CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	'Follow the money. Or jail time.' Perspectives on the role of senior management in secure software development
Authors	Ryan, Ita;Roedig, Utz;Stol, Klaas-Jan
Publication date	2026-08-14
Original Citation	Ryan, I, Roedig, U & Stol, K-J 2026, ''Follow the money. Or jail time.' Perspectives on the role of senior management in secure software development', ACM Transactions on Software Engineering and Methodology. https://doi.org/10.1145/3840288
Type of publication	Article (peer-reviewed)
Link to publisher's version	10.1145/3840288
Rights	© 2026, the Author(s). Publication rights licensed to ACM. For the purpose of Open Access, the authors have applied a CCBY public copyright licence to any Author Accepted Manuscript version arising from this submission. - https://creativecommons.org/licenses/by/4.0/
Download date	2026-09-23 20:32:44
Item downloaded from	https://hdl.handle.net/10468/19327

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

‘Follow the Money. Or Jail Time.’ Perspectives on the Role of Senior Management in Secure Software Development

ITA RYAN, UTZ ROEDIG, and KLAAS-JAN STOL, University College Cork, Ireland

The number of recorded vulnerabilities in software continues to rise, despite attempts to improve secure coding practice. Much software security research focuses on software developers’ abilities, tools and motivations. However, software security budgets and priorities ultimately derive from an organisation’s senior management. There is little research on senior management’s understanding of software security. We interviewed 21 professional developers, managers and consultants to assess perceptions of how management priorities shape software security in organisations. We found that management priorities are important in providing the resources needed for secure software development. Software-security understanding can be lacking in senior management, which, along with budget pressures, can affect its prioritisation. Developers can be self-motivated to code securely despite lack of management support. Disturbingly, some senior managers will actively subvert software security. Acquisitions, divestitures, downsizing, and funding issues may affect an organisation’s software security stance. We make a number of recommendations for policymakers, the C-suite and other stakeholders, and suggest directions for further research.

CCS Concepts: • **Software and its engineering → Software organization and properties**; • **Security and privacy → Human and societal aspects of security and privacy**.

Additional Key Words and Phrases: Secure software development, executive management, interview study

ACM Reference Format:
Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2026. ‘Follow the Money. Or Jail Time.’ Perspectives on the Role of Senior Management in Secure Software Development. *ACM Trans. Softw. Eng. Methodol.* 1, 1 (August 2026), 38 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Cybercrime and the use of cyberattacks in geopolitics are increasing [103]. Many hacking techniques depend on vulnerabilities in software [28, 49, 76]. Recorded vulnerabilities are proliferating. This is partly a consequence of rapid digital transformation, the professionalisation of cybercrime [11], and the use of AI for vulnerability discovery [35], but is also due to poor secure development. Research shows that many software developers and their workplaces pay insufficient attention to software security [9, 10, 68].

There is no dedicated literature in the area of executive decision-making on software security. We have observed that the terms **cybersecurity** and **software security** are frequently confused. Organisational decision-makers are largely aware of internal **cybersecurity** issues such as multi-factor authentication implementation, due to legislative and compliance consequences for data breaches. Poor **software security**, in contrast, often has no internal implications. Adverse consequences to the release of insecure software, such as customers being hacked and ransomwared, are external. The organisation, if not itself running the software it has released, does not suffer a

Authors’ Contact Information: Ita Ryan, iryan@ucc.ie; Utz Roedig, u.roedig@ucc.ie; Klaas-Jan Stol, k.stol@ucc.ie, University College Cork, Cork, Cork, Ireland.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1557-7392/2026/8-ART
https://doi.org/XXXXXXX.XXXXXXX

breach. In the United States (US), software licenses and the Uniform Commercial Code [22] protect organisations from their software's potentially disastrous effects on customers. Until the advent of the European Union (EU) Cyber Resilience Act (CRA) (2024) [98], there was little related consumer protection legislation. Therefore, models of managerial cybersecurity behaviour, which are partly predicated on legislative constraints, do not directly apply to the software security domain.

Good software security practices comprise activities throughout the software lifecycle. In this study we focus most on the subset of software security comprising the software **development** lifecycle (SDLC), and the process of **secure software development**, or **secure coding**. Security later in the lifecycle, when the software has been deployed to a production environment, comprises activities like patching, protection and runtime monitoring, and is often called **application security**.

Without leadership support, it is difficult to establish a trustworthy, ongoing secure software development regime in an organisation [24]. Naiakshina et al. conducted a series of laboratory studies on whether developers consider security when writing code to store passwords [61–63]. They repeatedly found that a very large majority of developers will not store passwords securely unless they are asked to. The security implications of password storage should be self evident. It is reasonable to extrapolate this finding to other secure-coding tasks. Furthermore, management support is needed to introduce software security interventions, like those that teach developers basic security practices, to organisations [50, 53, 101]. Thus the priorities that senior management, sometimes called the C-suite, set affect secure development. Most organisations already have multiple conflicting objectives and priorities, such as tight budgets and deadlines [68, 96] and the desire to limit staff numbers [93]. We were unable to find research specifically focused on senior management software-security prioritisation in organisations across industry. In this paper, we report on 21 interviews with professionals who either work in organisations where software is developed, or consult for a range of such organisations. Most of the informants in this study are working software developers. They must deal at the coal-face with budgeting and prioritisation decisions made at the top of the management hierarchy. Five participants have wide experience of senior management. We investigate the perceived influence of leadership priorities on software security practice. We discuss participants' opinions of whether CEOs and C-suite executives understand software security, and what motivates them to prioritise software development. Finally, we investigate factors that tend to cause a shift in software security emphasis in organisations. Our research questions (RQs) are as follows:

RQ1: What is senior management's understanding of, and attitude to, software security?
RQ2: How do management priorities affect software security within organisations?
RQ3: What factors motivate senior managers to prioritise software security in organisations?
RQ4: What factors might change an organisation's software security prioritisation?

Section 2 discusses background and related work in relation to these research questions, followed by a description of the research procedures and limitations in Section 3. We present the findings of this study in Section 4. In Section 5 we discuss the findings. We conclude in Section 6.

2 Background and Related Work

There is extensive research into software security using a developer lens. Researchers have asked what secure-coding abilities developers have [4, 5, 60–63, 100], how their tools should be designed [1, 2, 37, 38, 89, 104], how security-aware their information sources are [3, 29], what interventions would help them code more securely [74, 101], and what their secure-coding motivations are [10]. An important aspect of developers' secure-coding experience is the environment in which they work [10, 20, 58, 70, 75]. Lopez et al. noted that '*the security positions of organisations may have more weight in promoting security activity than does championing by individuals*' [54]. Assal and Chiasson

identified '*Competing priorities & no plan*' as a software security deterrent [10]. Such findings suggest that prioritisation of secure coding by senior management is important. As Espinha Gasiba et al. found when studying developer awareness of secure coding guidelines, '*without management understanding and approval, it is not possible to establish secure coding practices in a company*' [25].

2.1 Software security awareness

Research on *cyber*security discusses questions like the optimum amount of cybersecurity investment by organisations [36], and how policymakers can adjust investment levels [27]. Such advice presupposes a managerial awareness of the issues that, as discussed, is not always found in *software* security literature. Legislative requirements for secure software were almost non-existent until recently [22]. Do senior managers have the awareness and interest needed for software security to thrive? Certainly not always. In a study by Assal and Chiasson on security in the SDLC, one participant reported being told by their manager that security was not a focus [9]. Lopez et al. found in ethnographic work that developers had difficulty convincing senior management to undertake '*bigger security initiatives*' [54]. Some ethnographic studies [68] and descriptions of interventions [101] reveal poor software security direction from senior management.

This lack of input could be due to ignorance. We did not find research on senior management software security awareness and competence. Thus, we ask RQ1: what is senior management's understanding of and attitude to software security?

2.2 Senior management's prioritisation

Organisational priorities are set by senior management. Research on prioritisation of software security in an organisation must therefore explore whether it is a priority for senior management. Hu et al. established the influence of senior management on *information* security initiatives. Their participation increases legitimacy, commitment (including resource allocation), cultural acceptance and employee buy-in [43]. Similarly, in *software* security, once senior management backs a security drive '*it's just a standard change management practice*' [50]. In an ethnographic study of how an organisation developed a software security culture, Tuladhar et al. concluded that a key security enabler was senior management '*setting security as a goal*' [94].

An example of the resources under management's control is time. In a laboratory study involving coding tasks, one in six developers commented that the tight time limit made it difficult to consider security [3]. This issue is reflected in the work environment [52, 96]. Morales et al. described difficulties in a large multi-year, multiple-subcontractor 'DevSecOps' project [58] where the featured subcontractor did no security testing. They argued that leaders who do not emphasise software security communicate that it is not important; a point also made by Arizon-Peretz et al. [7, 8]. Poller et al. observed a development team's activities after the team had addressed penetration test issues [72]. They wanted to see if the team's secure-coding learnings were reflected in practice in the medium term. They found no improvement, and concluded that this was because management did not ask for or prioritise security.

Numerous studies describe techniques for organisations to enhance software security. Strategies include improving employees' understanding of security goals [7], designing secure-coding interventions tailored to specific developers [74] and contexts [50], and running regular meetings to surface secure-coding issues [95]. We argue that such interventions probably will not happen in organisations where secure coding is currently ignored. Senior decision-makers' interest is needed to make budget and resources available. Hence, we ask RQ2: How do management priorities affect software security within organisations?

2.3 Secure development motivations

What would motivate organisations that currently ignore code security to make it a priority? Assal and Chiasson advocated research on *'why some teams completely ignore software security, and what factors could encourage them to adopt a security initiative'* [9]. Oyetoyan et al. noted that *'We need to further investigate the drivers for increase in software security adoption in an organization,'* and speculated on what they might be; perhaps management security commitments [66]? In our own previous research, we found that researchers sometimes assume that organisations and managers prioritise software security [78]. They do not always check this assumption.

Researchers have noted a number of factors. Weir et al. observed that '*compliance checks, certification and recent relevant legislation have all caused an increased interest in security in the organization*' [101]. Laws, regulations and industry standards can affect the incorporation of privacy and security in software [64], as can a strong secure development culture, reaching all the way from individual developers to top executives [40]. External security drivers include '*gaining a larger market share or customer requirements*' [40]. To our surprise, we did not identify a study focused specifically on software security drivers and motivations in organisations. Thus, we ask RQ3: what factors motivate senior managers to prioritise software security in organisations?

2.4 Sudden changes in security prioritisation

Two pieces of research suggest that an organisation's secure coding posture can be affected by turbulence such as downsizing or acquisition. Weir et al. studied the effect of lightweight secure coding interventions [101]. One team lost three participants to redundancy during the 3-month intervention period. A year later, only one of those initially interviewed was still with the company. It is difficult for institutional memory to survive such comprehensive disruption. Lopez et al. undertook a year-long ethnographic study of secure development in two organisations [54]. Both were recently acquired and had a changed security focus, yielding an interesting contrast. In one, developers were proactive and newly enthusiastic about secure coding. In the other, developers had lost their previous software security confidence. They had entered a compliance nightmare, and felt less positive that their software was secure despite having to jump through multiple security hoops. It is possible that organisational turbulence may have a serious adverse impact on secure software development. Thus, we ask RQ4: what factors might change an organisation's software security prioritisation?

3 Research Design

We considered several ways of investigating the research questions and concluded that use of semi-structured interviews would best suit our goal. Qualitative techniques are appropriate for exploratory research where an existing theoretical framework has not been identified. We analysed the interviews using reflexive thematic analysis (RTA) [12].

3.1 Interview participants

We interviewed 21 professionals working in organisations where software was developed. Table 1 lists the study participants, with the identifiers used for them in this work. The participants were categorised into five different job roles, based on interview data. **Development team members**' view is practical and based on the reality of development in an organisation. While developers are usually not privy to discussions at board level, they have a keen awareness of the time, tools and training available for software security at the coalface. **Senior software developers** have more experience and have often worked in multiple organisations. They have an understanding of resources available to development teams in the organisation, and of the direction received from

Table 1. Interview participants, listed by role. Due to anonymisation requirements, numbers for demographics are aggregated in separate tables.

Development Team Members	
D1	Tech journalist, ex-developer
D2	Systems engineer intern
D3	Software engineer
D4	Software engineer
D5	Software engineer
D6	Software engineer
D7	Software engineer
D8	Open source developer
Senior Software Developers	
SD9	Senior software engineer
SD10	Senior business analyst/engineer
SD11	Senior software engineer
SD12	Senior software engineer
SD13	Senior software architect
SD14	Lead software engineer
Consultant Developers	
CD15	Director
CD16	Consultant developer
Senior Managers	
SM17	Global M&A lead
SM18	Cybersecurity IT Lead
SM19	Vice President
Management Consultants	
MC20	Management consultant
MC21	Management consultant

senior management. **Consultant developers** are independent developers who work with many different organisations and have profitable apps of their own. They are senior developers, but we classify them separately because their independent role gives them a good grasp of industry-wide challenges. **Senior managers** are aware of the main areas of concern and risk in the organisations they work with. They know what topics are under discussion at board level, and also what topics are not on the radar. Two of those we interviewed had experience in evaluating organisations for acquisition, giving them excellent insight into secure software development in multiple environments. **Management consultants** have a broad view of the industry, and they too have dealt with many different organisations. They are accustomed to quickly sizing them up and evaluating issues of concern. Identifiers in Table 1 are prefixed with role abbreviations, to assist the reader with context throughout the paper.

Three participants worked at different levels in the same organisation. We carefully reviewed their anonymisation requirements, and determined that we could not allocate country, organisation size or industry in Table 1. These details are aggregated in Table 2.

To aid understanding of their coding environments, we included anonymised technical details for developer participants in Table 3. These reflect current roles and technical skills. As can be observed, developers from a range of industries using multiple different tech stacks participated in

Table 2. Participant demographics. An asterisk* in the industry column indicates a range of industries.

