CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Software security is your highest priority. Do your developers know that?
Authors	Ryan, Ita;Roedig, Utz;Stol, Klaas-Jan
Publication date	2026-03-26
Original Citation	Ryan, I, Roedig, U & Stol, K-J 2026, 'Software security is your highest priority. Do your developers know that?', IEEE Security and Privacy, vol. 24, no. 3, pp. 43-51. https://doi.org/10.1109/MSEC.2026.3670641
Type of publication	Article (peer-reviewed)
Link to publisher's version	https://www.scopus.com/pages/publications/105034435238 - 10.1109/MSEC.2026.3670641
Rights	© 2026, IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. - https://creativecommons.org/licenses/by/4.0/
Download date	2026-11-06 09:55:57
Item downloaded from	https://hdl.handle.net/10468/18669

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Software Security Is Your Highest Priority. Do Your Developers Know That?

Ita Ryan, Utz Roedig, and Klaas-Jan Stol | University College Cork

Organizations producing software may claim to prioritize secure development, without supporting secure practices or a security culture. We ran a global survey on coders' workplaces and found gaps in practice, culture, and training. We advocate emphatically prioritizing and supporting secure coding.

An onslaught of cyberattacks is gathering pace around the world.[a] The cost of cybercrime continues to increase. Cyberattacks increasingly exploit zero-day vulnerabilities that were previously undiscovered.[b] Encouraging the development of secure hacker-resistant software is ever more essential.

It is difficult, if not impossible, to develop perfectly secure software. The escalating number of security vulnerabilities in production software suggests that, in fact, software developers sometimes do not address security at all—and certainly not in the holistic fashion recommended by experts. Advice has included providing developers with safe defaults, useful resources,[1] and usable libraries and tools.[2]

Usability of developer tools is key, but it does not tell the whole story. Naiakshina et al.[3] conducted a series of laboratory experiments on how student, freelance, and professional developers store passwords, a task that would seem self-evidently to require security. In each experiment, they evaluated the effect of asking, or not asking, the developer to code securely. When a developer was asked to code securely, the odds that their solution would contain secure password storage were up to 30 times higher. This effect was considerably greater than that of the library used. Even where freelance participants believed that they were developing production software, they were more than three times more likely to present a secure solution when asked to do so.[4] While these laboratory experiments were conducted in contrived settings and did not consider team and organizational factors such as available secure coding support, the effect held in repeated experiments with multiple developer types. Safer tools and libraries may be of little use if developers are not encouraged to use them.

This finding adds urgency to investigations of security prioritization in developers' coding environments, as did a survey of North American developers ($n = 123$), which found that the strongest secure coding deterrents are its perceived irrelevance and "competing priorities and no plan."[5] Meticulous tracking of whether individual developers are coding securely or not may be missing the point. Have they been *asked* to code securely? What *direction* are they receiving from the organization or environment in which they write code?

Existing surveys in this area have focused on human factors.[5] We did not find work analyzing training, attempting to measure practice and culture, or assessing practice for under-resourced developers. Previous studies often used paid recruitment sources, which do not always validate respondents.[6] To obtain a greater understanding of actual secure coding practices, culture, and training in real development environments, we undertook a research program that comprised, first, an extensive literature review on secure coding,[7] and second, a

[a]https://www.enisa.europa.eu/publications/enisa-space-threat-landscape-2025.
[b]https://socprime.com/blog/latest-threats/cve-2026-21509-vulnerability/.

Digital Object Identifier 10.1109/MSEC.2026.3670641

Copublished by the IEEE Computer and Reliability Societies

This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

large-scale international survey based on the literature review findings. The survey asked questions about several different aspects of software security and received responses from software coders on all continents except Antarctica (n = 1,048).

We wrote four papers analyzing different aspects of the survey results.[8,9,10,11] In this work, we summarize the key findings from these four papers that could benefit organizations seeking to improve their secure coding practices. In the first paper, we found that even when sound practices are established, security culture issues may undermine them, communicating to developers that security is not a priority.[8] In the second paper, we found correlations between positive developer sentiment and an explicit prioritization of software security.[9] In the third paper, we evaluated secure coding training and determined that good training should be relevant, up to date, and actionable.[10] Survey results indicated that most developers do not receive timely and relevant training. The fourth paper analyzed secure coding practices for five cohorts of isolated and potentially under-resourced developers.[11] While measures of practice may be more applicable to large organizations, secure coding tools could help under-resourced developers to access existing industry knowledge. We found evidence that such tools are underutilized in these cohorts.

