourced.

6.1 Limitations

Extraction of training features from the literature was done by a single author, which increases the chances of bias. However, the analysis was conducted rigorously, and notes were kept throughout. The current study is exploratory and designed to initiate discussion; further research could replicate and extend this study.

Coding of survey responses was also done by a single author. The responses were short, with the longest being only 583 characters (98 words) and the mean number of characters being 78. We considered this a sufficiently simple sample to be coded by one researcher, with consultation with co-authors when required.

7 CONCLUSION

There is little research available on types and effectiveness of secure-coding training. This paper makes a first attempt to address the issue. We began by analysing the training-relevant papers from a prior literature review, to define the features of good secure-coding training. These features are based on developer, security auditor and researcher observations in relevant qualitative research.

We then analysed the answers to training questions in our secure-coding survey (n=962) to find whether the training being offered to developers has the features we identified from the literature. Unfortunately, only 39.6% of respondents stated that they had been offered training relevant to software security or privacy in their coding environment. Only 220 respondents, or 22.9%, described their secure-coding training. Negative or very negative sentiments on the training were expressed by 22 respondents. At least five could not attend. We found indications that training was sometimes subpar, being compliance-focused, lacking context or relevance, or being insufficiently frequent. Positive developer tone correlated positively with the relevance of training. Research is needed on how to ensure excellent developer secure-coding training in industry.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their helpful comments and input. This publication was financially supported by Science Foundation Ireland under Grant numbers 18/CRT/6222, 13/RC/2077_P2, 13/RC/2094_P2, and 15/SIRG/3293.

REFERENCES

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

[2] Y. Acar, S. Fahl, and M. L. Mazurek. 2016. You are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users. In *IEEE Cybersecur. Dev.* 3–8.

[3] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek, and S. Fahl. 2017. Developers Need Support, Too: A Survey of Security Advice for Software Developers. In *IEEE Cybersecur. Dev.* 22–26.

[4] M. Almansoori, J. Lam, E. Fang, K. Mulligan, A.G.S. Raj, and R. Chatterjee. 2020. How Secure are our Computer Systems Courses?. In *2020 ACM Conference on International Computing Education Research*. ACM, New York, NY, USA, 271–281.

[5] R. Arizon-Peretz, I. Hadar, and G. Luria. 2022. The Importance of Security Is in the Eye of the Beholder: Cultural, Organizational, and Personal Factors Affecting the Implementation of Security by Design. *IEEE Transactions on Software Engineering* 48, 11 (2022), 4433–4446.

[6] R. Arizon-Peretz, I. Hadar, G. Luria, and S. Sherman. 2021. Understanding developers privacy and security mindsets via climate theory. *Empirical Software Engineering* 26, 6 (2021).

[7] H. Assal and S. Chiasson. 2018. Security in the Software Development Lifecycle. In *14th USENIX Conference on Usable Privacy and Security*. USA, 281–296.

[8] K. Beck. 2018. Hackers exploit casino's smart thermometer to steal database info. https://mashable.com/article/casino-smart-thermometer-hacked.

[9] J. Boote, E. Erlikhman, S. Gardner, and S. Migues. 2022. BSIMM 13. https://www.bsimm.com/.

[10] L. Braz, E. Fregnan, G. Çalikli, and A. Bacchelli. 2021. Why Don't Developers Detect Improper Input Validation?': DROP TABLE Papers. In *ACM/IEEE 43rd Int. Conf. Softw. Eng.* 499–511.

[11] S. Chung, L. Hansel, Y. Bai, and V. Popovsky. 2014. What approaches work best for teaching secure coding practices? (2014).

[12] K. G. Crowther and B. Rust. 2020. Built-in cybersecurity: Insights into product security for cyberphysical systems at a large company. *IEEE Secur Priv* 18, 5 (2020), 74–79.

[13] T. Espinha Gasiba, I. Andrei-Cristian, U. Lechner, and M. Pinto-Albuquerque. 2021. Raising Security Awareness of Cloud Deployments using Infrastructure as Code through CyberSecurity Challenges. In *The 16th Int. Conf. on Availability, Reliability and Security*. ACM, 1–8.

[14] T. Espinha Gasiba, U. Lechner, M. Pinto-Albuquerque, and D. Méndez. 2021. Is Secure Coding Education in the Industry Needed? An Investigation Through a Large Scale Survey. In *IEEE/ACM 43rd International Conference on Software Engineering (SEET)*. 241–252.

[15] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl. 2017. Stack Overflow Considered Harmful? The Impact of Copy Paste on Android Application Security. In *IEEE Symp. on Security and Privacy*. 121–136.