Country	Participants
US	10
Ireland	6
U.K.	2
Belgium	1
Germany	1
India	1
Organisation Size	**Participants**
Solo Consultant	3
Small (10-49)	6
Medium (50-249)	2
Large (250-19,999)	4
Very large (20,000)	4
NA	2
Industry Domain	**Participants**
Aerospace	2
Consumer Software*	2
Education	1
Energy	1
Finance	4
Health	3
Professional Services*	4
Security software	3
Telecoms	1

Table 3. Anonymised technical details for participating developers. These reflect current roles and technologies used.

Product	Technology	Users
Backend insurance software	AWS, C#, Go	Internal
Backend transport utility	Java Spring Boot, REST	Internal
Banking applications backend	.NET, C#	Businesses
Computer security product	Ruby on Rails	Consumers
Electric utility software	.NET, C#	Internal
Internal consultancy tool	.NET core with microservices	Internal
IoT and computer security tool	C++, Python, Rust	Consumers
Low-level mobile device development	Swift	Consumers
Mac and iOS utilities	Swift	Consumers
Crypto privacy-preserving telemetry aggregation	Google Cloud, Terraform, rust	External
Satellite data processing	Python	Internal
Security analysis software	C++, Python	Consumers
Telecoms applications	Java, Golang, Elm	Businesses
Web applications	C#, Javascript	Consumers

the study. Many participants have also worked in multiple other roles across different industries. Participants were asked to consider any relevant prior experience during the interviews, as well as their current roles. Therefore Table 3 is not a comprehensive list of the environments discussed.

3.2 Interview recruitment

Based on prior literature (see Section 2.1), we were aware that not all senior managers prioritise software security. Interviewing only senior managers could fail to uncover evidence of management bad practice. Interviewees could potentially suffer from social desirability bias, and might be inclined to exaggerate how highly they prioritise secure development. Therefore we wished to interview participants from across the organisational hierarchy. We had two sources of recruitment: contacts from a previous survey study and our own professional contacts. Data saturation is not considered a useful concept in RTA, with the emphasis instead on having '*adequate data to tell a rich, complex and multi-faceted story about patternings related to the phenomena of interest*' [14]. We approached recruitment with this in mind.

3.2.1 Recruitment from previous study. While conducting a previous large-scale survey [81], with participants recruited online, we solicited contact details from software professionals interested in being interviewed at a later date. Most of the volunteers from this previous study had no relationship with the authors. Potential bias caused by undertaking the previous survey is discussed in Section 3.7. Thirteen participants were sourced from this pool of professionals.

3.2.2 Recruitment from personal contacts. The remaining eight interviewees were our own professional contacts from different arenas. They were selected for their specific software development-related experience and expertise, with many working in very senior decision-making roles, and increased the diversity of viewpoints. All contacts asked to participate did so. They seemed to enjoy the interviews and expressed an interest in the subject and results of the study. All expressed that the issues discussed were relevant and worth raising.

3.2.3 Additional recruitment details. Compensation for participation was not offered. The appointed interview duration was 30-60 minutes. Interview length varied depending on the interviewee's chattiness, experience, and in-depth knowledge; average interview duration was 36 minutes. Ethics considerations are discussed in Section 3.5. Appendix A contains the interview questions.

3.3 Interview analysis

Interviews were conducted by the lead author and were semi-structured. We devised a list of questions designed to explore issues of software security and decision-making within organisations. We retained the flexibility to ask additional questions when the interviewee introduced interesting topics. The lead author also analysed the interviews, using RTA with careful note-taking. This form of analysis was chosen because it is a flexible technique designed to explore and generate themes in qualitative datasets [18]. Coding in RTA is acknowledged to be subjective. Indeed, the explicit acknowledgement that the researcher is '*situated in various ways, and [...] reads data through the lenses of their particular social, cultural, historical, disciplinary, political and ideological positionings*' is considered one of RTA's strengths [13]. RTA is therefore best implemented by a single researcher who is thoroughly familiar with the entire dataset.

Byrne's worked example of Braun and Clarke's approach, which incorporates developments in Braun and Clarke's thinking subsequent to their canonical paper [12], was used as a guide during analysis [16]. NVivo was used for coding, with additional work in a word processor and spreadsheet. We used a combination of deductive and inductive coding of interview transcripts to generate themes from the interviews. Phase one involved transcribing and rereading the interviews, noting and considering interesting points made by the subjects, and becoming familiar with the data. During phase two, we conducted two stages. The first involved reading through each transcript again, assigning descriptive codes to blocks of text to categorise them and group related subjects together. In the second stage, the descriptive codes were re-examined and inductive coding was

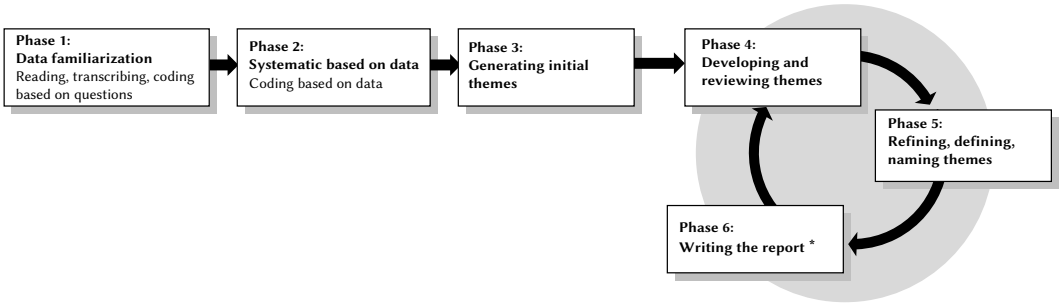

Fig. 1. Our reflexive thematic analysis (RTA) workflow. The final three phrases took place iteratively, concluding with the last iteration of phase 6, symbolised by an asterisk.

used to identify ideas and common experiences within the data. This involved recoding multiple items, eliminating single-item codes, and beginning to think in terms of groups of codes. These codes were generated from analysis of the data; most could not have been predicted from the list of questions. In the third phase the codes were revisited and pondered to generate themes. During these phases, and throughout the work, notes of the process were kept in NVivo, Trello, and an ongoing journal.

As is normal in RTA, we constantly revisited and considered the original data in an iterative way. Beginning the fourth phase, to avoid getting bogged down in process and encourage new thinking we adopted a completely fresh approach. We made physical copies of all codes and grouped them by hand, continually revisiting the original data, and named the resulting themes. We then compared these new themes with those generated during the original analysis. Assessing and resolving the differences between the two deepened our understanding of the data. Later in the fourth phase we reviewed the themes against the original data and against each other to assess whether they were coherent, high quality and informative. Some codes were revisited and sub-themes were moved, merged or removed.

During the fifth phase the thematic framework was analysed. Themes and sub-themes were written up. Completing the final report is the sixth formal phase of RTA. However, Braun and Clarke note that the phases they suggest are not prescriptive but are provided primarily to help researchers to find their way [13]. In practice we found that aspects of phases four (reviewing themes against original data and each other), five (thematic framework) and six (writing the report) took place iteratively, as can be seen in Figure 1. The last sixth-phase iteration, symbolised by an asterisk in Figure 1, concluded the work.

3.4 Positionality

As is common in RTA, we briefly make a statement on our positionality. The lead investigator (IR) has a bachelor's degree in philosophy, a master's degree in applied computing, a doctorate in software security, and extensive computing experience. She worked as a professional software developer and consultant for over two decades, across multiple industries and using numerous languages and development methods. She has been a technical lead for multiple software projects in several different industries. She has done courses in project planning and management, and has read extensively about software engineering management. The second author (UR) is a professor of computer science with a long track record on security. The last author (KS) is professor of software engineering, with a particular focus on social processes in software engineering and empirical methods.

3.5 Ethics Considerations

The interviews involved research on human subjects. Therefore, we obtained approval from our institution's Institutional Review Board (IRB).

Participants signed a consent form advising them that doing the interview was entirely voluntary and that they could withdraw at any time before, during or after their interview. Interview subjects were given an interview information sheet which discussed what would happen to their interview data. This included the following sentence *'All of the information you provide will be kept confidential and anonymous, and will be available only to the three-person research team.'* We are therefore not in a position to make interview transcripts available as a research artefact.

All interviews were transcribed and anonymised. Subsequent to transcription, audio records were destroyed. We were entrusted with some particularly sensitive information; when it was used in this paper, we omitted the relevant participant ID as an extra precaution.

Many of the precautions advised by the IRB related to ensuring that vulnerable interview participants did not experience harm from the interview process. For example, we were advised to consult participants on their feelings about the interview before ending the interview. We followed these precautions, although we did not consider software developers, consultants and senior executives to be vulnerable participants. However, in two interviews (one with a software developer, one with a top-level executive) we realised early on that the participant, while enthusiastic, was wholly ignorant about software security. When this occurred we omitted interview sections containing numerous detailed questions. We were concerned that the questions would embarrass the participant by repeatedly (and pointlessly) drawing attention to their lack of familiarity with software security practice. This was an unexpected dilemma which we felt we handled ethically.

3.6 Threats to Validity

Braun and Clarke provided twenty questions that can be used to assess the quality of thematic analysis, with a particular emphasis on RTA [13]. Topics explored by the questions include whether the choice and explanation of methods and methodology are adequate, and whether analysis is well developed and justified. In our analysis, we attempted to adhere to the principles and practice of RTA.

When assessing the validity of qualitative research such as this, quantitative evaluation metrics such as construct, internal and external validity are not considered applicable [19]. Instead, assessment focuses on trustworthiness [6, 86], using the lens applicable to the work under consideration [19]. In assessing threats to the validity of this study, we considered the four qualitative research validation criteria identified by Lincoln and Guba [51]: credibility, transferability, dependability and confirmability.

3.6.1 Credibility. Credibility concerns the degree to which the findings in the work ring true and are convincing [51]. In order to establish credibility we used rich description of our findings, directly quoting the opinions and experiences of our interviewees where we felt such quotes added value. Many of the interviewees had clearly engaged thoughtfully with secure coding issues in their work environment and in the general environment. Their opinions were considered, refined, and often colourfully expressed. By choosing to allow interviewees' voices to speak throughout the text, we sought to enhance the credibility of our analysis and provide an accessible audit trail of how our conclusions were reached.

Triangulation, providing alternative sources that confirm findings, is widely used as evidence of quality in qualitative research. RTA does not foreground the concept of objectivity. Its emphasis is on engaging with material honestly while acknowledging that all researchers bring their own biases. Indeed, Braun and Clarke suggest that objectives such as coding consensus may be reached

at the expense of the granularity and richness of findings, causing, for example, codebooks to be created even before the voices of interviewees are heard [14]. Nevertheless, analysis does not take place in a vacuum. An extensive review of the literature informed our initial research focus and the questions asked in the interviews [78]. Section 2 discusses some of this work. Sources that resonate with our findings are described and referenced, providing a form of triangulation.

3.6.2 Transferability. Rather than discussing '*external validity*,' a criterion commonly discussed in quantitative studies including survey studies and laboratory experiments, Lincoln and Guba use the term '*transferability*.' Or, to which extent are this study's findings applicable in other settings?

Qualitative research findings are not statistically generalisable in the same way as findings in quantitative studies [39]; nor is that the goal of qualitative inquiries such as this study [87]. Instead, qualitative studies are better suited to develop better understanding, and to achieve theoretical generalisability. In this study, we seek to shed light on hitherto unexplored questions to develop a better understanding and inform future research. Studies of this type may '*generalise through transferability*' if readers can envisage applying the findings of the research in their own environments, or using them to inform practice [85]. Relatedly, the findings may also have *'naturalistic generalisability'* [85]. That is, they may resonate with readers with relevant experience. What Creswell and Miller [19] called '*thick description*' can help readers achieve a feeling of verisimilitude, i.e., *'the feeling that they have experienced, or could experience, the events being described in a study.'*

To establish whether this study did, in fact, create verisimilitude, we obtained informal reviews of our results from three experts familiar with the software development process. We asked them two questions: '*Does this paper come across as believable?*' and '*Do the findings resonate with you?*' Expert A is a former developer, compliance consultant and revenue officer who has just retired from work in Italy. They stated:

> 'What you have found and reported in this article fully resonates with my experience, direct or indirect. I am aware of recent incidents whose root cause was unsecure coding, and it came from a poor culture in management, for all the reasons and with all the nuances you well describe.'

Expert B is a Senior Security Engineer with over 15 years of experience working for a Big Tech company in Ireland. They commented '*Yes, the responses are believable ... several findings resonate strongly ... Financial and legal consequences are, in my view, the most effective way to motivate C-suite attention to software security.*' Expert C, a financial organisation vice president with several decades of software development experience, currently working in the US, said:

> 'Yes this is very believable and very much resonates with my experience as a developer. There are often senior developers on the front lines that think about, and get concerned about, security issues. Management often have either a much more theoretical approach, or much less serious approach.'

These responses serve to establish a likelihood that the study has naturalistic generalisability.

3.6.3 Dependability. Rather than discussing reliability, as would be typical for quantitative studies, Lincoln and Guba suggested the term '*dependability*,' or, the extent to which findings are reliable and the extent to which variation in findings can be understood. Dependability in RTA can be considered in terms of whether the researchers clearly outlined their ontological and epistemological position, and carefully described their coding and interpretative process. In Section 3.4 we outlined our positionality. In Section 3.3 we described how we analysed the data. By providing details of the analytic process and the thoroughness with which it was conducted, we allow readers to assess the dependability of the process.

3.6.4 Confirmability. Confirmability is associated with generalisability and neutrality. Two concerns in particular should be considered: the researcher's bias (including the fact that a single investigator conducted the interviews and led the analysis), and participant selection. In RTA, the researcher's bias and influence on study findings are openly acknowledged and embraced. Note that biases remain present in other research, in, for example, the choice of subject and the wording of questions. The difference is that RTA recognises that such biases are inevitable, but may be an asset to the study. We may wish to demonstrate that the study's findings are a coherent reflection of informants' contributions. We do this by allowing the participants' voices to speak throughout the text; the '*thick descriptions*' mentioned earlier play a role here [19].