Surveying Developers in the Wild

We designed an online global developer survey, which was approved by our institution's ethics review board. Respondents were asked to give their informed consent when submitting their responses. We used convenience sampling to obtain a good global cross-section of developers from varied backgrounds. We recruited respondents through several channels, emphasizing that no knowledge of, or interest in, software security was required. Approximately 30 respondents were recruited via personal contacts and during an industry talk. The remainder were recruited by asking more than 350 popular online programming experts, who used a wide range of languages and platforms, to post about the survey. At least 50 experts posted the survey to their communities. We had received a total of 1,100 responses from 59 countries when the survey closed on 31 January 2022.[c] Of these, 1,048 were from frequent coders, of whom 962 passed predetermined screening tests; these 962 responses were used for further analysis.

We used R for statistical analysis, with Spearman's rho for correlations, and applied Bonferroni correction where necessary. In the data analysis figures to follow, the value of "n" is given with each figure. It varies because the survey's nonscreening questions were not mandatory. Coding of training responses was done by the lead investigator. Responses were brief, so this was deemed sufficient. Further detailed descriptions of our methodology can be found in our previous work.[8,9,10,11]

What Are Organizations Doing?

Our primary focus in the survey was on software security practice and culture, which were analyzed in the first paper.[8] Researchers have not converged on a single method to measure software security practice. Therefore, we created an empirically grounded instrument to measure practice by asking about the 12 secure coding activities that are most commonly present in organizations across industry (Table 1). Weir et al.[12] identified these activities by analyzing data from the long-running BSIMM project, which does detailed research into the software security practices of more than 100 organizations every year. We asked respondents whether each of these 12 common activities (CAs) is used in their coding environment. We added the use of the 12 CAs together to give a number from zero to 12, which we call the "CA score." While a lightweight security measurement, it is grounded in empirical observation.[12] Figure 1 shows the number of CAs identified by the 962 valid survey respondents. We found that there was a wide range of security stances in our respondents' working environments, confirming previous research on a smaller scale[5] showing that some work environments do not adopt secure coding practices.

Even something as serious as a code breach does not necessarily cause improvements. More than half of organizations, teams, or individuals who experienced a breach in their code improved their security practices in the long term. However, approximately 25% of individuals and teams who experienced a breach did not improve their secure coding practices. The remainder implemented security improvements that were short-lived. Process improvements were abandoned within a few months.

Security Culture Is Important

The first paper analyzing survey results also addressed the issue of security culture since poor security culture is one possible reason why security practices are not undertaken, and warning signs like code breaches are not leveraged for improvement. Haney et al.[13] defined a security culture as "a subculture of an organization in which security becomes a natural aspect in the daily activities of every employee." Without it, security practice can have a perfunctory box-ticking character. Some respondents' free-text comments vividly illustrated this difference. For example

[c] These data are available at https://osf.io/tfxp9/?view only=5fb840be1d6b43d19c54ebade77419de

Table 1. The 12 CAs. Respondents were asked, "Are you aware of this practice in your coding environment?"

No.	Question
Q1	Static analysis tools are brought into the code review process to make the review more efficient and consistent.
Q2	Compliance constraints are translated into software requirements for individual projects and are communicated to the engineering teams.
Q3	QA efforts go beyond functional testing to perform basic adversarial tests and probe simple edge cases and boundary conditions, with no particular attacker skills required.
Q4	QA targets declarative security mechanisms with tests derived from requirements and security features. A test could try to access administrative functionality as an unprivileged user, for example, or verify that a user account becomes locked after some number of failed authentication attempts.
Q5	Penetration test tools are used internally.
Q6	Emergency codebase response can be done. The organization or team can make quick code and configuration changes when software (for example, application, API, microservice, and Q infrastructure) is under attack.
Q7	Defects found in operations are entered into established defect management systems and tracked through the fix process.
Q8	Bugs found in operations monitoring are fed back to development and may change developer behavior. For example, viewing production logs may reveal a need for increased logging.
Q9	Penetration testing results are fed back to engineering through established defect management or mitigation channels, with development and operations responding via a defect management and release process.
Q10	Host and network security basics are in place across any data centers and networks and remain in place during new releases.
Q11	Security-aware reviewers identify the security features in an application and its deployment configuration (authentication, access control, use of cryptography, etc.) and then inspect the design and runtime parameters for problems that would cause these features to fail at their purpose or otherwise prove insufficient.
Q12	External penetration testers are used to identify security problems.