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

[17] N. Ford. 2023. List of Data Breaches and Cyber Attacks in 2023. https://www.itgovernance.co.uk/blog/list-of-data-breaches-and-cyber-attacks-in-2023.

[18] M. Gutfleisch, M. Schöps, S. Sayin, F. Wende, and M.A. Sasse. 2022. Putting Security on the Table: The Digitalisation of Security Tabletop Games and its Challenging Aftertaste. In *IEEE/ACM 44th International Conference on Software Engineering (Software Engineering Education and Training)*. 217–222.

[19] M. Guzdial. 2010. Does contextualized computing education help? *ACM Inroads* 1, 4 (2010), 4–6.

[20] J.M. Haney and W.G. Lutters. 2018. '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)*. 411–425.

[21] J. Haney, M. Theofanos, Y. Acar, and S. Spickard Prettyman. 2018. "We make it a big deal in the company": Security Mindsets in Organizations that Develop Cryptographic Products. In *14th Symp. Usable Priv. Secur.* 357–373.

[22] M.G. Jaatun and D. Soares Cruzes. 2021. Care and Feeding of Your Security Champion. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. IEEE, 1–7.

[23] E. Larios-Vargas, O. Elazhary, S. Yousefi, D. Lowlind, M. L. W. Vliek, and M.-A. Storey. 2023. DASP: A Framework for Driving the Adoption of Software Security Practices. *IEEE Trans. Softw. Eng.* (2023).

[24] T. Lopez, H. Sharp, T. Tun, A. K. Bandara, M. Levine, and B. Nuseibeh. 2019. "Hopefully We Are Mostly Secure": Views on Secure Code in Professional Practice. In *12th Int. Workshop on Cooperative and Human Aspects of Softw. Eng.* 61–68.

[25] T. Lopez, T.T. Tun, A. K. Bandara, M. Levine, B. Nuseibeh, and H. Sharp. 2020. Taking the Middle Path: Learning About Security Through Online Social Interaction. *IEEE Software* 37, 1 (2020), 25–30.

[26] V. Mohan and L. B. Othmane. 2016. SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps. In *11th Int. Conf. on Availability, Reliability and Security*. 542–547.

[27] A. Mokhberi and K. Beznosov. 2021. SoK: Human, Organizational, and Technological Dimensions of Developers' Challenges in Engineering Secure Software. In *European Symp. on Usable Security*. ACM, 59–75.

[28] A. Naiakshina, A. Danilova, E. Gerlitz, E. von Zezschwitz, and M. Smith. 2019. "If you want, I can store the encrypted password": A Password-Storage Field Study with Freelance Developers. In *Conf. Hum. Factors Comput. Syst.* ACM.

[29] Nature. 2019. It's time to talk about ditching statistical significance. *Nature* 567 (2019), 283.

[30] D. Oliveira, M. Rosenthal, N. Morin, K.-C. Yeh, J. Cappos, and Yanyan Zhuang. 2014. It's the psychology stupid: how heuristics explain software vulnerabilities and how priming can illuminate developer's blind spots. ACM, 296–305.

[31] Stack Overflow. 2020. Stack Overflow Developer Survey - Gender. https://insights.stackoverflow.com/survey/2020/#demographics.

[32] T. D. Oyetoyan, D. S. Cruzes, and M. G. Jaatun. 2016. An Empirical Study on the Relationship between Software Security Skills, Usage and Training Needs in Agile Settings. In *11th Int. Conf. on Availability, Reliability and Security*. 548–555.

[33] H. Palombo, A.Z. Tabari, D. Lende, J. Ligatti, and X. Ou. 2020. An Ethnographic Understanding of Software (In)Security and a Co-Creation Model to Improve Secure Software Development. In *16th Symp. Usable Priv. Secur.* 205–220.

[34] J. Parker, M. Hicks, A. Ruef, M. L. Mazurek, D. Levin, D. Votipka, P. Mardziel, and K. R. Fulton. 2020. Build It, Break It, Fix It: Contesting Secure Development. *ACM Transactions on Privacy and Security* 23, 2 (2020), 10:1–10:36.

[35] O. Pieczul, S. Foley, and M. E. Zurko. 2017. Developer-centered security and the symmetry of ignorance. In *New Security Paradigms Workshop*. ACM, 46–56.

[36] V. Pikulin, D. Kubo, K. Nissanka, S. Bandara, M.A. Shamsiemon, A. Yasmin, A. Jayatilaka, A. Madugalla, and T. Kanij. 2023. Towards Developer-Centered Secure Coding Training. In *38th IEEE/ACM ASE Workshops*. 24–31.