3.7 Limitations

Thirteen participants were recruited from respondents to a previous anonymous online survey [81]. Participating in this survey could have biased participants. Survey completion did not involve direct interaction with the researchers, except for about 5% of respondents who were asked to complete and publicise it. The survey was targeted at all software developers and promoted on social media. It did not require any software security expertise. Participants undertook the survey anonymously. If they were prepared to volunteer for a later interview study they were directed to a separate one-question survey where their contact details were recorded. For most respondents, their first direct interaction with the researchers came when they were contacted over a year later. At this time they were reminded about the survey and asked if they were still prepared to participate in the interview study. Four participants explicitly mentioned the survey during their interview, remarking that they did not remember their survey responses. Several others observed, when accepting the interview invitation, that they did not remember the survey.

Interview study participants can suffer from social desirability bias. In this study, this could have caused them to exaggerate their software security practice. To mitigate this, the recruitment materials emphasised that no software security expertise was necessary. Additionally, the interview questions were detailed. It quickly became apparent when participants did not have experience in particular areas. In practice, we did find some mismatches between how securely participants believed they coded and the precautions they adopted, which are part of the findings, and are discussed in Section 4.3.2.

Some participants discussed their own lack of software security practice, and even their software-security malpractice (see Section 4.1.2), suggesting that social desirability bias was effectively mitigated, for those participants at minimum.

Participants might not be comfortable speaking negatively about their employers in a software-security context. We attempted to mitigate this danger by emphasising the participant anonymity maintained throughout the process. The fact that many interviewees did not hold back when discussing employer shortcomings suggests that this bias was successfully mitigated. No participants withdrew from the study after committing to do an interview. Some participants might have limited understanding of senior management's decision-making processes and all the considerations involved, which could cause them to misinterpret management decisions.

Most of the interview participants were from English-speaking or European countries, with nearly half from the United States. Participants from other countries could have different insights. Investigating this topic would be a useful subject of future research.

Only five of our participants were from software organisations where software development is a primary focus. Others were from industries where software may be bought in or consumed, rather than being developed in-house. This may have biased the findings; environments where development is a more peripheral concern may not be as security focused as those in software organisations. However, our participants all worked in environments where at least some software

was produced, and secure development was relevant to their organisations. It is difficult to predict how findings might differ if the composition of the sample was different. Intuition suggests that we might not have discovered any instances of management subversion in very software-focused organisations. On the other hand, the literature has reported clear instances of managerial negligence in such environments in ethnographic research [68, 78]. Similarly, when outsourcing we might expect contractual security guarantees for software, but research by Morales et al. suggests that these do not always operate as expected [58].

We did not break down analysis of developers' experience by their technical role, for example front-end, back-end or DevOps developer. We did not observe differences in relevant security experiences based on such roles. These roles are not always clearly distinct; some of our participants were full-stack developers or had had different development roles in previous positions. We did observe that the two consultant developers in our study had a relatively high level of software security awareness and self-reliance, but they were nevertheless dependent on senior management for organisational software security budgets and compliance.

Multiple interviewers are often recommended for studies like this one, to reduce interpretation bias. However, in the presence of multiple interviewers participants might not have been as forthcoming. We attribute their frank responses to the lead author's extensive personal experience of the difficulties of secure coding, which facilitated an atmosphere of empathy and trust in the interviews. The originators of RTA, Braun and Clarke, describe it as an approach that embraces '*the subjective skills the researcher brings to the process*' [13]. The specific positionality of the researcher is an essential component of the research, necessarily influencing everything from choice of topic to interpretation of results. Other investigators would approach the research from a different background and with different expertise, and their results would therefore differ in ways that cannot be predicted.

4 Results

Throughout the interviews, at all participant levels, we found mentions of conflict over software security. For the interviewees, secure coding is in constant tension with other goals. Time is always in short supply. The priorities set by management dictate what it will be spent on. In the next section we provide contextualising information from interview analysis. The sections after that explore the answers to the research questions.

4.1 Contextualising information

4.1.1 Software development is fast-moving and security best practice changes. MC20 had a pessimistic view of software quality:

> 'I think the key thing is we just keep changing the tools. Every couple of years we say, *"Oh, this is going to be different. This is gonna be better*". Jeez I'm maybe working 30 years now. And it's just the same stuff. It's just rebadged and it's slightly different, but the concepts are exactly the same. The tool sets are slightly different and we end up in a situation where we continuously write bad software with new tools.'

Many other participants felt that software security was improving. For example, the Rust programming language is gradually replacing older, issue-prone languages like C and C++. Nevertheless, software development is a fast-moving profession. As SD14 said, *'best practice changes every six months. You're never going to stay on top of it if you're just a general person, so you need the experts there to dig in.'* One reason for the rapid pace of change is that many of developers' most common tools now reside in the cloud, where software providers make constant configuration changes. Recent research indicates that authentication, configuration, documentation and other issues with

cloud services are widespread [42]. While on-premise systems are inaccessible unless a port is opened, on-cloud systems have some insecure defaults, and may be exploited unless they are carefully secured [71]. A couple of interviewees felt that the cloud infrastructure set up by their organisations was very secure and encouraged best practice. However, several mentioned the risk of accidentally exposing data. There are multiple configuration and security settings for many cloud systems. D7 is *'concerned about just how much time and effort you have to invest as a customer of these things into learning their nuances and how to configure them securely,*' explaining that when using more than one of them there's *'kind of a combinatorial explosion*':

> 'GitHub is enormously complicated, right? And Google Cloud is enormously complicated. Now suppose I want to set up my GitHub stuff so that it can automatically deploy code into Google Cloud. Well now I have to reason about the intersection or I would argue the cross product of the surface of complexity of those two things together. And it's difficult.'

CD15 described configuring AWS and Azure access for new team members: *'you get items 1 to 20 of 674 available security roles, and they're all called things you've never heard of.*' D3 worried about smaller organisations' ability to keep up. Others felt that there was a lack of control over cloud infrastructure.

The advent of large language models (LLMs) such as ChatGPT [65], which happened while we were conducting interviews for this study, has entailed a sea change in software development [48]. Subsequent to the release of ChatGPT, we introduced an interview question on the likely impact of recent AI developments on code security. Many interviewees felt that poorly-understood AI-generated code would be insecure. The feeling that such code would rapidly appear in codebases was widespread. D5 was scathing:

> 'If the issue is that humans cannot write secure enough code, an AI model which does not have any further understanding beyond making it look like what a human would write. I don't yet see the value.'

Issues with developers copying poorly-understood code are certainly not new, and there has been extensive study of developers' use of copy-and-paste [63, 83]. CD15 picked up on this, saying *'tools like GitHub Copilot are making it easier to write code without understanding what the code does, which is just the same copy-and-paste repeated at scale.'* A few interviewees thought that use of AI-generated code by novices would put additional pressure on senior developers during code review. There were concerns that developers could use LLMs as a learning tool to gain understanding, not grasping that the information they received could be wrong. Experienced development lead SD14 commented: *'I'd be amazed if it's not happening, to be honest.'* SD11 said, *'I worry about it a lot.'* Without a good understanding of the possible issues, senior management could get caught up in AI hype and fail to control for the risks.

Subsequent events suggest that these concerns were well founded. CodeLMs (language models for code) have developed as essential developer tools, but they are vulnerable to poisoning and other attacks, and developer awareness of potential security issues is low [17, 56]. Meanwhile, 'tokenmaxxing', which is a developer productivity metric that counts how many units of AI compute the developer uses, has become fashionable in management circles [21, 46, 102]. (The term 'tokenmaxxing' is also used for the developer practice of gaming the metric by deploying agents for personal use or busywork [73]). A single-minded management focus on AI could distract from other considerations like the security of the code produced.

4.1.2 Prioritising software security can have an adverse impact on usability for developers. Increased software security can affect platform usability for developers. Several developers described usability

issues with Dependabot, a GitHub tool for updating third-party libraries, that were caused by a security improvement in the tool. D7 discussed a new security level in a mobile operating system that presented challenges to developers for the system, even those working in the same organisation. CD16 used the term *'security theatre'* to describe new security bars for mobile devices, which they said had inadequate explanation and support for app developers. Several developers described situations where rigid rules were unenforceable, did not work and wasted time. Regarding secure coding policies, CD15 said that *'it was too easy to write bad code that ticked all the boxes.'* Policies on updating dependencies were widely seen as too all-encompassing, and were ignored by more than one participant. Questions on security tools raised many responses familiar from the literature [44, 67, 88, 89]; some are too complex, slow and unreliable. D3 remarked that there can be pushback when tools suggest changes because *'there's a lot of ego around code.'* Too many alerts from security tools can cause developers to ignore or abandon them. Or, as SD12 told us, *'Those that we knew how to fix, we did. Some others we suppressed [...] Of course, maybe some of them are not really false positive.*' An app developer told us about a workaround they used to avoid mobile app security restrictions. Concerned that, as a result, their program could be an entryway for hackers, they consoled themselves by reflecting that it is not widely used and *'not everybody uses it in root mode.'*

4.1.3 Developers may code securely despite management. Some developers and teams are self-motivated to code securely regardless of management priorities. We asked developers whether secure coding requirements got in the way. Many felt that they did not, with D3 saying *'I would say no, because that's part of the job [...] if you think they're getting in the way, you're probably not a great programmer.'* Asked about their motivations, security-aware developers often equated security and quality. They appear to include software security in their mental models for good software, and are motivated to produce code that *'is as good as it can be on all aspects* (SD11).' They do not want to have to deal with the technical or reputational fallout that a breach could cause. CD15 gave a graphic description of the consequences of discovering a security issue:

> 'There's been a few incidents [...] there have been occasions where something went wrong and you suddenly realise *"this could potentially be quite bad."* And we don't know how long it's been bad for and then follows a period of waiting where you fix the defect, you notify the people responsible, and then you wait. And you wait and you wait. And when after 18 months, you haven't seen any articles in The Register about it, you know [...] maybe that one was okay, Maybe we got away with it.'

A couple of developers mentioned peers whose security enthusiasm sparked their curiosity. Peer example is a known security incentive [41, 105]. Asked whether they considered themselves software security drivers in their organisation, several responded that they did. This manifested as reminding others to check security and access, pushing security in code reviews and at architecture meetings, and introducing static analysis tools to the build. Several participants observed that security was valued and seen as a positive by the technical staff, but not prioritised by senior management. The danger here is that if the drive for security is not prioritised and supported, it will not be resilient. Asked how being a security driver manifested, SD11 said *'trying to modernise practices, trying to get people more informed, get processes in place that allow for these practices to be in place even when I'm not part of the organisation any more.'* When the motivated developer leaves, good practices may fall by the wayside. Good process is key to good security but its long-term maintenance requires management support.

4.1.4 Developer security wish lists reveal a desire for greater security prioritisation. When asked what changes would improve secure coding in their environment, developers' answers frequently reflected inadequate security prioritisation and budget. D7 argued that security tools should be free

and widely available. Several developers mentioned training-related aids, for example *'better tools and documentation for configuring software as a service*' (D7), *'an awareness of what [code analysers] do and why one would use them*' (SD13), *'more time, possibly better training available*' (SD11), and *'resources, but guided*' (D6).

Several interviewees advocated for having a security team. SD13 wished for *'having a security officer attached to the project [...] that would be a very good thing if there was someone to worry about all this for me.'* Working in an environment without secure coding support, D6 felt that true security *'would have to come from above. They would have to say "You need to have better security in your applications, and this is the task that we're giving you rather than some afterthought."'* Similar ideas came from SD9:

> 'I think security must be enforced through some sort of procedures like periodically or part of the day to day activities [...] it has to be enforced before it becomes part of the developer's self.'

Confirmation of the importance of security support can be found in a paper by Tøndel et al., who observed *'a drop in security focus*' after the departure of a Security Officer from a company they were studying [96].

4.1.5 Good software security practice is expensive in terms of time and money. Good software security is expensive, as CD15 reminds us:

> 'I think there is this Utopian ideal that some armchair keyboard warriors have like, *"well, no, you just don't let the boss decide. You just make it secure.*' Yeah, that's not how real companies work, you know? It's like *"We don't have any money to pay salaries." "Why not?" "Oh, because the software team was busy doing security instead of building anything we could actually sell for the last six months."*'

Coding securely day-to-day can often be relatively cost free once developers are familiar with good coding practice. But other aspects of security are not cheap. SD11 told us that in a previous job *'we were encouraged and even incentivised to try to break our own systems from a security standpoint. It's better if we find and break that then if an attacker does that, but I understand, it can be expensive to run a team that way.*' As tools improve and use of AI is leveraged, this expense may decrease. However, most excellent security tools are costly. For example, SM17 discussed obliging acquisition candidates to submit code to a security scanner, adding that *'this is a top tier tool and the companies typically don't have access to this level of tooling. We often uncover things for them that they're surprised by, and actually, it's of huge value to them.'*

4.1.6 Smaller organisations may have fewer security practices in place. In prior research, we have documented a correlation between organisation size and software security activity [82]. Unprompted, some interview participants suggested that organisation size had an effect on software security practice. For example, MC21 described restrictions on open-source libraries but added the caveat *'I would say that's in the large [organisations]. The small to medium; I doubt very much you'd see it there.'* Also discussing use of open source, CD15 said that:

> 'It comes down to a pragmatic understanding of the probability of a vulnerability, the probability of the vulnerability actually getting exploited, the fallout, if it was to do so, and the cost of mitigating against that [...] *"We'll take the risk."* I've not really heard any organisations say it that succinctly, but I think that's the attitude that a lot of certainly startups and smaller places take, if security is not their core business.'

4.2 RQ1: What is senior management's understanding of and attitude to software security?

In order to understand how software security is prioritised by management we must begin by investigating whether senior management understand software security and its implications.

4.2.1 It is often assumed by senior management that software security is in place. Software security is frequently described as a non-functional requirement like efficiency and maintainability. There can be an assumption that it is present, without the corresponding clear direction from management [88]. When security fails it often fails invisibly. CD15 said: '*the biggest problem with security is that nothing actually crashes when it's broken*,' adding '*you can check the locks on all the doors that you know about, but you can never be 100% confident that there isn't a hole in the wall that you missed*.'