QA: quality assurance.

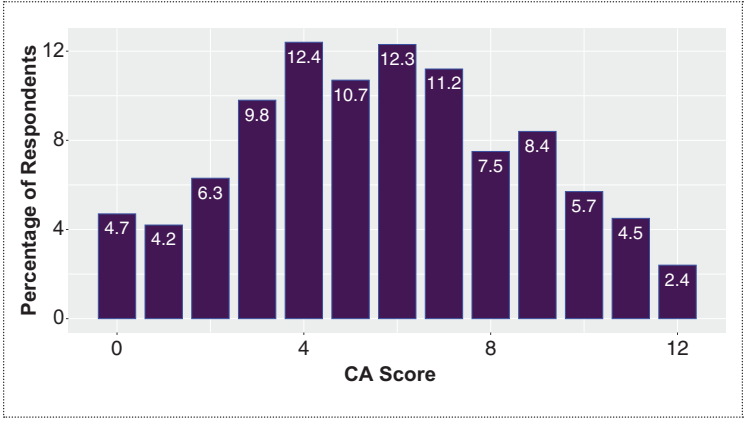


Figure 1. The CA score, a measure of secure coding practice, for the work environments of all valid survey respondents (n = 962).

> "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."

To evaluate security culture, we devised a number of questions grounded in a systematic review of prior literature.[8] The responses to these questions suggested that more than half of coding environments may suffer from poor security cultures. Figures 2 and 3 show grouped Likert-scale graphs that illustrate the correlations between respondents' CA scores (security practice) and their answers to two of these security culture questions. For example, respondents in work environments with low CA scores tended to indicate that software security was not prioritized in their environment (see Figure 2). The top line in the same graph gives the breakdown for those with the highest possible CA score of 12. It shows that the vast majority of developers in work environments with a CA score of 12 reported that software security is prioritized in their work environment. Yet Figure 3 shows that well over 50% of respondents in environments with a CA score of 12 spent *less than 2 hours a week* on software security. This provides a hint that a secure coding culture may be lacking in some work environments. A single question cannot unearth issues with certainty. For example, a low amount of time explicitly spent on secure development could

reflect a process where security is so baked in that it permeates all activities rather than being considered a separate practice. However, together, the culture questions suggested in the study, which ask about topics like tool adoption, security prioritization, support, and communications, can be used to probe more deeply into the secure coding stance of a work environment. Business leaders can use the CA score and security culture questions to help understand what practices they should introduce and what implicit signals they are sending to developers about the importance of software security.

Developers Are Trying

The work environment constrains developers' ability to act and can facilitate or discourage security initiatives.[14] Developers' individual software security sentiment is another factor in code security. Developers' personal experience and ability affect their response to secure coding tasks. In the second paper,[9] we assessed developer sentiment for survey respondents working within organizations ($n = 863$).[9]

We asked developers whether they were interested in coding securely (see Figure 4), whether they had the knowledge to code securely (see Figure 5), and whether they enjoyed coding securely (see Figure 6), three questions that together we describe as developer sentiment. It is clear that their feelings of interest and enjoyment exceed their feelings of competence, reflected in their assessment of their knowledge (see Figures 4 and 6 versus Figure 5).

We investigated how sentiment correlated with measurements of security practice and culture. We used factor analysis to distill responses to the security culture questions down to two factors: environment and empowerment.[9] The "environment" factor is associated with time, support, a perception of security prioritization, and good secure coding communication. The "empowerment" factor is associated with the developer's ability to proactively introduce security tools and raise security concerns. We examined how these two factors correlate with developer sentiment and the CA score. Within organizations, the environment factor correlates with all three measures of developer sentiment and with the CA score that measures software security practice (see Table 2). The empowerment factor correlates with the CA score only; interest, knowledge, and enjoyment were not correlated with empowerment. These results suggest that positive developer secure coding sentiment is facilitated by explicitly prioritizing secure coding, communicating about it, budgeting time for it, and providing related supports.