[37] A. Poller, L. Kocksch, S. Türpe, F.A. Epp, and K. Kinder-Kurlanda. 2017. Can Security Become a Routine? A Study of Organizational Change in an Agile Software Development Group. In *2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. ACM, New York, NY, USA, 2489–2503.

[38] P. Rajba. 2018. Challenges and mitigation approaches for getting secured applications in an enterprise company. In *13th Int. Conf. on Availability, Reliability and Security*. ACM.

[39] A. Ruef, M. Hicks, J. Parker, D. Levin, M.L. Mazurek, and P. Mardziel. 2016. Build It, Break It, Fix It: Contesting Secure Development. In *2016 ACM SIGSAC CCS*. ACM, New York, NY, USA, 690–703.

[40] I. Ryan, U. Roedig, and K.J. Stol. 2023. Unhelpful assumptions in software security research. In *2023 ACM SIGSAC CCS*. ACM.

[41] I. Ryan, U. Roedig, and K.-J. Stol. 2022. Insecure Software on a Fragmenting Internet. In *Cyber Research Conf. - Ireland (Cyber-RCI)*. 1–9.

[42] I. Ryan, U. Roedig, and K.-J. Stol. 2023. Measuring Secure Coding Practice and Culture: A Finger Pointing at the Moon is not the Moon. In *IEEE/ACM 45th International Conference on Software Engineering*. 1622–1634.

[43] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2023. Study Protocol for 'Assumptions in Software Security Literature'. https://osf.io/7a2dr/

[44] I. Ryan, K.-J. Stol, and U. Roedig. 2023. 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)*. 37–44.

[45] I. Ryan, K.-J. Stol, and U. Roedig. 2023. Studying Secure Coding in the Laboratory: Why, What, Where, How, and Who?. In *2023 IEEE/ACM 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. 23–30.

[46] L. Sajwan, J. Noble, C. Anslow, and R. Biddle. 2021. Why Do Programmers Do What They Do? A Theory of Influences on Security Practices. In *Workshop on Usable Security and Privacy*.

[47] M. Tahaei and K. Vaniea. 2019. A Survey on Developer-Centred Security. In *IEEE European Symp. on Security and Privacy Workshops (EuroS&PW)*. 129–138.

[48] T. W. Thomas, M. Tabassum, B. Chu, and H. Lipford. 2018. Security During Application Development: An Application Security Expert Perspective. In *Conf. Hum. Factors Comput. Syst.* ACM, New York, NY, USA.

[49] N. Tomas, J. Li, and H. Huang. 2019. An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps. In *Int. Conf. on Cyber Security and Protection of Digital Services (Cyber Security)*. 1–8.

[50] A. Tuladhar, D. Lende, J. Ligatti, and X. Ou. 2021. An analysis of the role of situated learning in starting a security culture in a software company. In *17th Symp. Usable Priv. Secur.* 617–631.

[51] Akond Ashfaque Ur Rahman and Laurie Williams. 2016. Security practices in DevOps. In *Symposium and Bootcamp on the Science of Security*. ACM, 109–111.

[52] K.R. van Wyk and J. Steven. 2006. Essential Factors for Successful Software Security Awareness Training. *IEEE Security & Privacy* 4, 5 (2006), 80–83.

[53] C. Vasquez. 2023. CISA's Goldstein wants to ditch 'patch faster, fix faster' model. https://cyberscoop.com/cisa-goldstein-secure-by-design/.

[54] D. Votipka, K. R Fulton, J. Parker, M. Hou, M. L Mazurek, and M. Hicks. 2020. Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It. In *29th USENIX Security Symp.* USENIX Assoc., 109–126.

[55] D. Votipka, M. L Mazurek, Hongyi Hu, and Bryan Eastes. 2018. Toward a Field Study on the Impact of Hacking Competitions on Secure Development.

[56] C. Weir, I. Becker, J. Noble, L. Blair, A. Sasse, and A. Rashid. 2019. Interventions for software security: creating a lightweight program of assurance techniques for developers. In *IEEE/ACM 41st Int. Conf. Softw. Eng.* 41–50.

[57] J. Witschey, O. Zielinska, A. Welk, E. Murphy-Hill, C. Mayhorn, and T. Zimmermann. 2015. Quantifying developers' adoption of security tools. In *10th Joint Meeting of ESEC/FSE*. 260–271.

[58] G. Wurster and P. C. van Oorschot. 2008. The developer is the enemy. In *2008 New Security Paradigms Workshop*. ACM, 89–97.