In many organisations, management seems optimistic about code security, and appears to either know nothing about it or to assume that secure code is the default. SD13 attributed this to its invisibility:

> 'From the perspective of the business, it's a non-functional requirement. It's like when people want a new office building and they talk about the number of desks and the lighting and how much natural light there is and these things. But they don't talk about the toilets. They expect toilets and running water to be there.'

MC20 described how management expect software security to not only be there, but to be thoughtfully engineered:

> 'If you're driving a car, you're going to assume that the engine is sound. You're assuming that it's manufactured to a certain standard. It's not. We don't engineer software, we develop software and it's not as rigorous a process as people think it is.'

4.2.2 Senior managers' understanding of software security depends on their background. Several developers observed that their management did not have the relevant competencies to understand secure development. D4 felt that leadership in their organisation were only '*very peripherally aware of [software security]*,' being '*more from the insurance domain*.' D3 said that in one organisation they encountered, '*senior leadership were journalists in the food space. It would be unfair to expect them to have any experience whatsoever in software security*,' adding that '*part of being in senior management is not knowing all the things that go on underneath you [...] they can't know everything*.' While this is true, there is undoubtedly a benefit to having a rudimentary understanding of issues which could become a systemic threat to an organisation's existence.

Senior managers had not observed a marked degree of attention to software security in their working environments to date. MC21 felt that '*it does warrant some focus in and around C-suite tables, to the extent that I have not witnessed it in my last 15, 20 years*.' SM18 felt that management should '*give a level of importance to it at a higher level and then obviously allocate resources after that, so understand what the risk is*.' SM19, working for a large multinational, felt that things were improving: '*senior management is getting much smarter. They are starting to bring in stakeholders and C-suite execs that have a security mindset*.'

4.2.3 Senior management's focus is subject to competing tensions. Cybersecurity and software security are different, with very different preventive measures, potential adverse effects and potential legal consequences. We observed a blurring of these distinct identities when talking to senior management, and indeed D6 described an over-reliance on IT security that may have caused management over-confidence resulting in low secure-coding emphasis. Senior managers often do not distinguish between cybersecurity and software security. In management teams that MC21 has encountered, software security '*just doesn't feature as part of the discussion*.' MC20 said that '*I would*

say a lot of people are focused on monitoring and detection as opposed to prevention and software. I just think it's easier to detect and monitor than it is to build in.'

This focus on detection may be partly due to hiring tensions. Secure software development needs people, and software security experts are not only expensive but are also in short supply [32]. SM18 was concerned that while there are many security recommendations, to act on them their organisation needs '*champions*'; '*people who are able to fix things, to be able to implement the changes.*' The need for more people with the requisite security expertise was reiterated by others.

4.2.4 Senior managers may sabotage software security. Top management may ignore software security issues, and may even subvert security practice. Determining the true priority given to software security within a specific organisation can be difficult. There can be inconsistency between management's public posture and actual software-security activities in an organisation. D6 was forthright when asked if their manager cares about software security: '*I think if you asked them that, then they would say yes, but I'd probably have to say no.*' Later, they suggested that managers would '*put on their biggest smile*,' but '*it's just not really anyone's priority.*' D3 felt similarly: '*Obviously no one's like, you know, twirling an evil moustache or anything like, "we don't want any security" [...] but the priority is always features.*'

One interviewee had seen management try to circumvent their own stated secure coding policy:

> 'I can think of a couple of incidents where somebody publicly has to say that "*This is our commitment to security*" and then privately afterwards will be like, (whispers) "*Do you need to do that?*" And you're like, "*Well, you just said we were taking this seriously.*" And they're like, "*Yeah, but you know what I mean.*" '

Another participant told us: '*So right now we're using an application that doesn't actually have authentication. It's not like minimum, it's just not there. And it's very critical. But there's this huge rush to get the work done.*'

We were entrusted with the description of a security workaround that we cannot describe in detail here lest the participant be identified. The essence was that there were careful, audited precautions taken to protect customer identity for sensitive data. The mapping between customers and their data was removed from the software entirely and stored offline. These precautions were, however, silently bypassed at top management's request. A spreadsheet mapping customers to their sensitive data was maintained on the network, to facilitate easy processing by secretarial staff.

We do not argue that managerial security sabotage is universal, or even widespread. This is a qualitative study devoted to unearthing rich data and themes in the area of senior management's effect on and attitude to software security. Observations of managerial misbehaviour of varying degrees from multiple participants reveal that such misbehaviour exists. Researchers and policy-makers should be aware of it. Therefore, we believe that this finding would reward further study.

Answer to RQ1: Senior managers may assume that software security is in place. Their understanding of software security depends on their background. They may not distinguish between cybersecurity and secure development, and even when they do their focus on it is often in tension with other requirements and constraints. In some cases, managers may actively downplay or sabotage software security, perhaps due to conflicting demands.

4.3 RQ2: How do management priorities affect software security within organisations?

Software security is in constant tension with other goals. How this tension is resolved depends largely on the priorities set by management, which vary between organisations. In the following sections we discuss different prioritisations and their consequences.

4.3.1 If software security is not clearly prioritised, functionality will take precedence. A very experienced IT consultant, MC20 speculated that '*maybe 10% of developers would consider secure coding as a practice.*' It should be impossible for developers not to consider secure coding if it is prioritised within an organisation. But unclear responsibility for software security is a well-known issue [88, 91, 96]. Often, secure coding is seen as causing missed deadlines, and can be ignored if it impacts the time to market. As SD9 told us, '*it needs thought and it needs more time. And time is that commodity that nobody has, and the functionality needs to get out as fast as possible.*' Asked if they felt supported writing secure code, D6 gave an emphatic negative:

> 'I've got stuff to do and there's just never really any time and even if there isn't a backlog, someone will make one, as soon as it's free. There's just never any time to think about it.'

Several others, asked what the biggest barrier is to their coding securely, replied immediately '*time.*'

Participants working in a project's early stages told us that security would not be addressed until functionality was in place. The immediate objective was to produce working code. Only then would security be considered. This has long been considered bad practice [57], with interesting confirmatory evidence from Fulton et al. [33]. Sometimes insecure software is released based on management risk-reward calculations. Asked about causes of big secure-coding behaviour changes observed during their career, SM19 replied '*time to market*', adding later '*Come hell or high water, [leadership will] move that code out. They'll move that release out and then accept risk.*' The security vs. time-to-market tension is pervasive in industry. By contrast, D8 had experience coding open source, and remarked that in open source '*deadlines tend to be a bit more flexible so that you don't get the "let's just get something that looks like it works out and we can fix it later."* ' Several participants also raised the visibility of open source code as a positive attribute, with defects being picked up quickly (D2) and '*a community of people out there who is willing to test your software*' (D5). However, they mentioned that there is a tension there also; some of those assessing the software may be hackers looking for exploits [15, 34].

4.3.2 Where security is not a management priority, developers are lost, with no guidance. Where there is a complete absence of relevant support, developers may abandon secure coding. D1 told us that '*in my professional working environment, the extreme lack of security is what led me to on my own, not really attempt [to code securely].*' Developers may lack secure-coding training [79], and do not always include software security in their mental models for good software [9, 33]. They may not understand security-related events. They are not always good judges of whether their code should be written securely. Many participants stated that they suffered from '*a lack of knowledge*' (D5, SD10, D2), '*[lack of] knowledge of what the vulnerabilities are and really how they manifest in code*' (SD11). Others are largely unaware of secure coding practice, but they do not know this. They have latched on to one piece of advice they have encountered (for example, to use a modern language, or to use exception handling) and believe that by following it they have secured their code.

Some uncharacteristic uneasiness was apparent when participants were asked how they checked whether their code was secure. A couple of them obfuscated, describing instead security precautions they take during development, such as coding in a memory safe language like Rust. Developers do not always have a good grasp of their own security posture. D7 explained:

> 'you make trade-offs in order to get to your MVP [minimum viable product] or to 1.0 [...] so sometimes defence-in-depth security stuff, or that kind of rigorous analysis that would let you have an idea of what your security posture is, will get deferred.'

Others listed possible ways of checking code, but without claiming to have used any of them. D6 was direct:

> 'Probably the most straightforward answer that I can give is that most of the time I don't [check whether the code is secure] it's fun to talk about [...] but it's not a priority. So I guess if I were to give you a rating like 1 to 10: really the bare minimum, like 3 or 4 in terms of safety.'

Participants experienced a lack of training in both universities (D7) and the workplace, where, as D6 said, '*if you don't have that particular guidance from your mentors [...] it's really hard because there's a lot of jargon and it's hard to piece together.*' In prior work we established a widespread dearth of secure-coding training in the workplace [79]. For this study we asked developers whether they had been offered secure coding training. Several had not. For others, online training was theoretically available from generic suppliers like Pluralsight or LinkedIn Learning, but it was not mandatory and there was no time to take it. Some mentioned the availability of on-the-job training and learning opportunities in their workplace. SD11 was pleased that their organisation was finally providing information on vulnerabilities and how to fix them, instead of just standard advice on non-coding topics like phishing. Questions about barriers, and potential aids, to code security, revealed a widespread desire for more training and support.

Guidance from the broader industry was also seen as inadequate. CD16 described trying to research good security information online, saying '*the information is very kind of spotty [...] the security information that needs to be known is not concisely found. It is so spread out and often so hidden.*'

4.3.3 Where they have not been told to prioritise security, developers often defer to others. Developers who had not received software security as a priority tended not to think about it, or to defer to others. D6, for example, said that '*I'll have to be honest with you, during my daily life in my job, that's not really something I ever think about [...] it's just sort of in the background. And I hope that third party libraries take care of it for me.*' Others rely on audits, bug bounties, third-party libraries, continuous integration (CI) security checks, or on secure defaults in infrastructure-as-code scripts to check the security of their code. One participant described security improvements to the configuration of SQL servers in their organisation, noting that such cybersecurity changes can reduce the impact of threats such as SQL injection.

4.3.4 Even when software security is prioritised, compliance may be prioritised over security culture. Participants working in industries subject to regulatory constraints, such as banking, had detailed software-security processes in place. Unfortunately, some described aspects of secure coding practice in their organisations as a box-ticking exercise. For MC20, the problem lay with development teams, engaging late with security teams to tick a box. This suggests that software security culture is poor. Security should be under constant discussion [94]. However, MC20 also expressed a familiar [90] cynicism: '*the reality is when the security team get involved, there is an element of they're coming to stop you as opposed to engage and collaborate.*'

For SD14, the box-ticking was on the management side, providing CI functionality and security tools without a security team to explain and support the process. They felt that teams were getting flooded with support tickets automatically generated by the CI pipeline, resulting in cognitive load. In the absence of a supporting security team, developers '*have to become experts in everything, and nothing to do with what actually they're trying to build.*'. They added '*the problem is of course when the effort is there, then they can say, "Oh, we're so secure. We've implemented X, Y and Z", but on the ground is it really?*' This illustrates one of the problems with moving from traditional, manually-evaluated security gates to automated security testing. Analysing test results can be

time-consuming for developers if security expertise is not available. There can be squabbles over their validity. When introduced to legacy code, security tools produce an overwhelming number of issues, many of them incomprehensible to the uninitiated. Without in-house experts to explain and advise, the opportunity to build a security culture is lost. Other participants expressed appreciation of automated vulnerability checks but specified that the system must be well-maintained.

D4 and SD10 had experienced the recent creation of security teams in their organisations, and felt a resulting increased expectation that they themselves would code securely. However, they did not find these teams helpful. SD10, who was highly motivated to code securely, but seemed lost, spoke about reaching out to them, saying '*the IT security team I think has other bigger things at the moment but just giving some direction on, on where to begin.*' D4 complained that their security team did not address an issue they raised. D6 found theirs so slow and red-tape bound that they were effectively a hurdle to secure coding.

Another example of poor culture was a plea for help from SD10. Their organisation mandates secure coding and includes it in their annual goals. However, when asked whether they used a written security checklist, they replied to the first author: '*I would ask you a question, are there resources to help me get started on that path? That's kind of what I think's lacking, you know?*' Nevertheless, even where there is a compliance mentality, secure-coding regulation can have positive outcomes for developers. CD16 learned best practice for storing user passwords during a mandated code audit conducted by a '*super security nerd [...] and the good thing was that that nerd was very kind of accessible [...] and we could nerd it out then. And he was also willing to explain things that I didn't understand, and didn't just brush me over.*' Furthermore, where compliance is required, this gives interested developers a good argument to push for increased software security resources from management.

4.3.5 Where software security is prioritised, developers have the resources to code securely. Developers who worked in organisations where they felt secure coding was appreciated talked about how those values were reflected within the work environment. Developers told us that there was enough time available to code securely. They were able to put thought into implementing their own security checks, such as adding extra logging, or implementing static analysis checks and security code reviews. SD12 had found multiple issues via manual fuzzing and was working on automating fuzzing for all new code. Others felt that resources such as training and time were likely to be available if requested. Two developers identified this explicitly as being a result of support from upper management, with D4 saying that top management at their organisation were '*very receptive*', and SD14 telling us that '*the whole push on software security has come from the very top.*'

Many participants felt that secure development emphasis in the work environment improves understanding. Reasons given included learning from tools and from peers. D3 expressed this well:

> 'In companies that encourage security to be front of mind, there are more people who care about it, so there's more opportunity for discussions about it and people to bounce ideas off of and learn more.'

Working in an atmosphere that values software security improves developers' software security interest. Developers were positive about work environments where secure-coding training was given, or was explicitly available. Other positives in the work environment included having good secure-coding values but an absence of blame culture, as CD15 described:

> 'Everybody involved, I think, appreciates that there are too many variables and too many moving parts for "relying on a person not to make mistakes" being a reasonable security policy.'

Answer to RQ2: Clear prioritisation of software security is vital. It signals that security is important. It helps to prevent ad-hoc and inconsistent security activity that leads to developer burnout. It supports processes and skills that embed secure development in the company culture. Without it, developers are lost, training and culture are absent, and any processes introduced will be abandoned when the motivated developer leaves.