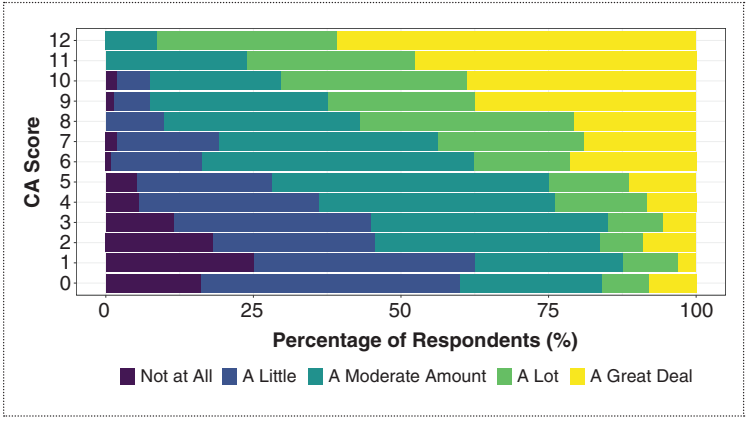


Figure 2. How highly do you think your team prioritizes software security? ($n = 896$). Most developers with a top CA score of 12 (91%) feel that their team prioritizes security a lot or a great deal.

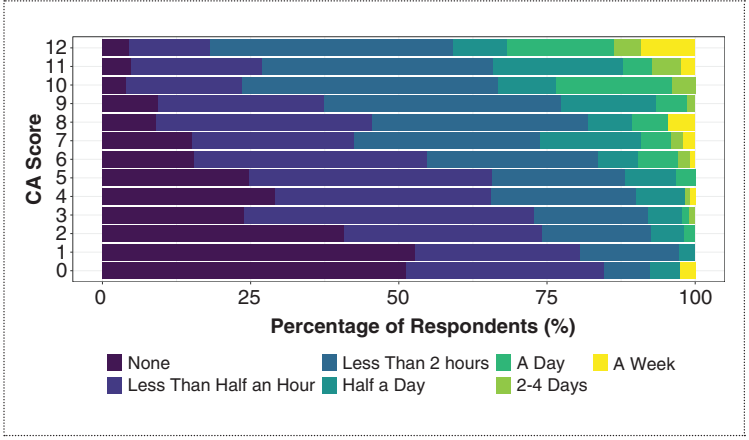


Figure 3. Roughly how much time do you spend on software security in an average week ($n = 878$)? Most developers with a top CA score of 12 (59%) spend approximately 2 hours a week or less on software security.

Train Your Developers!

The third paper[10] analyzed the responses to questions on software security training that were asked in the survey. Providing developers with secure coding training is an excellent way to reinforce the message that software security is a priority. An extensive review of software security literature revealed a surprising shortage of analysis on the best approaches to training developers to code securely.[7] This could be due in part to a gradually assembled academic consensus that secure coding training does not increase the security of code produced. This consensus is contrary to the extensive professional development experience of the lead investigator. It appears to be based on insufficiently fine-grained analysis, as discussed in this third paper.[10]

We did not find studies of the impact of secure coding training on software security, of the content of the training adopted by industry, or of the most

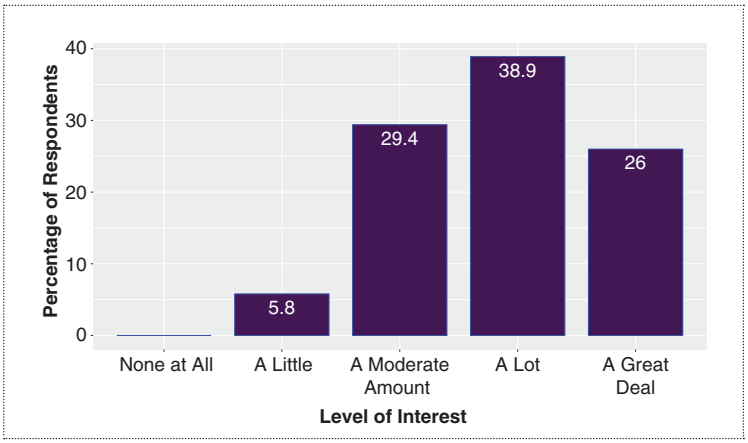


Figure 4. What level of interest do you have in developing secure software (n = 862)?

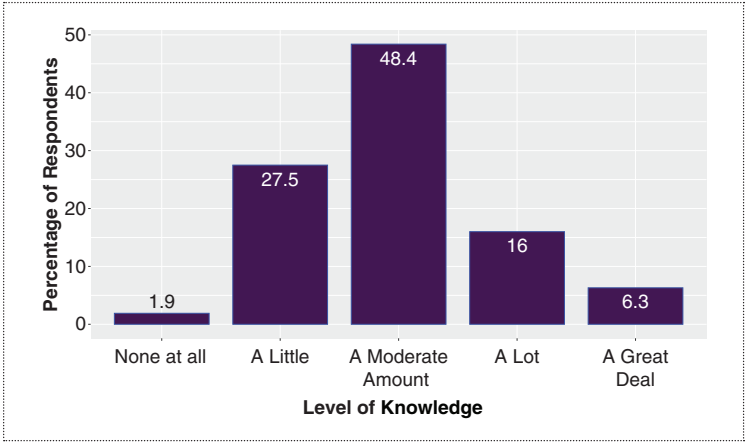


Figure 5. What level of software development security knowledge would you say you have (n = 863)?

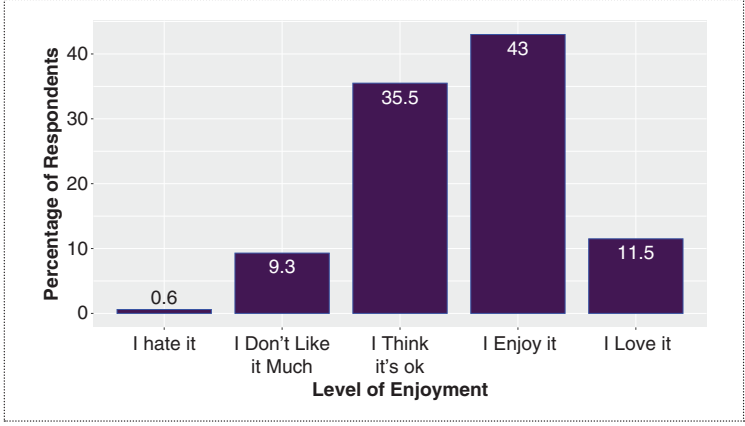


Figure 6. How much do you enjoy the challenge of attempting to code securely (n = 814)?

important attributes of successful secure coding training. Given these research gaps, we wished to assess our respondents' experience of secure coding training. We therefore analyzed both the content and sentiment of their free-text descriptions of secure coding training.

General security training in the workplace includes familiar advice such as not clicking on suspicious e-mail links and using secure passwords. Secure coding training requirements are different. One difference is the large range of languages and tools used by developers. Survey respondents used 73 distinct programming languages and 128 other development tools. Each of these will have its own security peculiarities, about which we argue that developers should be informed. We are not suggesting that all developers should receive intensive cryptography training and become security experts. Rather, we note that developers should be able to identify and avoid common security pitfalls. For example, to avoid injection vulnerabilities, developers should carefully check text that was input directly by a user. Even something as innocent as a user-supplied postal address may contain malicious code that a user is trying to "inject" into the system. Training can provide developers with the awareness, knowledge, and tools to routinely and confidently deal with such issues in their coding language of choice.

We identified seven studies in the literature review[7] that touched on training and that made secure coding training recommendations. Parsing these recommendations, we assembled the list of desirable training attributes shown in Table 3. The numbers indicate the number of studies (from the seven) in which the attribute was either discussed or implied. To find out whether these attributes are present in the secure coding training received in industry, we analyzed the answers to the two training-related questions in our developer survey.

Some of the attributes in Table 3, such as that training should be relevant, actionable, and up to date, may seem so self-evidently desirable as to be trivial to discuss. However, survey results revealed that just over 60% of respondents had never been offered software security or privacy training of any kind, excellent or otherwise. There were 381 respondents (39.6%) who had received secure coding training, of whom 220 described it. Of these, an encouraging 91 described training that appeared to be very relevant. However, the frequency of training for 45 of them was unknown, while seven of them received annual training and three had only one-off sessions. In the professional development experience of the lead author, annual or one-off software security training is inadequate. Only 36 respondents (3.74%) described training that, as well as being relevant, was monthly or more frequent. Many more respondents may have had satisfactory training; however, our training questions were not sufficiently fine-grained to ascertain this. Relevance and frequency

were unknown for most respondents. Given the general absence of data on developer secure coding training, the need for further research in this area is indicated.

Respondents' tone varied widely when describing the training they had received (see Table 4). Non-neutral sentiment on training was more positive than negative, though the very negative comments were telling. For example: "Sh***y prerecorded video trainings with multiple choice quizzes."