4.4 RQ3: What factors motivate senior managers to prioritise software security in organisations?

Insight into what motivates senior management to prioritise security could provide useful strategies to encourage it. We found that personal, external, and regulatory factors affect senior management motivation.

4.4.1 Increased visibility and understanding improve motivation. The absence of software security can be seen as a business risk, like natural disasters, rather than a technology risk. Participants felt that managers would be more motivated towards secure coding if they deeply understood the risk. Several participants talked about raising the visibility of software security at board level, for example by giving senior management a dashboard to monitor software security metrics and quality. Others suggested tabletop exercises with leadership.

An inadvertent trigger for awareness is an internal or external security breach. With experience in senior management, SM19 and MC21 felt that an internal breach would definitely push software security further up the management agenda. Security breaches have been cited in other research as motivators for increased software security [9]. In prior work we found that there was a long-term increase in personal and organisational software security practice after approximately 55% of code breaches [80]. Interestingly, those of our participants who had experienced a breach such as a ransomware attack within their organisations described it as a trigger for increased *cybersecurity* rather than *software* security (SD10, D6, MC20). However, breaches in external organisations appeared to initiate a spirit of enquiry amongst management, in the experience of some interviewees. CD15 told us:

> 'I remember my manager, the managing director at the time, we'd been trying to get him to take security seriously. He came over to my desk and he's like this to me: "Could that happen to us?" '

D3 had had a similar conversation with management. Others mentioned the Log4Shell incident [69], the NotPetya ransomware's effect on Merck [31], increased federal scrutiny of US-based banks, and the effect of vulnerabilities on competitors as motivators for greater software security interest.

SM18 suggested that, since actual breaches to the organisation are undesirable, documenting such well-known breaches with adverse consequences would be useful. Case studies could be brought to senior management to increase software security awareness:

> 'These are already working for cybersecurity, but I don't think software security stands out so maybe more along the lines of, from a business perspective, what companies have lost from not building in security by design into software?'

4.4.2 The need to meet client expectations can be a software security driver. Several participants mentioned meeting client expectations as a software security driver for senior management. This aligns with prior findings from relevant literature [54, 88, 96, 101]. SD12's organisation works on security software, and needs to show that their own software is secure to earn customers' trust. SM18 gave client expectations as the top reason for prioritising software security, followed by legal and reputational concern, which they saw as linked.

4.4.3 Regulatory compliance, standards, and greater legal obligations can motivate senior management. Compliance was seen as a good software-security motivator for senior management. CD15 used the payment-card PCI-DSS accreditation process as an example of successful compliance requirements, saying '*these people actually know what they're doing. And this is clearly based on experience of breaches and problems over many years.*' It is sometimes argued that an emphasis on standards or regulatory compliance can result in a failure to focus on software security itself [77]. This is true, but an absence of standards and regulations can result in a state of zero security, which is worse. Regulation and compliance requirements can at least trigger a security conversation, and can provide security-conscious staff with ammunition. However, D3 discussed the problems with the current risk landscape for organisations:

> 'It's very hard to quantify what the risks are of insecure code. And at least in the modern world, insecure code hasn't been all that expensive. We may look at some of the big breaches, you know, fine, they pay for credit reporting for 100,000 people for a year. It's just that that cost is cheap enough that it's not worth trying to fix the underlying problems.'

MC20 was also in favour of a degree of regulation, saying '*I think if you're relying on good practice without any validation or audit, it's not going to happen [...] there has to be consequences.*' Several interviewees work in regulated environments, and described how regulatory compliance takes resources, and is therefore in tension with other organisational goals. SM17 explained the complex legal requirements in the US and Germany, among other countries, for organisations that are considered part of critical industry. Discussing Germany's BSI-Act, which regulates KRITIS (critical infrastructure) [23], they said that '*you need security tools to evidence, and you need secure software development practices so that you're not showing really bad numbers.*' Over years of dealing with acquisition candidates, they have found that '*where we've seen really good governance on software and infrastructure security, a big driver is the regulatory requirement*,' singling out GDPR where penalties can amount to 5% of global revenue. This sentiment was echoed by SM19, who noted, however, that regulations such as GDPR and SOX slow down time to market. CD15 introduced a resulting concern:

> 'Software and the companies that rely on software have this expectation that you can do amazing things very, very quickly and saying to everyone, "*no, no, no, we need to slow this down. We need to start certifying websites the way we certify aircraft or we certify medical devices.*" You know, while you were there doing your paperwork, some competitor from an unregulated territory would run rings around you and steal all your customers.'

This is somewhat disingenuous. Websites are not the only type of software. There is a spectrum of impact from a brochure website to a nuclear power station, and there is also a spectrum of regulation. Asked whether legislation around software security would increase the focus on it, MC21 responded: '*100%!*' Others also felt that changing the financial implications of insecure software would help in the security battle. SD12 introduced another real-world simile: '*eventually it will have to appear like it is engineering, like the bridges used to collapse all the time in the 18th century, and now it is quite rare.*' The recent introduction of the CRA in Europe [98], which mandates security considerations when developing software for the EU, should be a step forward in this process.

Organisations are not people, and regulatory consequences or punishments for businesses do not always affect management motivation, as observed by CD16:

> 'If someone finds out that you did it wrong, then you're in trouble [...] but of course, that doesn't work with law, with legal law. If you are big enough, you can always drag

> it out [...] the CEO or the COO or CTO probably is in the end responsible for it. They usually earn so much money that they say just, "*Oh, I'm already a multimillionaire. F*** it, I'm just quitting*." And then you have to go legal after them and then the damage is already done.'

One of our respondents had a solution for this when asked what developments they thought would push software security higher up the agenda for senior management: '*Follow the money. Follow the money. Or jail time.*'

Answer to RQ3: Personal, external and regulatory factors affect senior management motivation. Increasing the visibility and understanding of software security within the organisation improves motivation. A grasp of consequences improves understanding. Client expectations can trigger increased software security motivation, as can regulatory and compliance pressure. However, if there are no personal consequences such pressures may have limited value.

4.5 RQ4: What factors have study participants observed that changed an organisation's software security prioritisation?

Research on secure software often assumes that a state of excellent security is an end goal towards which organisations strive. However, sometimes organisations travel rapidly in the opposite direction. Improvements or deteriorations in software security were of interest to us. They may provide lessons for security advocates and policymakers.

4.5.1 It is difficult to impose security as a priority when outsourcing software. Where a tight budget exists, there can be a tension between the need for secure code and the desire to outsource coding to save money. This is a real issue for small organisations without the financial heft to enforce supplier compliance. It is also a problem in large organisations. SM19 felt that an offshore model had balance-sheet pros but added:

> 'There's probably a drop off in the quality of the security that's getting baked into these products, which then leaks more security vulnerabilities and anomalies into the code that gets propagated to our production environments.'

There was little mention of outsourcing by our participant pool. Most interviewees discussed the software that was developed within their organisations. An examination of how outsourcing affects secure development would be an interesting topic for further research.

4.5.2 Organisational turbulence affects prioritisation. Organisational turbulence such as financial constraints, downsizings, acquisitions and divestitures may affect prioritisation of software security. This may be due to the cost of doing software security well. CD15 talked a little about organisations going bankrupt: '*I was working at [redacted] when they were placed into administration [...] the impact there was every single person who knew how the system worked was told, "give us your keys and go home."*'

Asked whether organisational turbulence affects software security in their experience, D3 discussed how, the more institutional knowledge is lost, '*the more likely you are to make mistakes or to break assumptions that might have been part of the security posture of that software from before*,' adding that organisational turbulence '*makes it significantly harder*.' D6 had observed multiple departures from their organisation, and they described the resulting feeling: '*there's so much stuff to do and so little time to do it that I have to focus on getting this working. And if it's not secure, no one is even going to know who to blame. So I'm not going to focus on it.*' D5 described a downturn in software security during a period of organisational turbulence that ended in a downsizing: '*there was a lot of pressure to [...] "we just need this by this date, it has to work and it doesn't matter how you*

make it work." ' Downsizing can also result in outsourcing, with the potential security implications we saw in section 4.5.1.

SM17 had observed a diminution of software security concerns in divested organisations, and an increase in acquired ones. CD15 on the other hand discussed how the difficulty of fixing security in a newly-acquired organisation can mean that it never happens. SM17 had considerable experience of assessing software security in organisations that were due to be acquired. In one case, they found that the target organisation had a higher level of software security than the acquirer, but that was very unusual:

> 'Companies being acquired, in my experience, fall into two categories. One: financially, they're not doing well, or two: they know exactly what they're doing and they're sizing themselves for acquisition. And in both cases, you're talking about reducing costs, rationalising, only having the people you need to serve your consumers [...] So the organisation that devs are operating in when they're being acquired is often budget restricted. For different reasons, but it's the reality of it. So [...] we know they're not going to be in great shape, typically.'

Asked whether they had encountered sudden increases or decreases in security, they zeroed in on financial constraints:

> 'I think companies that are going through any sort of a financial crunch, or that they're just being shaped for sale, can impact security. I think the financial crunch is more severe [...] because some of the first things that get cut: training budgets, R&D, security is close behind that then [...] just keep the product afloat, just keep the company afloat, just fix what we need to fix so that the customers keep paying us money.
> And even regulatory fears start to recede if the company is non-viable, right? Because it's like, what are you protecting? I think the idea of protecting your customers and your information and your PII [(personally identifiable information)] or PHI [(personal health information)], it starts to become secondary. Folks know as well it's coming, so some of your best talent might walk. Your more senior people especially.'

Answer to RQ4: Acquisition by a security-aware organisation can result in improvements to secure development. Downsizing can reduce institutional knowledge. Budget constraints are a strong driver of deterioration in secure programming, and may ultimately result in reduced protections for customers as organisations or products reach end of life.

5 Discussion

Researchers in *cybersecurity* economics have attempted to balance costs with expected losses when calculating firms' optimal cybersecurity investment, concluding that underinvestment will always occur when the cost (to others) of externalities is not considered [36]. This suggests a likelihood that there is an underinvestment in *software* security, because poor software security can result in wide-ranging adverse external consequences such as the hacking of customers, the cost of which is difficult to calculate. In fact it is not clear that senior management is even sufficiently aware of software security concerns to attempt the calculation.

Different industries are subject to different cybersecurity obligations. Those with greatest cybersecurity constraints are critical industries such as nuclear, energy and water. Such industries may also have regulatory software security constraints. For example, there are four pages on '*Developer Testing and Evaluation*' in version 5 of the 465-page *US Nuclear Regulatory Commission Regulatory Guide 5.71*. In the EU, requirements have tightened for '*essential*' and '*important*' entities under NIS 2 [97], which includes mandatory secure development lifecycles. Such entities would therefore be

expected to have software security practices in place. Some will embrace a software security culture; in others management may treat software security practices as a compliance burden. We have found that some managers will even subvert software security practice for convenience, although given our small sample we do not suggest that such malpractice is widespread.

The EU CRA introduces basic software security requirements for all organisations producing software for sale in the EU [98]. Its main obligations will apply from December 2027. Industries not categorised as critical may be subject to no relevant regulations until then, even though almost all products in all industries in an always-online world are vulnerable [64]. To quote the Act [98]:

> 'Under certain conditions, all products with digital elements integrated in or connected to a larger electronic information system can serve as an attack vector for malicious actors. As a result, even hardware and software considered to be less critical can facilitate the initial compromise of a device or network, enabling malicious actors to gain privileged access to a system or to move laterally across systems'

The participants in our research came from a range of industries, some highly regulated, some not. Almost all expressed a belief that software security was important in their work environment. This study has a number of clear implications for different stakeholders, summarised in Table 4. We discuss these below.

5.1 Prioritisation of software security

The leadership team sets the priorities for an entire organisation, and so a failure to *visibly* prioritise software security in an organisation sends the implicit message that it is not a priority [8]. The analysis for RQ2 included feedback from developers that echoed themes found in other research [10, 58]. Without clear priorities, developers flounder. We found that when management do not prioritise software security, teams or lone developers may do so; while commendable on an individual level, this situation has parallels with shadow cybersecurity [47]. It can result in undirected, ad-hoc software security activity that is inconsistent, and sometimes not best targeted to the greatest threats. Individuals may burn out or leave, abandoning unsupported software security efforts.

Help, advice, and resources are at the top of developers' wish-lists when it comes to security, which is not surprising given the fast pace of change in software development. Genuine management prioritisation includes allocating a budget for secure processes and hiring expert staff to support them. Investing in expertise encourages a good security culture, helping to prevent issues like an outsourcing mismatch, or a compliance mentality [7, 8, 40]. We found that environments where software security is prioritised provide developers with the resources they need. We advise C-suite managers who have a desire to support software security to overtly prioritise it, introducing explicit software security budgeting, time allocation, monitoring and continuous support. Policymakers should consider ways to encourage such prioritisation. Researchers should be aware of the relevance of senior management support to developers and teams. They should explicitly discuss this issue when evaluating secure coding practice.

5.2 Senior management awareness and motivation

Previous research has unearthed factors that encourage software security in organisations, but has not focused directly on software security motivations for senior management [40, 64, 101]. When investigating RQ1 we found that not all senior managers have the skills or experience to identify software security as important. We advocate training for senior management on how software security differs from cybersecurity, and why it deserves explicit attention and budgeting.

Table 4. Findings and Recommendations

Finding and Sources	Recommendations
1) Clear prioritisation of software security is vital. It signals that security is important. It helps to prevent ad-hoc and inconsistent security activity, and supports processes that embed secure development in the company culture. See sections 4.1.4, 4.3.1, 4.3.2, 4.3.3, and 4.3.5.	**C-suite**: Explicitly and visibly prioritise software security. **Policymakers**: Consider ways to incentivise prioritisation. **Researchers**: When assessing secure-coding practice, also assess the priorities signalled by senior management.
2) Good software security practice is fluid and can impact other areas of development. See sections 4.1.1, 4.1.2, and 4.3.4.	**C-suite**: Serious attempts to support secure development require expert guidance and attention.
3) Software security has associated costs. See section 4.1.5.	**C-suite**: Genuine prioritisation of secure software requires a budget and monitoring. Lip service is not enough.
4) Senior management are rarely software security experts. See sections 4.2.1, 4.2.2, 4.2.3, 4.4.1 and 4.4.2.	**C-suite/technical staff**: promote senior management understanding of software security. Include training. **Clients**: request software security assurances and verification.
5) Outsourcing can make software security more difficult. See section 4.5.1.	**C-suite**: Outsourcing debates should include the effect on the organisation's software security control.
6) Smaller organisations may have fewer security practices in place. See section 4.1.6.	**Policymakers**: consider ways to facilitate software security, especially for poorly-resourced organisations.
7) Senior management may sabotage software security and are not always motivated to prioritise software security. See sections 4.2.4 and 4.4.3.	**Policymakers**: Regulations and legal obligations should include personal obligations for senior management. **Researchers**: The prevalence of senior management malpractice and the effects of relevant legislation should be investigated. **C-suite**: Senior management should guard against this behaviour.
8) Organisational turbulence affects prioritisation. See section 4.5.2.	**Policymakers**: Consider organisational turbulence effects on software security. **Researchers**: Further research on the effects of organisational turbulence is needed.

Our research for RQ3 into management's perceived security motivations confirmed motivating factors such as client expectations, and compliance and regulatory concerns. Visibility also emerged as a major contributor to managerial interest. Participants contributed several ideas on how technical

staff can engage senior management's interest in software security. These included training, creating a software health dashboard where bug and vulnerability counts and fixes are tracked, using tabletop exercises to simulate software issues, and introducing case studies as a warning device to explain how catastrophic issues have affected other organisations in the past.

There is a chicken-and-egg aspect to introducing the C-suite to software security. Senior management may not engage unless already aware of its importance. The EU's NIS 2 Directive [97] mandates personal corporate responsibility for cybersecurity, including software security, for certain critical or important industries. It signals to the C-suite that these topics are important, and incentivises engagement with them. EU software security enthusiasts in industries not subject to NIS 2 can leverage the requirements of the CRA, which will have practical implications for businesses from 2027, to introduce senior management to software security concepts. Small organisations have fewer reporting requirements under the Act. We suggest that policymakers should explore additional ways to reduce their administrative burden while supporting them to secure their code. It is our hope that other jurisdictions will follow the EU's lead and introduce legislation to mandate secure software development.

Client pressure has been shown in multiple studies to be a trigger for increased software security. Concerned clients should ensure that they push for secure development practices, requesting assurances and verifying compliance. Writing secure development requirements into contracts should be done thoughtfully, avoiding gameable metrics. Morales et al. gave a memorable account of some ways that this process can go wrong [58]. Outsourcing development can lead to a drop in the security level of an organisation's codebase, a potential issue that should be factored in by C-suites considering project outsourcing.

5.3 Managerial bad practice, subversion and deprioritisation

Poor software security practice by management may not always be attributable to ignorance or lack of understanding. Interviews with software developers and consultants who observed actual managerial behaviour in real companies yielded disturbing anecdotes. One organisation omitted authentication entirely on a new product due to time pressure. Another violated audited software security policies on the direction of the CEO, for the sake of convenience. Managerial misconduct and subversion of good practice, sometimes called '*corporate social irresponsibility*' has been documented in other domains [106]. It has not previously been described or investigated in the secure development area. This finding is a new contribution to software security research, indicating that some software security transgressions are unrelated to developer decisions, originating directly with top executives.

While some of the senior managers we interviewed admitted ignorance of software security detail, none discussed actively bypassing or subverting software security practices. Indeed, such admissions would be unlikely to come from interviews with senior managers themselves. This shows the value of interviewing participants from other roles to gain insight into managerial behaviour.

As far as we are aware, this is also the first work to research factors causing abrupt changes in software security. When analysing answers for RQ4, we found that noticeable improvements can occur when management begins to prioritise security, or when organisations are acquired by more security-aware entities. Conversely, acquisition can also cause a deterioration in previously excellent practices. We found that events such as downsizing and bankruptcies, which we call organisational turbulence, can degrade an organization's software security posture. Senior management may decide to allow software security to be underfunded in order to size the organisation for acquisition, due to impending bankruptcy, or for other reasons. An example of customer harm: since 2023, a series of zero-day vulnerabilities on legacy network edge devices has caused rashes of ransomware

attacks on corporations [45, 55, 92]. One device had a version of Linux from 2013, unsupported since 2020 [26]. The faults are attributed by some to private equity acquisition and layoffs [30].

This finding is a new contribution to software-security research. It raises the question of whether there should be increased legislative safeguards for customers. Poor software security puts both customers and wider society at risk. There is currently no legal obligation in the US to ensure that consumer software is secure when it is sold, or that there is an ongoing security maintenance strategy [22]. The EU has only recently introduced legislation mandating a secure lifecycle for all products with a software component [98]. Should greater security obligations exist, and if so, whose obligations should they be? An interviewee identified a central problem with regulatory and legislative motivations; if things go wrong, decision-makers can leave. In the US, senior executives can be personally prosecuted for inadequate internal financial reporting controls [84]. Is it socially desirable for CEOs, senior management, or board members to also be potentially legally liable for inadequate or subverted secure development processes? Liability for poor software is a subject of ongoing debate [59, 99]. Those objecting to senior management liability may suggest that it would have a chilling effect on recruitment, development and innovation.

5.4 Further research

The findings in this paper open up several new avenues of research. We have advocated increased regulation governing software security. As discussed, the NIS 2 Directive recently took effect across the EU, with implications for critical and important industries [97]. Major elements of the CRA will become applicable in 2027, and will cover all other software sold in the EU [98]. Research on the effects of these pieces of legislation on software security behaviour would be both timely and welcome. Its findings could inform development of software security legislation in other jurisdictions.

We did not have an opportunity in this work to study the effect of senior management decisions in a range of countries or in specific industries. A study of how C-suite executives approach software security in specific legislative, geographical and industrial contexts could help to tailor software security legislation and support.

Perhaps related to the previous topic, we see value in examining the effect of outsourcing on software security. One participant suggested that outsourcing coding can degrade the codebase. Another pointed out that small organisations are essentially at the mercy of large suppliers, and do not have the leverage to impose security preferences. We are not aware of research evaluating the effect on secure software of either context.

We found some instances of managerial sabotage of software security precautions. Our sample is small and these may be rare events. Further research is needed to aid understanding of managerial malpractice in the software security area. Increased insight on this would be useful to policymakers and other stakeholders.

Given our findings on the effects of organisational turbulence, we advocate research on its long-term effects on software security. Such research could examine the impact of tight budgets, corporate cost-cutting, and corporate failure, including bankruptcy. AI-related rapid downsizing is a new source of turbulence. Its likely impact on software security urgently needs evaluation.

Relatedly, there is increasing pressure from senior management to maximise AI use for software development tasks, for example using the ‘tokenmaxxing’ metric as a proxy for productivity [21, 46, 102]. This new variant of outsourcing may have adverse (or indeed beneficial) effects on code security. We see a need for research on the effect of senior management emphasis on AI adoption, including the range of criteria and metrics used, on the security of software produced.

6 Conclusion

We interviewed 21 professionals working in or consulting for organisations that develop software, querying their perceptions of the influence of senior management software-security prioritisation on software practice. We found that where software security is not prioritised by leadership, developers often seem to be lost and in need of secure-coding training, time and support. Some developers and teams attempt to code securely despite a lack of management interest, but it is difficult to sustain good practice without the necessary resources. We suggest that management invest in developer secure-coding training and support, and budget for the development of a positive secure-coding culture.

We examined the motivations perceived to cause organisational leadership to prioritise code security, finding that education, breaches in other organisations, and compliance and regulatory requirements feature highly. We explored current senior management attitudes to software security, finding that many top executives do not have the background to understand it. Some prioritise it, but others pay lip service to software security while ignoring or even actively subverting it. We explored what triggers tend to cause a shift in software security in organisations. Acquisition by a security-aware organisation can increase it. Downsizing rounds and funding constraints can have adverse software-security effects.

Based on our findings, we suggest providing training on software security and its implications to senior management teams. We commend recent EU software security legislation as an essential step forward in mandating secure coding. We advocate further research on legislative and practical deterrents to insecure software, including the imposition of legal obligations on top-level executives, to ensure good software security practice throughout the lifetime of commercial software.

Ensuring that software is developed securely is in tension with multiple other goals. Without individual as well as corporate accountability, it is difficult to see how organisations can be encouraged to invest suitably in software security. If they do not, the digital fragility of our society will continue to increase, with adverse consequences for privacy, business, and critical infrastructure.

Acknowledgments

We thank the anonymous reviewers and editorial team for their constructive feedback, which has led to a better paper. This publication has emanated from research conducted with the financial support of Research Ireland under grant numbers 18/CRT/6222, 13/RC/2077_P2 and 13/RC/2094_P2, and under the US-Ireland R&D Partnership Programme, grant number 23/US/3939. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] Yasemin Acar, Michael Backes, Sven Bugiel, Sascha Fahl, Patrick McDaniel, and Matthew Smith. 2016. SoK: Lessons Learned from Android Security Research for Appified Software Platforms. In *2016 IEEE Symposium on Security and Privacy (SP)*. 433–451.

[2] Yasemin Acar, Michael Backes, Sascha Fahl, Simson Garfinkel, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. 2017. Comparing the Usability of Cryptographic APIs. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 154–171. https://doi.org/10.1109/SP.2017.52

[3] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. 2016. You Get Where You're Looking for: The Impact of Information Sources on Code Security. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 289–305. https://doi.org/10.1109/SP.2016.25

[4] Yasemin Acar, Christian Stransky, Dominik Wermke, Michelle L Mazurek, and Sascha Fahl. 2017. Security Developer Studies with GitHub Users: Exploring a Convenience Sample. In *Thirteenth Symposium on Usable Privacy and Security, (SOUPS 2017)*. 81–95.

[5] Omar Alrawi, Ranjita Pai Kasturi, Chaoshun Zuo, Zhiqiang Lin, Ruian Duan, and Brendan Saltaformaggio. 2019. The betrayal at Cloud City: An empirical analysis of cloud-based mobile backends. In *Proceedings of the 28th USENIX*

Security Symposium. 551–566.

[6] Maureen Jane Angen. 2000. Evaluating Interpretive Inquiry: Reviewing the Validity Debate and Opening the Dialogue. *Qualitative Health Research* 10, 3 (May 2000), 378–395. https://doi.org/10.1177/104973230001000308

[7] Renana Arizon-Peretz, Irit Hadar, and Gil 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. https://doi.org/10.1109/TSE.2021.3119721

[8] Renana Arizon-Peretz, Irit Hadar, Gil Luria, and Sofia Sherman. 2021. Understanding developers privacy and security mindsets via climate theory. *Empirical Software Engineering* 26, 6 (2021), 34.

[9] Hala Assal and Sonia Chiasson. 2018. Security in the Software Development Lifecycle. In *Proceedings of the Fourteenth USENIX Conference on Usable Privacy and Security*. USENIX Association, 281–296.

[10] Hala Assal and Sonia Chiasson. 2019. "Think secure from the beginning": A Survey with Software Developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 1–13. https://doi.org/10.1145/3290605.3300519

[11] Christine Barry. 2026. Cybercrime in 2026: Faster, smarter and fully industrialized. https://blog.barracuda.com/2026/01/02/cybercrime-in-2026--faster--smarter-and-fully-industrialized. *Barracuda* (2026).

[12] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[13] Virginia Braun and Victoria Clarke. 2020. One size fits all? What counts as quality practice in (reflexive) thematic analysis? *Qualitative Research in Psychology* 18, 3 (2020), 328–352. https://doi.org/10.1080/14780887.2020.1769238

[14] Virginia Braun and Victoria Clarke. 2021. To saturate or not to saturate? Questioning data saturation as a useful concept for thematic analysis and sample-size rationales. *Qualitative Research in Sport, Exercise and Health* 13, 2 (2021), 201–216. https://doi.org/10.1080/2159676X.2019.1704846

[15] Tony Burgess. 2026. Malware Brief: When the supply chain becomes the attack surface. https://blog.barracuda.com/2026/03/05/malware-brief-supply-chain-attack-surface. *Barracuda* (2026).

[16] David Byrne. 2022. A worked example of Braun and Clarke's approach to reflexive thematic analysis. *Quality & Quantity* 56, 3 (2022), 1391–1412. https://doi.org/10.1007/s11135-021-01182-y

[17] Yuchen Chen, Weisong Sun, Chunrong Fang, Zhenpeng Chen, Yifei Ge, Tingxu Han, Quanjun Zhang, Yang Liu, Zhenyu Chen, and Baowen Xu. 2025. Security of Language Models for Code: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* (Aug. 2025). https://doi.org/10.1145/3735554

[18] Victoria Clarke and Virginia Braun. 2017. Thematic analysis. *The Journal of Positive Psychology* 12, 3 (2017), 297–298. https://doi.org/10.1080/17439760.2016.1262613

[19] John W. Creswell and Dana L. Miller. 2000. Determining Validity in Qualitative Inquiry. *Theory Into Practice* 39, 3 (Aug. 2000), 124–130. https://doi.org/10.1207/s15430421tip3903_2

[20] Anastasia Danilova, Alena Naiakshina, and Matthew Smith. 2020. One size does not fit all: A grounded theory and online survey study of developer preferences for security warning types. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 136–148. https://doi.org/10.1145/3377811.3380387

[21] AJ Dellinger. 2026. Cast Adrift, Meta Employees Have No Idea Who the 'Token Legend' Is Anymore. https://gizmodo.com/cast-adrift-meta-employees-have-no-idea-who-the-token-legend-is-anymore-2000744189. *Gizmodo* (2026).