Based on their free-text responses, developers value good secure coding training. This is particularly important in the current era. Developers increasingly rely on generative artificial intelligence (AI) for coding assistance, exposing them to a troubling potential for introducing poorly understood vulnerabilities.[15]

Isolated Developers May Be Under-Resourced

Our fourth paper on the survey results studied five cohorts of potentially under-resourced developers[11] (see Table 5). Secure development methods and advice such as the BSIMM tend to be targeted at large and well-resourced organizations. Many developers operate in environments with less support. They may be less informed about secure coding—and more likely to ignore it due to isolation. We evaluated to what extent the 12 CAs were performed by these cohorts, including an "all developers" cohort for context. In general, the highest incidence of most activities was reported by developers in medium-sized organizations, probably reflecting the resources available. Developers in small organizations, however, were the most likely to report an ability to respond swiftly to emergencies (see Q6 in Figure 7). This could reflect limited bureaucracy and high code prioritization in small software organizations. This and the activity in Q7, tracking operations defects, are of value even when security is not prioritized, and they were widely reported by developers working in organizations. In contrast, Q4 (Figure 7) involves the security-specific activity of verifying that functions like protecting resources from unauthorized access are tested by quality assurance (QA). The occurrence of this and other security-specific activities was under 40%, even for medium-sized organizations.

The CA score activities are based on observations of large organizations that are well resourced. There are no equivalent observations available for under-resourced developers. Therefore, the survey data fill an important gap. We expected to see relatively low numbers for CA score activities among the five under-resourced cohorts. However, we speculated that there could be widespread use of secure coding tools in these groups because the use of such tools allows isolated developers to access community and industry knowledge. The results from the survey

Table 2. Correlations between software security interest, knowledge, and enjoyment, the CA score, and security culture environment and empowerment factors in the sample.

	Interest	Knowledge	Enjoyment	CA Score
Interest	1			
Knowledge	**0.46**	1		
Enjoyment	**0.44**	**0.25**	1	
CA score	**0.2**	**0.31**	0.11	1
Environment	**0.28**	**0.36**	**0.18**	**0.63**
Empowerment	0.12	0.13	0.12	**0.25**

Only relationships with p-values <0.001 are considered statistically significant. These are in bold.

Table 3. Desirable attributes for secure coding training extracted from seven relevant papers.

Attribute	Number of Papers Discussed	Number of Papers Implied
Relevance		
Relevant	7	0
Actionable	7	0
Up to date	6	1
Iteratively improved	4	2
Error based	4	0
Include risks	4	0
Training tools	1	0
Time allocated		
Continuous	4	1
Ongoing mentoring	2	0
Include time	3	0
Mandatory	1	3
Engagement		
Engaging	2	0
Varied delivery	4	0
Interactive	2	0
Gamified	2	0
Additional training attributes		
Available expert	3	0
Peer review	2	1
Train managers	1	0
Non-tech skills	1	0

The figures on the right indicate how many of the seven papers either explicitly discussed the attribute or implied that it should be present.

Table 4. Developer tone on secure coding training.

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

Table 5. Categories of potentially isolated developers.

Cohort	*n*	Description
Lone coders in organizations	64	Codes alone (not on a team), in an organization with size >1
Coders in small organizations	234	Codes on a team within a small organization (fewer than 50 employees)
Coders in medium-sized organizations	129	Codes on a team within a medium-sized organization (between 50 and 199 employees)
Coders not working for organizations	67	Codes in a nonorganizational environment
Freelancers	23	Codes in a single-person organization
All Developers	962	All valid survey respondents

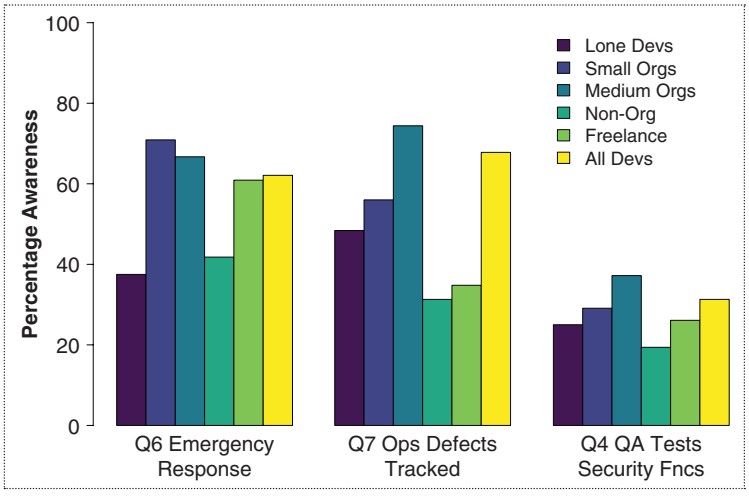


Figure 7. Results for three sample CAs for all five developer categories and the whole sample. Developers in small organizations are the most aware of emergency response capabilities in their organization, and the rate of operations defect tracking is reasonable for all cohorts. However, all cohorts' awareness of security-specific activities, such as checking security functions, is relatively low. Org: organization; Fncs: functions; Ops: operations; Devs: developers.