[22] Dorothy E. Denning. 2015. Toward more secure software. *Commun. ACM* 58, 4 (2015), 24–26. https://doi.org/10.1145/2736281

[23] Deutschland Federal Office for Information Security. 2026. What are Critical Infrastructures? https://www.bsi.bund.de/EN/Themen/Regulierte-Wirtschaft/Kritische-Infrastrukturen/Allgemeine-Infos-zu-KRITIS/allgemeine-infos-zu-kritis_node.html.

[24] Tiago Espinha Gasiba, Iosif Andrei-Cristian, Ulrike Lechner, and Maria Pinto-Albuquerque. 2021. Raising Security Awareness of Cloud Deployments using Infrastructure as Code through CyberSecurity Challenges. In *The 16th International Conference on Availability, Reliability and Security*. ACM, 1–8. https://doi.org/10.1145/3465481.3470030

[25] Tiago Espinha Gasiba, Ulrike Lechner, Maria Pinto-Albuquerque, and Daniel Méndez. 2021. Is Secure Coding Education in the Industry Needed? An Investigation Through a Large Scale Survey. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 241–252. https://doi.org/10.1109/ICSE-SEET52601.2021.00034

[26] Sead Fadilpašić. 2024. Ivanti Pulse Secure was using decade-old Linux and outdated libraries — no wonder it was such a popular target for hackers. https://www.techradar.com/pro/security/ivanti-pulse-secure-was-using-decade-old-linux-and-outdated-libraries-no-wonder-it-got-hacked. *techradar* (2024).

[27] Alessandro Fedele and Cristian Roner. 2022. Dangerous games: A literature review on cybersecurity investments. *Journal of Economic Surveys* 36, 1 (2022), 157–187. https://doi.org/10.1111/joes.12456

[28] Rui Fernandes, Nuno Lopes, Joaquim Gonçalves, and John Cosgrove. 2025. Comparing Traditional Hacking Tools and AI-Driven Alternatives. In *2025 13th International Symposium on Digital Forensics and Security (ISDFS)*. 1–5. https://doi.org/10.1109/ISDFS65363.2025.11012027

[29] Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. 2017. Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security. In *2017 IEEE Symposium on Security and Privacy (SP)*. 121–136. https://doi.org/10.1109/SP.2017.31

[30] Lorenzo Franceschi-Bicchierai. 2026. VPN flaws allowed Chinese hackers to compromise dozens of Ivanti customers, says report. https://techcrunch.com/2026/02/23/vpn-flaws-allowed-chinese-hackers-to-compromise-dozens-of-ivanti-customers-says-report/. *TechCrunch* (2026).

[31] Jen Frost. 2024. Merck settles $1.4 billion cyberattack case against insurers. https://www.insurancebusinessmag.com/us/news/cyber/merck-settles-1-4-billion-cyberattack-case-against-insurers-471908.aspx. *Insurance Business* (2024).

[32] Kelsey R. Fulton, Anna Chan, Daniel Votipka, Michael Hicks, and Michelle L. Mazurek. 2021. Benefits and Drawbacks of Adopting a Secure Programming Language: Rust as a Case Study. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. USENIX Association, 597–616.

[33] Kelsey R. Fulton, Daniel Votipka, Desiree Abrokwa, Michelle L. Mazurek, Michael Hicks, and James Parker. 2022. Understanding the How and the Why: Exploring Secure Development Practices through a Course Competition. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1141–1155. https://doi.org/10.1145/3548606.3560569

[34] Dan Goodin. 2024. Backdoor found in widely used Linux utility targets encrypted SSH connections. https://arstechnica.com/security/2024/03/backdoor-found-in-widely-used-linux-utility-breaks-encrypted-ssh-connections/. *ars Technica* (2024).

[35] Dan Goodin. 2026. Overrun with AI slop, cURL scraps bug bounties to ensure “intact mental health”. https://arstechnica.com/security/2026/01/overrun-with-ai-slop-curl-scraps-bug-bounties-to-ensure-intact-mental-health/. *ars Technica* (2026).

[36] Lawrence A. Gordon, Martin P. Loeb, William Lucyshyn, and Lei Zhou. 2014. Externalities and the Magnitude of Cyber Security Underinvestment by Private Sector Firms: A Modification of the Gordon-Loeb Model. *Journal of Information Security* 6, 11 (2014), 24–30. https://doi.org/10.4236/jis.2015.61003

[37] Peter Leo Gorski, Yasemin Acar, Luigi Lo Iacono, and Sascha Fahl. 2020. Listen to Developers! A Participatory Design Study on Security Warnings for Cryptographic APIs. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 1–13. https://doi.org/10.1145/3313831.3376142

[38] Matthew Green and Matthew Smith. 2016. Developers are Not the Enemy! The Need for Usable Security APIs. *IEEE Security and Privacy* 14, 5 (2016), 40–46.

[39] Egon G Guba. 1981. Criteria for assessing the trustworthiness of naturalistic inquiries. *ECTJ* 29, 2 (1981), 75–91.

[40] Julie Haney, Mary Theofanos, Yasemin Acar, and Sandra Spickard Prettyman. 2018. “We make it a big deal in the company”: Security Mindsets in Organizations that Develop Cryptographic Products. In *Proceedings of the Fourteenth Symposium on Usable Privacy and Security*. USENIX Association, 357–373.

[41] Julie M. Haney and Wayne 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.

[42] Sumair Ijaz Hashmi, Shafay Kashif, Lea Grober, Katharina Krombholz, and Mobin Javed. 2026. Mapping the Cloud: A Mixed-Methods Study of Cloud Security and Privacy Configuration Challenges. In *Network and Distributed System Security (NDSS) Symposium 2026*.

[43] Qing Hu, Tamara Dinev, Paul Hart, and Donna Cooke. 2012. Managing Employee Compliance with Information Security Policies: The Critical Role of Top Management and Organizational Culture*. *Decision Sciences* 43, 4 (2012), 615–660. https://doi.org/10.1111/j.1540-5915.2012.00361.x

[44] Jan Jancar, Marcel Fourné, Daniel De Almeida Braga, Mohamed Sabt, Peter Schwabe, Gilles Barthe, Pierre-Alain Fouque, and Yasemin Acar. 2022. “They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*. 632–649. https://doi.org/10.1109/SP46214.2022.9833713

[45] Connor Jones. 2026. January blues return as Ivanti coughs up exploited EPMM zero-days. https://www.theregister.com/2026/01/30/ivanti_epmm_zero_days/. *The Register* (2026).

[46] Tim Keary. 2026. Is The Cult Of ‘Tokenmaxxing’ Just Another Fad Or The New Normal? https://www.forbes.com/sites/timkeary/2026/04/13/is-the-cult-of-tokenmaxxingjust-another-fad-or-the-new-normal/. *Forbes* (2026).

[47] Iacovos Kirlappos, Simon Parkin, and M. Angela Sasse. 2014. Learning from “Shadow Security:” Why Understanding Non-Compliant Behaviors Provides the Basis for Effective Security. In *Proceedings 2014 Workshop on Usable Security*. Internet Society, San Diego, CA. https://doi.org/10.14722/usec.2014.23007

[48] Jan H. Klemmer, Stefan Albert Horstmann, Nikhil Patnaik, Cordelia Ludden, Cordell Burton, Carson Powers, Fabio Massacci, Akond Rahman, Daniel Votipka, Heather Richter Lipford, Awais Rashid, Alena Naiakshina, and Sascha Fahl. 2024. Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3658644.3690283

[49] Ravie Lakshmanan. 2026. CISA Flags SolarWinds, Ivanti, and Workspace One Vulnerabilities as Actively Exploited. https://thehackernews.com/2026/03/cisa-flags-solarwinds-ivanti-and.html. *The Hacker News* (2026).

[50] Enrique Larios-Vargas, Omar Elazhary, Soroush Yousefi, Derek Lowlind, Michael L. W. Vliek, and Margaret-Anne Storey. 2023. DASP: A Framework for Driving the Adoption of Software Security Practices. *IEEE Transactions on Software Engineering* (2023), 1–29. https://doi.org/10.1109/TSE.2023.3235684

[51] Yvonna S. Lincoln and Egon G. Guba. 1985. *Naturalistic Inquiry*. SAGE.

[52] Luigi Lo Iacono and Peter Leo Gorski. 2017. I Do and I Understand. Not Yet True for Security APIs. So Sad. In *Proceedings 2nd European Workshop on Usable Security*. Internet Society, 11. https://doi.org/10.14722/eurousec.2017.23015

[53] Tamara Lopez, Helen Sharp, Thein Tun, Arosha Bandara, Mark Levine, and Bashar Nuseibeh. 2019. 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, 7 pages. https://doi.org/10.1109/CESSER-IP.2019.00014

[54] Tamara Lopez, Helen Sharp, Thein Tun, Arosha Bandara, Mark Levine, and Bashar Nuseibeh. 2022. Security Responses in Software Development. *ACM Transactions on Software Engineering and Methodology* (2022).

[55] Jessica Lyons. 2025. Ivanti makes dedicated fans of Chinese spies who just can't resist attacking its buggy kit. https://www.theregister.com/2025/05/23/ivanti_chinese_spies_attack/. *The Register* (2025).

[56] Yunlong Lyu, Yixuan Tang, Peng Chen, Tian Dong, Xinyu Wang, Zhiqiang Dong, and Hao Chen. 2026. "Tab, Tab, Bug": Security Pitfalls of Next Edit Suggestions in AI-Integrated IDEs. (2026). https://doi.org/10.48550/arXiv.2602.06759

[57] Gary McGraw. 2004. Software security. *IEEE Security and Privacy* 2, 5 (2004), 80–83. https://doi.org/10.1109/MSECP.2004.1281254

[58] Jose Andre Morales, Thomas P. Scanlon, Aaron Volkmann, Joseph Yankel, and Hasan Yasar. 2020. 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, 8. https://doi.org/10.1145/3407023.3409186

[59] Micah R Musser. 2025. Software Torts and Software Contracts: Reframing the Developer's Duty. *NYUL Rev.* 100 (2025), 1666.

[60] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, and Matthew Smith. 2020. On Conducting Security Developer Studies with CS Students: Examining a Password-Storage Study with CS Students, Freelancers, and Company Developers. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 1–13. https://doi.org/10.1145/3313831.3376791

[61] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, and Matthew Smith. 2019. "If you want, I can store the encrypted password": A Password-Storage Field Study with Freelance Developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 1–12. https://doi.org/10.1145/3290605.3300370

[62] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. 2017. Why Do Developers Get Password Storage Wrong? A Qualitative Usability Study. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 311–328. https://doi.org/10.1145/3133956.3134082

[63] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. 2018. Deception Task Design in Developer Password Studies: Exploring a Student Sample. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. USENIX Association, 297–313. https://www.usenix.org/conference/soups2018/presentation/naiakshina

[64] Leysan Nurgalieva, Alisa Frik, and Gavin Doherty. 2023. A Narrative Review of Factors Affecting the Implementation of Privacy and Security Practices in Software Development. *Comput. Surveys* 55, 14s (2023), 320:1–320:27. https://doi.org/10.1145/3589951

[65] OpenAI. 2026. ChatGPT. https://openai.com/chatgpt.

[66] Tosin Daniel Oyetoyan, Daniela Soares Cruzes, and Martin Gilje Jaatun. 2016. An Empirical Study on the Relationship between Software Security Skills, Usage and Training Needs in Agile Settings. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*. 548–555. https://doi.org/10.1109/ARES.2016.103

[67] Tosin Daniel Oyetoyan, Bisera Milosheska, Mari Grini, and Daniela Soares Cruzes. 2018. Myths and facts about static application security testing tools: An action research at Telenor Digital. In *Agile Processes in Software Engineering and Extreme Programming: 19th International Conference, XP 2018*. Springer International Publishing, 86–103.

[68] Hernan Palombo, Armin Ziaie Tabari, Daniel Lende, Jay Ligatti, and Xinming Ou. 2020. An Ethnographic Understanding of Software (In)Security and a Co-Creation Model to Improve Secure Software Development. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, 205–220.

[69] Penligent. 2026. Log4Shell CVE Still Matters in 2026 — What CVE-2021-44228 Taught Us About Dependency RCE, Detection, and Proof-Based Remediation. https://www.penligent.ai/hackinglabs/log4shell-cve-still-matters-in-2026-what-cve-2021-44228-taught-us-about-dependency-rce-detection-and-proof-based-remediation/.

[70] Olgierd Pieczul, Simon Foley, and Mary Ellen Zurko. 2017. Developer-centered security and the symmetry of ignorance. In *Proceedings of the 2017 New Security Paradigms Workshop*. ACM Press, 46–56. https://doi.org/10.1145/3171533.3171539

[71] Scott Piper. 2024. Setting secure defaults on AWS and avoiding misconfigurations. https://www.wiz.io/blog/how-to-set-secure-defaults-on-aws.

[72] Andreas Poller, Laura Kocksch, Katharina Kinder-Kurlanda, and Felix Anand Epp. 2016. First-time Security Audits as a Turning Point? Challenges for Security Practices in an Industry Software Development Team. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1288–1294. https://doi.org/10.1145/2851581.2892392

[73] Vyom Ramani. 2026. What is tokenmaxxing: A game employees are playing to show how much AI they use. https://www.digit.in/features/general/what-is-tokenmaxxing-a-game-employees-are-playing-to-show-how-much-ai-they-use.html. *digit.in* (2026).

[74] Irum Rauf, Marian Petre, Thein Tun, Tamara Lopez, Paul Lunn, Dirk van der Linden, John Towse, Helen Sharp, Mark Levine, Awais Rashid, and et al. 2021. The Case for Adaptive Security Interventions. *ACM Transactions on Software Engineering and Methodology* 31, 1 (2021), 9:1–9:52.

[75] Irum Rauf, Dirk van der Linden, Mark Levine, John Towse, Bashar Nuseibeh, and Awais Rashid. 2020. Security but Not for Security's Sake: The Impact of Social Considerations on App Developers' Choices. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. ACM, 141–144. https://doi.org/10.1145/3387940.3392230

[76] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2022. Insecure Software on a Fragmenting Internet. In *2022 Cyber Research Conference - Ireland (Cyber-RCI)*. https://doi.org/10.1109/Cyber-RCI55324.2022.10032675

[77] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2023. 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.

[78] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2023. Unhelpful assumptions in software security research. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3576915.3623122

[79] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2024. Training Developers to Code Securely: Theory and Practice. In *2024 IEEE/ACM 5th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*.

[80] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2025. "ImmediateShortTerm3MthsAfterThatLOL": Developer Secure-Coding Sentiment, Practice and Culture in Organisations. In *2025 IEEE/ACM 47th International Conference on Software Engineering - Software Engineering in Practice (ICSE-SEIP)*.

[81] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. 2026. Weak Programmers Need Not Apply, LLMs Welcome! Survey Screening in the AI Era. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26)*. ACM, 13 pages. https://doi.org/10.1145/3744916.3787789

[82] Ita Ryan, Klaas-Jan Stol, and Utz 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. https://doi.org/10.1109/EnCyCriS59249.2023.00010

[83] Ita Ryan, Klaas-Jan Stol, and Utz 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. https://doi.org/10.1109/EnCyCriS59249.2023.00008

[84] Michael D. Scott. 2008. Tort Liability for Vendors of Insecure Software: Has the Time Finally Come? *Maryland Law Review* 62, 2 (2008). https://papers.ssrn.com/abstract=1010069

[85] B. Smith. 2018. Generalizability in qualitative research: misunderstandings, opportunities and recommendations for the sport and exercise sciences. *Qualitative Research in Sport, Exercise and Health* 10, 1 (2018), 137–149. https://doi.org/10.1080/2159676X.2017.1393221

[86] Klaas-Jan Stol, Paris Avgeriou, Muhammad Ali Babar, Yan Lucas, and Brian Fitzgerald. 2014. Key factors for adopting inner source. *ACM Transactions on Software Engineering and Methodology* 23, 2 (April 2014), 1–35. https://doi.org/10.1145/2533685

[87] Klaas-Jan Stol and Brian Fitzgerald. 2018. The ABC of Software Engineering Research. *ACM Transactions on Software Engineering and Methodology* 27, 3 (2018), 1–51. https://doi.org/10.1145/3241743

[88] Mohammad Tahaei and Kami Vaniea. 2019. A Survey on Developer-Centred Security. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 129–138.

[89] Mohammad Tahaei, Kami Vaniea, Konstantin (Kosta) Beznosov, and Maria K Wolters. 2021. Security Notifications in Static Analysis Tools: Developers' Attitudes, Comprehension, and Ability to Act on Them. In *Proceedings of the 2021*

CHI Conference on Human Factors in Computing Systems. ACM, 1–17. https://doi.org/10.1145/3411764.3445616
[90] Tyler W. Thomas, Madiha Tabassum, Bill Chu, and Heather Lipford. 2018. Security During Application Development: An Application Security Expert Perspective. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada). ACM, 1–12. https://doi.org/10.1145/3173574.3173836
[91] Inger A. Tondel, Martin Gilje Jaatun, and Daniela Soares Cruzes. 2020. IT Security Is From Mars, Software Security Is From Venus. *IEEE Security Privacy* 18, 4 (2020), 48–54. https://doi.org/10.1109/MSEC.2020.2969064
[92] Kevin Townsend. 2024. Edge Devices: The New Frontier for Mass Exploitation Attacks. https://www.securityweek.com/edge-devices-the-new-frontier-for-mass-exploitation-attacks/. *Security Week* (2024).
[93] Charlotte Trueman and Jon Gold. 2023. Tech layoffs in 2023: A timeline. https://www.computerworld.com/article/3685936/tech-layoffs-in-2023-a-timeline.html. *Computerweek* (2023).
[94] Anwesh Tuladhar, Daniel Lende, Jay Ligatti, and Xinming Ou. 2021. 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*. 617–631.
[95] Inger Anne Tøndel, Daniela Soares Cruzes, Martin Gilje Jaatun, and Kalle Rindell. 2019. The Security Intention Meeting Series as a Way to Increase Visibility of Software Security Decisions in Agile Development Projects. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES)*. ACM.
[96] Inger A. Tøndel, Daniela S. Cruzes, Martin G. Jaatun, and Guttorm Sindre. 2022. Influencing the security prioritisation of an Agile software development project. *Computers and Security* 118, C (2022). https://doi.org/10.1016/j.cose.2022.102744
[97] European Union. 2022. *Directive (EU) 2022/2555 of the European Parliament and of the Council (NIS 2 Directive)*. EU. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32022L2555&qid=1779100207079
[98] European Union. 2024. *Regulation (EU) (Cyber Resilience Act) EEA 2024 2847 2024*. EU. https://eur-lex.europa.eu/eli/reg/2024/2847/oj/eng
[99] Dirk van der Linden and Awais Rashid. 2018. The Effect of Software Warranties on Cybersecurity. *ACM SIGSOFT Software Engineering Notes* 43, 4 (2018).
[100] Daniel Votipka, Kelsey R Fulton, James Parker, Matthew Hou, Michelle L Mazurek, and Michael Hicks. 2020. Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It. In *Proceedings of the 29th USENIX Security Symposium*. USENIX Association, 109–126.
[101] Charles Weir, Ingolf Becker, James Noble, Lynne Blair, M. Angela Sasse, and Awais Rashid. 2020. Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers. *Software - Practice and Experience* 50, 3 (2020), 275–298.
[102] Joe Wilkins. 2026. Amazon Employees Forced to Hit Quotas on AI Use, Immediately Start Using it for Everything Except Work. https://futurism.com/artificial-intelligence/amazon-quotas-ai-use. *Futurism* (2026).
[103] World Economic Forum. 2026. Global Cybersecurity Outlook 2026. https://www.weforum.org/publications/global-cybersecurity-outlook-2026/in-full/3-the-trends-reshaping-cybersecurity/.
[104] Glenn Wurster and P. C. van Oorschot. 2008. The developer is the enemy. In *Proceedings of the 2008 New Security Paradigms Workshop*. ACM, 89–97. https://doi.org/10.1145/1595676.1595691
[105] Shundan Xiao, Jim Witschey, and Emerson Murphy-Hill. 2014. Social influences on secure development tool adoption: Why security tools spread. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, 1095–1106. https://doi.org/10.1145/2531602.2531722
[106] Michael Zhang, Glyn Atwal, and Maya Kaiser. 2021. Corporate social irresponsibility and stakeholder ecosystems: The case of Volkswagen Dieselgate scandal. *Strategic Change* 30, 1 (2021), 79–85. https://doi.org/10.1002/jsc.2391

A Protocol for Interview Questions

These are the questions asked during the interviews.

A.1 All Participants: Opening Question

As you know, this interview is being recorded. Is that OK with you?

A.2 Questions for Developers

A.2.1 Demographic information.

- What country do you live in?
- What job title and role do you have in your current organisation?
- How many years of development experience do you have?

- Do you have other relevant experience?
- What kind of software do you mostly work on?
- What technologies do you use?

This section is for solo developers only

- You are self-employed, is that right?
- Do you ever work embedded within organisations?
- Do you ever work with other software developers?

This section is for developers in organisations only

- Are you contract or permanent?
- What kind of environment do you currently work in, big company or small? Multinational?
- Is it a software organisation?
- What is the organisation's main focus?
- How many other software developers do you work with regularly?

A.2.2 Working environment information. **Read this definition to the interviewee**: Software security is the ability of software to withstand malicious attack. Good ability to withstand attack is achieved by considering security when coding software.

- What do you think are the main security risks to the software you work on?
- Do you try to code securely?

If yes:

- What motivates you to code securely? How did you start?
- Do you code security tools, or do you focus on writing code securely, or both?
- Do you feel that you succeed in making your code secure? How do you check?

This section is for solo developers only

- Where do you look for support to write secure code?
- Do your clients ask about code security?

This section is for developers in organisations only

- Do you generally feel supported to write secure code?
- Have you ever been asked to focus on functionality and ignore secure coding for a while?
- Would you consider yourself to be a software security driver in your organisation?
- How is that manifested?
- How is it viewed – by colleagues – by management?

If no:

- If you wanted to write secure code, would you have support to do that?

A.2.3 Information on specific practices. (Not asked if participant has already indicated that there are no software security practices in their coding environment.)

I have a list of secure coding practices here. Can you tell me whether you have encountered them in your coding environment? (For solo developers: can you tell me whether you use them?)

- (**Solo not asked**): Secure coding procedures that must be followed
- A written security checklist
- Use of security tools
  - Do you think there is overhead attached to using them?
  - (**Solo**): How did you find them? Did you find setup difficult? Why?
    (**Others**): Did you introduce any yourself? Was that difficult? Why?

- Automation such as DevSecOps. What are the benefits and drawbacks to this?
- A policy on keeping dependencies such as libraries up-to-date? (**Solo**): Do you follow it? (**Others**): Is it followed?
- (**Solo**): Secure coding training or research
  (**Others**): Secure coding training, or support for secure coding research if you are already expert
- (**Solo**): Discussion of code security aspects with others
  (**Others**): Frequent discussion of code security aspects (more than once a week)
- Methodical consideration of code security when a project begins
- (**Solo**): Taking time to do a good job on code security, in your opinion
  (**Others**): Enough time allocated to do a good job on code security, in your opinion

This section is for developers in organisations only:

- Rewards or thanks for doing a good job on code security
- Adverse personal consequences if you ignore code security
- Does your immediate manager seem to care about software security?

All developers:

- Do you feel you've learned anything about software security from your development role?
- Do software security requirements get in the way of getting your job done?
- Are there other security aspects in your workplace that you'd like to mention? (Example if asked, requirements evaluations or code reviews that consider security).
- As far as you know, has code you wrote ever caused a security breach? Do you want to tell us about it?
- (**Solo**): As far as you know, has a company or team you dealt with ever experienced a security breach? Explain.
  (**Others**): As far as you know, has your company or team ever experienced a security breach? Explain.
- Do you work with open source? What are the differences in secure coding practice in that environment?
- What do you think would be the best possible improvement to support you [**not solo**: and your team] to write secure code? At any level; industry, organisation or tool.
- Do you work with cloud services? If so, do you find that there are any security implications?
- What do you feel is the biggest barrier to writing secure code at the moment?

This section is for developers in organisations only

- Do you think that the importance you attach to software security is the same as the importance your organisation attaches to it?
- What do you think is the attitude of top management at your company – like the CEO, or the C-Suite – to software security?
- Do you have any theories as to why they have that attitude?
- Are there inconsistencies between what they say and what they do?
- Do you have any thoughts on a difference in attitude between contract and permanent employees when it comes to software security?

A.3 Questions for consultants with industry knowledge

- What is your current job title?
- What is your role in your current organisation?
- Approximately what size is your organisation?

- What role do you have in relation to software security?
- Software security is different from general cybersecurity, because it involves taking steps during the software development process to ensure that the code produced is secure. Do you think that the organisations you work with prioritise software security?
  If no:
  – Why not?
  If yes:
  – What do you think are the drivers for this?
- How old is the drive for software security in the industry?
- What do you think is the attitude of top management at most organisations – like the CEO, or the C-Suite – to software security?
- Do you have any theories as to why they have that attitude?
- Are there inconsistencies between what they say and what they do with regards to software security?
- Speaking about organisations in general, what one or two developments do you think could push software security higher up the agenda for all management C-suites? (If they are not sure what I mean: For example, the ability for ransomwared customers to sue / removal of insurance support for ransomware recovery costs).
- Is your assessment of the importance of software security aligned to that of your organisation?
- What do you think is the biggest barrier to developers and organisations in writing secure code?
- Have you encountered any security implications to working with cloud services?
- Have you encountered issues with the open source supply chain? How are they addressed?
- What do you think is the most important thing a management team can contribute to software security within the organisation?
- The BSIMM authors suggest that historically, a drive towards secure software has only succeeded when top-down – i.e., when the main instigator is in management. In recent years they've said that this is changing and that an engineering-led software security drive now has a chance of success within an organisation, probably due to DevSecOps. What is your opinion on this?
- As far as you know, has a company or team you worked with ever experienced a security breach? Explain.

A.4 Questions for senior managers

- What is your current job title?
- What role do you have in relation to software security in your current organisation?
- Approximately what size is your organisation?
- How many developers are there in the organisation?
- Software security is different from general cybersecurity, because it involves taking steps during the software development process to ensure that the code produced is secure. Does your organisation prioritise software security?
  If no:
  – Why not?
  If yes:
  – What do you think are the drivers for this?
- How old is the drive for software security in your organisation?
- Speaking about organisations in general, what one or two developments do you think could push software security higher up the agenda for all management C-suites? (If they are not

sure what I mean: For example, the ability for ransomwared customers to sue / removal of insurance support for ransomware recovery costs).
- Is your assessment of the importance of software security aligned to that of your organisation?
- Would you consider yourself to be a software security driver in your organisation?
- What do you think would be the best possible improvement to help you to support your team in writing secure code?
- What do you think is the biggest barrier to you supporting your team in writing secure code?
- What has been your experience of software security in previous roles?
- What do you think is the most important thing a management team can contribute to software security within the organisation?
- The BSIMM authors suggest that historically, a drive towards secure software has only succeeded when top-down – i.e., when the main instigator is in management. In recent years they've said that this is changing and that an engineering-led software security drive now has a chance of success within an organisation, probably due to DevSecOps. What is your opinion on this?

A.5 All Participants: Broader industry

- Has organisational turbulence – downsizing, mergers, etc – ever had an effect on secure coding practice in your experience?
- Has there been any occasion in your career where you've encountered a big increase or decrease in secure coding behaviour in an organisation or team? What do you think were the causes?
- What do you think will be the impact of recent AI developments on software security?
- **(Except consultants)**: What has been your experience of software security in previous roles?
- How do you feel about secure coding in the general developer population?
- Do you think that more legislation or regulation would increase overall code security?
- Do you have anything more you would like to add about this topic?

A.5.1 Concluding Questions. Thanks for doing this interview. How did you find the experience? Did it raise any issues for you?

You may have further thoughts about this later. If you'd like to fill me in on anything else, you can contact me at the university email address in the interview document.