(see Figure 8) were disappointing. In small organizations, only 32% of developers stated that such secure coding tools were present. Even in medium-sized organizations, only 45% of developers reported them.

We advise that policymakers should consider isolated developers, open source teams, and small organizations when devising policy for software security. Software security advice and standards tailored to these categories of developers are needed. The accessibility, usability, and price of security tools and other security aids should be considered by the industry and policymakers in a holistic way. Business leaders should ensure that development teams make maximum use of the excellent tools available and should provide relevant training and expert support.

A Call for Action

Software developers are much more likely to code securely if they are explicitly asked to. Yet developers in industry suffer from poor software security practices and culture in at least half of software development organizations. Some developers express dissatisfaction with, or cynicism about, the secure coding practice and culture where they work. Even when organizations do state that they prioritize software security, they often do not supply the resources and training that developers need.

Business leaders should investigate whether attention to code security is well supported and prioritized in their organizations. They should visibly and overtly provide developers with good secure coding support. This includes supplying training that is timely, uses a range of media, and is tailored to the languages and tools in use within the organization. Large or specialist organizations often develop training in house. For smaller organizations, interactive, relevant, and engaging tools are available commercially. Mentoring and communication are also crucial for code security and should be supported. This is particularly important given the potential for over-reliance on the security of AI coding tools. Developers must be equipped to detect when such tools generate insecure code.

Policymakers also have a role to play. There have been improvements in attention to under-resourced organizations, with recent European Union (EU) legislation encouraging awareness raising and training activities for "microenterprises" at the national level,[d] including provision of access to security assessments and sandboxes. Such initiatives will result in an all-around improvement in software security given the tight coupling in the software ecosystem.

Academia can support a drive to more secure coding by undertaking further research on how to provide useful software security training and support to microenterprises, open source teams, and other under-resourced developers. Academia can also play a role in improving software security education.

[d]EU Cyber Resilience Act, 2024.

Business, policy-makers and academia all have a role to play in advancing software security. Above all, we encourage decision-makers in organizations to make software security a genuine priority. They should communicate its importance clearly and regularly to developers throughout the organization. They should give them the training and support they need to understand software security and to integrate it into their daily development practice. ■

Acknowledgment

This publication has emanated from research conducted with the financial support of Research Ireland under Grants 18/CRT/6222, 13/RC/2077_P2, and 13/RC/2094_P2. For the purpose of Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission. This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the University College Cork (UCC) Social Research Ethics Committee, and performed in line with the UCC Code of Research Conduct.

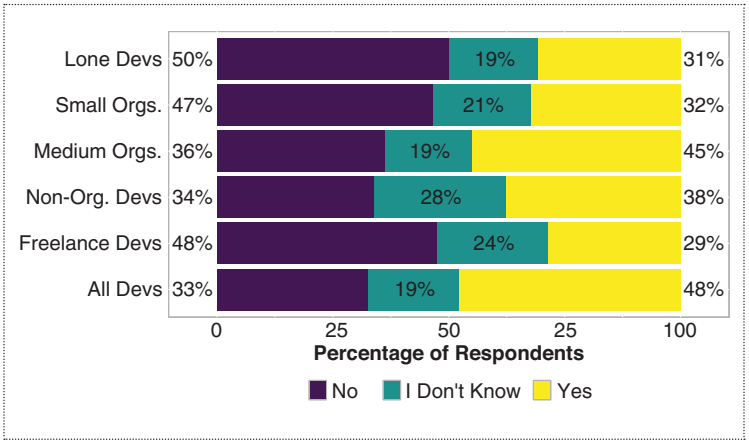


Figure 8. Are there security tools available in your environment? Only 29% of freelance developers use security tools. The numbers are low for most potentially under-resourced developer cohorts.

References

1. Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "You get where you're looking for: The impact of information sources on code security," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Piscataway, NJ, USA: IEEE Press, 2016, pp. 289–305, doi: 10.1109/SP.2016.25.
2. 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 *Proc. IEEE Cybersecurity Develop. (SecDev)*, 2016, pp. 3–8, doi: 10.1109/SecDev.2016.013.
3. 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 *Proc. CHI Conf. Human Factors Comput. Syst.*, New York, NY, USA: ACM, 2020, pp. 1–13.
4. 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 *Proc. CHI Conf. Human Factors Comput. Syst.*, New York, NY, USA: ACM, 2019, pp. 1–12.
5. H. Assal and S. Chiasson, "Think secure from the beginning": A survey with software developers," in *Proc. CHI Conf. Human Factors Comput. Syst.*, New York, NY, USA: ACM, 2019, pp. 1–13.
6. I. Ryan, U. Roedig, and K.-J. Stol, "Weak programmers need not apply, LLMs welcome! Survey screening in the AI era," in *Proc. IEEE/ACM 48th Int. Conf. Softw. Eng. (ICSE)*, New York, NY, USA: ACM, 2026. [Online]. Available: https://cora.ucc.ie/items/070de25e-66ae-4b6a-94b2-d675e6c511d2
7. I. Ryan, U. Roedig, and K.-J. Stol, "Unhelpful assumptions in software security research," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA: ACM, 2023, pp. 3460–3474.
8. 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 *Proc. IEEE/ACM 45th Int. Conf. Softw. Eng. (ICSE)*, 2023, pp. 1622–1634, doi: 10.1109/ICSE48619.2023.00140.
9. I. Ryan, U. Roedig, and K.-J. Stol, ""ImmediateShortTerm3MthsAfterThatLOL": Developer secure-coding sentiment, practice and culture in organisations," in *Proc. IEEE/ACM 47th Int. Conf. Softw. Eng. – Softw. Eng. Pract. (ICSE-SEIP)*, 2025, pp. 551–562, doi: 10.1109/ICSE-SEIP66354.2025.00054.
10. I. Ryan, U. Roedig, and K.-J. Stol, "Training developers to code securely: Theory and practice," in *Proc. ACM/IEEE 4th Int. Workshop Eng. Cybersecurity Crit. Syst. (EnCyCriS) IEEE/ACM 2nd Int. Workshop Softw. Vulnerability*, 2024, pp. 37–44.
11. I. Ryan, K.-J. Stol, and U. Roedig, "The state of secure coding practice: Small organisations and 'lone, rogue coders'," in *Proc. IEEE/ACM 4th Int. Workshop Eng. Cybersecurity Crit. Syst. (EnCyCriS)*, 2023, pp. 37–44, doi: 10.1109/EnCyCriS59249.2023.00010.
12. C. Weir, S. Migues, M. Ware, and L. Williams, "Infiltrating security into development: Exploring the world's largest software security study," in *Proc. 29th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2021, pp. 1326–1336.
13. 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 *Proc. 14th Symp. Usable Privacy Secur.*, USENIX Association, 2018, pp. 357–373.

14. T. Lopez, H. Sharp, T. Tun, A. Bandara, M. Levine, and B. Nuseibeh, "Security responses in software development," *ACM Trans. Software Eng. Method.*, vol. 32, no. 3, pp. 1–29, 2022, doi: 10.1145/3563211.
15. A. Kudriavtseva, N. A. Hotak, and O. Gadyatskaya, "My code is less secure with Gen AI: Surveying developers' perceptions of the impact of code generation tools on security," in *Proc. 40th ACM/SIGAPP Symp. Appl. Comput.*, New York, NY, USA: ACM, 2025, pp. 1637–1646.

Ita Ryan is a senior postdoctoral researcher at University College Cork, T12 K8AF Cork, Ireland, and has extensive software development experience. Her research interests include software security, cybersecurity, and security for the Internet of Things. Ryan received her Ph.D. in software security from University College Cork in 2024. Contact her at iryan@ucc.ie.

Utz Roedig joined the School of Computer Science and Information Technology (CSIT), University College Cork (UCC), T12 K8AF Cork, Ireland, in January 2019. His research interests include computer networks and security with a focus on the Internet of Things. Roedig received his Dr.-Ing. in computer science from the Darmstadt University of Technology, Darmstadt, Germany, in 2002. He has published more than 160 peer-reviewed articles in this field, and his research collaborations with industry partners have resulted in several patents. Contact him at u.roedig@ucc.ie.

Klaas-Jan Stol is a professor of software engineering with the School of Computer Science and IT at University College Cork, T12 K8AF Cork, Ireland. His research interests include software development practice, including social processes such as onboarding and collaboration. Stol received a Ph.D. from the University of Limerick. Contact him at k.stol@ucc.ie.